CP/M Plus Anwender-Handbuch CPC 6128/Joyce

Jürgen Hückstädt

CP/M Plus Anwender-Handbuch CPC 6128/Joyce

Ein unentbehrliches Nachschlagewerk für die praktische Arbeit mit CP/M Plus und seinen Hilfsprogrammen. Mit zahlreichen Beispielen und ausführlichen systemspezifischen Daten zur internen Speicherorganisation und zu Schnittstellen.

Markt&Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Hückstädt, Jürgen:
CP-M-Plus-Anwender-Handbuch CPC 6128/Joyce : e. unentbehrl. Nachschlagewerk
für d. prakt. Arbeit mit CP/M Plus u. seinen Hilfsprogrammen ;
mit zahlr. Beispielen u. ausführl. systemspezif. Daten zur internen Speicherorganisation
u. zu Schnittstellen / Jürgen Hückstädt. –
Haar bei München : Markt-und-Technik-Verlag, 1986.
ISBN 3-89090-197-2

15 14 13 12 11 10 9 8 7 6 5 4 3
89 88 87 86

ISBN 3-89090-197-2

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

Inhaltsverzeichnis

	Vorwort	9
1	**Einführung**	11
1.1	Was ist CP/M?	12
1.2	Geschichtliche Entwicklung	17
1.3	Gerätezusammenstellung	19
1.3.1	Bildschirm und Tastatur (Konsole)	19
1.3.2	Diskettenlaufwerk	20
1.3.3	Drucker	22
1.4	Wichtige Grundbegriffe für Anfänger	23
1.4.1	Der CP/M-Start	23
1.4.2	Die drei Grundelemente von CP/M	25
1.5	CP/M 2.2 und CP/M Plus	28
2	**Allgemeine Grundlagen**	31
2.1	Einige Tippversuche	31
2.2	Etwas über Dateien	35
2.3	Das Dienstprogramm DISCKIT3	43
2.4	Etwas über den Diskettenaufbau	47
2.5	Dateien kopieren	48
2.6	Residente und nichtresidente Befehle	50
2.7	Dateien umbenennen	51
2.8	Dateien löschen	52
2.9	Mehrere Befehle gleichzeitig	53
2.10	Das Arbeiten mit gekaufter Software	53
3	**Spezielles über CP/M Plus**	55
3.1	Datum und Uhrzeit	55
3.2	Directory-Label und -Timestamps	57
3.3	Passwords und Timestamps für Dateien	60
3.4	Schreibschutz für Laufwerk	66
3.5	Dateiattribute	67
3.6	Nützliches über den SET-Befehl	68
3.7	Benutzerbereiche	69
3.8	Mehr über DIR und SHOW	70
3.9	Gerätezuordnung	75
3.10	Zeichensatz und Tastaturbelegung 77	
3.11	Bildschirmfarben	80
3.12	HELP	81

4	**Der Editor**	83
4.1	Was ist ein Editor?	83
4.2	Das Dienstprogramm ED	85
4.2.1	Textdatei anlegen	85
4.2.2	Zeilen einfügen/löschen	88
4.2.3	Text korrigieren	92
5	**PIP - Ein universelles Kopierprogramm**	95
5.1	Allgemeines	95
5.2	Laden und Aktivieren von PIP	96
5.3	Kopieren von Diskette zu Diskette	97
5.4	Dateien aneinanderhängen	99
5.5	Datenaustausch mit anderen Peripheriegeräten	101
5.6	Spezielle Befehle	104
5.6.1	Dateiausschnitte	104
5.6.2	Groß- und Kleinschrift	105
5.6.3	Textausgabe mit Zeilennummern	106
5.6.4	Verify	107
5.6.5	Kopieren geschützter Dateien	107
5.6.6	Zusammenfassen binärer Dateien	107
6	**Stapelverarbeitung**	109
6.1	Allgemeines	109
6.2	SUBMIT	109
6.3	Übergabe von Parametern an Programme	113
6.4	Die Datei PROFILE.SUB	114
7	**Debugging**	119
7.1	DUMP	119
7.2	SID	122
7.2.1	SID laden	122
7.2.2	SAVE	123
7.2.3	Speicher ansehen	125
7.2.4	Speicher ändern	126
7.2.5	Assembler/Disassembler	128
7.2.6	Weitere Möglichkeiten	128
8	**Assemblerprogrammierung**	131
8.1	Ein paar Worte vorweg	131
8.2	Der Quelltext	132
8.3	Quelltext assemblieren	136

9	**Hinter den Kulissen**	141
9.1	Interne Speicherorganisation	141
9.2	Directory und File-Control-Block (FCB)	143
9.3	BDOS-Aufrufe	146
9.4	Mehr über das BIOS	156
9.5	Die Nullseite	160
9.6	Residente System-Erweiterungen (RSX)	161
9.7	System-Control-Block (SCB)	161
9.8	Anpassen von CP/M-Software	163
10	**CP/M-Befehlsübersicht**	165
	AMSDOS	166
	BASIC	167
	C10CPM3	168
	COPYSYS	169
	DATE	170
	DEVICE	171
	DIR/DIRSYS	173
	DISCKIT/DISCKIT3	177
	DUMP	178
	ED	179
	ERA	188
	GENCOM	189
	GET	191
	HELP	192
	HEXCOM	193
	INITDIR	194
	J12DCPM3	195
	LANGUAGE	196
	LIB	197
	LINK	199
	MAC	201
	PALETTE	202
	PAPER	204
	PATCH	205
	PIP	206
	PROFILE.SUB/PROFILE.ENG/PROFILE.GER	214
	PUT	215
	REN	217
	RMAC	218
	SAVE	219
	SET	220
	SET24X80	226

SETDEF 227
SETKEYS 229
SETLST 230
SETSIO 231
SHOW 233
SID 234
SUBMIT 244
TYPE 246
USER 247
XREF 248

Anhang 249

Stichwortverzeichnis 253

Übersicht weiterer Markt&Technik-Bücher 257

Vorwort

Dieses Buch ist für alle Schneider CPC 6128- und JOYCE-Benutzer geschrieben, die sich mit CP/M Plus beschäftigen. Obwohl der CPC 6128 in erster Linie ein BASIC-Computer und der JOYCE ein reiner Bürocomputer ist, können doch beide Modelle auch unter CP/M Plus betrieben werden. Dadurch entstehen weitere ungeahnte Anwendungsmöglichkeiten. CP/M Plus ist nämlich ein universelles Betriebssystem, das auf zahlreichen Computern implementiert ist. Deshalb steht auch eine große Anzahl von CP/M-Programmen zur Verfügung, mit denen Sie jetzt auch auf Ihrem Schneider-Computer arbeiten können.

Selbst wenn Ihnen das Wort CP/M im Augenblick noch ein Fremdwort ist, mit dem Sie nicht viel anfangen können, werden Sie spätestens nach der Durcharbeitung der ersten beiden Kapitel in der Lage sein, CP/M-Programme anzuwenden, die es für Ihren Computer zu kaufen gibt. Die Zahl der fertig für den CPC 6128 und JOYCE angebotenen Programme nimmt ständig zu und schon jetzt gibt es die Spitzenprogramme WordStar (Textverarbeitung), dBase II (Dateiverwaltung) und MultiPlan (Tabellenkalkulation) zu einem sehr günstigen Preis im Handel.

Die nachfolgenden Kapitel sind so aufgebaut, daß sie sowohl für den Einsteiger als auch für den Profi, der vielleicht schon mit CP/M 2.2 auf anderen Computern gearbeitet hat, einen unerläßlichen Begleiter darstellen. Sie benötigen zwar keinerlei Grundkenntnisse, es wäre jedoch von Vorteil, wenn Sie sich bereits mit den einfachsten Grundlagen der BASIC-Programmierung auskennen würden. Wir versuchen nämlich, den Übergang von BASIC zu CP/M in einer leichtverständlichen Form zu vollziehen.

Aber spätestens, wenn Sie die wichtigsten Grundbegriffe über CP/M Plus kennen, werden Sie tiefer in das CP/M-Betriebssystem einsteigen und die letzten Möglichkeiten aus ihm herausholen wollen. So erfahren Sie im dritten Kapitel einiges darüber, wie Sie Ihre CP/M-Programme vor unberechtigtem Zugriff anderer mit einem Password schützen können.

In den darauffolgenden Kapiteln lernen Sie den CP/M-Texteditor ED und das Kopierprogramm PIP richtig anzuwenden, denn beides gehört zur Standardausrüstung eines jeden CP/M-Computers. Als weiteres erfahren Sie etwas über die Stapelverarbeitung, mit deren Hilfe Sie eine Vielzahl von Einzelanweisungen durch Abarbeiten einer Befehlsdatei ausführen können. Falls Sie sich näher für das Innere des CP/M Betriebssystems interessieren, finden Sie in den Kapiteln 7 bis 9 Wissenswertes über Assemblerprogrammierung, Debugging und CP/M-Standardroutinen.

Das Buch schließt mit einer ausführlichen CP/M-Befehlsübersicht mit zahlreichen Beispielen. Besonders dem Profi steht somit noch zusätzlich ein Nachschlagewerk zur Verfügung, aus dem er sich alle benötigten Informationen schnell beschaffen kann.

Viel Freude beim Studium dieses Buches und beim Arbeiten mit CP/M Plus wünscht Ihnen

Der Autor

1 Einführung

Lieber Leser, als stolzer Besitzer eines Schneider CPC 6128 oder JOYCE PCW 8256 haben Sie sich zusätzlich dieses Buch gekauft, um den Umgang mit CP/M Plus zu erlernen. Vielleicht haben Sie sich Ihren Computer einzig und allein deshalb zugelegt, weil Sie sich ausschließlich mit CP/M Plus beschäftigen möchten. Sie bringen bereits einige Grundkenntnisse mit oder haben sogar auf einem anderen Computer schon mit CP/M gearbeitet. Jetzt möchten Sie Ihre Kenntnisse vertiefen und interessieren sich für die systemspezifischen Eigenschaften des CPC 6128 oder JOYCE.

Wahrscheinlich gehören Sie aber zu dem weitaus größeren Anwenderkreis, der sich einen Computer zunächst aus einem anderen Grund zugelegt hat. Falls Sie einen CPC 6128 besitzen, mag Sie in erster Linie das leistungsstarke BASIC interessiert haben, das mit seinen vielen Graphik- und Soundbefehlen unzählige Möglichkeiten bietet. Sollten Sie aber einen JOYCE erworben haben, geschah dies wahrscheinlich aus beruflichen Gründen. Hierbei handelt es sich nämlich um ein voll ausgebautes System mit Bildschirm, Drucker und Floppylaufwerk, das in erster Linie für die Textverarbeitung entwickelt wurde und eine äußerst preiswerte Komplettlösung für so manches Büro darstellt. Beim Kauf erfuhren Sie vielleicht nebenbei, daß Ihr Computer auch CP/M-fähig ist, worunter Sie sich zunächst nichts Konkretes vorstellen konnten. Erst im Laufe der Zeit wurde Ihr Interesse für CP/M geweckt, nachdem Sie erfahren hatten, daß es sich dabei um ein universelles Betriebssystem handelt, für das eine große Anzahl fertiger Programme erhältlich ist. Ganz gleich, ob Sie nun zu der einen oder der anderen Gruppe gehören, dieses Buch ist für Sie geschrieben, um Sie mit allen notwendigen Informationen zu versorgen, die Sie zum Arbeiten mit CP/M Plus benötigen.

Bevor wir uns nun näher mit CP/M Plus befassen, beachten Sie bitte den folgenden wichtigen Hinweis: Für Schneider-Computer gibt es zwei verschiedene CP/M Versionen, nämlich CP/M 2.2 und CP/M Plus. Letztere ist eine Weiterentwicklung von CP/M 2.2, die auch als CP/M 3.0 bezeichnet wird. Während der JOYCE nur mit CP/M Plus arbeitet, wird der CPC 6128 mit beiden Versionen geliefert. Dieses Buch behandelt aber ausschließlich CP/M Plus, das viele Vorzüge gegenüber der Version 2.2 aufweist. Deshalb sollten Sie normalerweise auch nur mit CP/M Plus auf diesem Computer arbeiten, denn die meisten Programme, die unter CP/M 2.2 entwickelt wurden, laufen auch unter CP/M Plus. Da lediglich wenige Ausnahmen CP/M 2.2 benötigen, wollen wir uns mit dieser Version hier nicht näher befassen. Weitere Einzelheiten darüber finden Sie im CP/M 2.2 Anwenderhandbuch für den CPC 464, 664 und 6128, das vom gleichen Verfasser stammt und im Markt&Technik Verlag erschienen ist.

1.1 Was ist CP/M?

Falls Sie zu der zweiten angesprochenen Anwendergruppe gehören, können Sie sich vermutlich unter dem Begriff CP/M noch nicht allzuviel vorstellen und Sie wissen nicht so recht, was Sie damit anfangen sollen.

Die meisten von Ihnen haben sich bestimmt schon mit BASIC befaßt, einer leicht zu erlernenden Programmiersprache, die im allgemeinen standardmäßig auf den Home- und Personal-Computern implementiert ist. Beim CPC 6128 steht sie sofort nach dem Einschalten des Computers zur Verfügung, während sie beim JOYCE erst von Diskette nachgeladen werden muß. Wenn Sie mit BASIC arbeiten, erhalten Sie ein Programm, das Sie sofort ausführen können. Sie tippen nur die entsprechenden BASIC-Zeilen korrekt ein und nach dem Befehl RUN läuft das Programm ab.

Ganz so einfach geht es mit CP/M allerdings nicht, denn CP/M ist keine Programmiersprache, sondern lediglich ein Betriebssystem. Um dies zu verstehen, müssen wir uns zunächst einmal in groben Zügen mit dem Aufbau des Computersystems beschäftigen. Verlieren Sie aber nicht gleich den Mut! Die folgenden Erläuterungen sind so abgefaßt, daß sie für einen Laien bzw. Computerneuling leicht verständlich sind. Falls Sie sich bereits mit CP/M auskennen, können Sie die folgenden Absätze überspringen.

Wir werden jetzt anhand einiger Grundlagen der BASIC-Programmierung versuchen, den Übergang von BASIC zu CP/M zu vollziehen.

Wenn Sie den CPC 6128 einschalten, erscheint eine Einschaltmeldung, das Wort "Ready" und ein gelbes Kästchen (Cursor) auf dem Bildschirm. Sie können dann sofort BASIC-Programme eingeben oder von Kassette bzw. Diskette laden und ausführen.

Dies kommt aber nicht von ungefähr: Selbst wenn der Computer noch kein Programm enthält, arbeitet er dennoch unaufhörlich und führt ohne Unterbrechung interne Maschinenroutinen aus. Ähnliches gilt übrigens auch für den JOYCE, der in Wartestellung geht, bis Sie eine CP/M- oder Loco-Script-Diskette eingelegt haben.

Der Computer setzt sich aus einer Vielzahl von Bausteinen zusammen. Der wichtigste ist die CPU, ein Z80A-Mikroprozessor, der ständig arbeitet und interne Maschinenroutinen ausführt. Da er den Computer sozusagen "am Leben erhält", kann man ihn auch als sein "Herz" betrachten. Darüber hinaus wird die CPU noch von einigen anderen Bausteinen unterstützt, die besondere Aufgaben, meist für Steuerzwecke, übernehmen. Darunter fallen Bausteine, die den Datentransfer mit der Floppy oder Kassette abwickeln,

den Drucker ansteuern sowie ein Videochip für die Bildschirmausgabe oder ein Soundchip zur Tonerzeugung, um nur einige Beispiele zu nennen.

Nicht zu vergessen sind die Speicherbausteine, die entweder fest vorprogrammierte Routinen enthalten (ROM) oder flüchtig sind (RAM). So sind in den ROMs sämtliche Routinen gespeichert, die der Computer nach dem Einschalten benötigt, damit Sie BASIC-Programme direkt eingeben und abarbeiten können. Aus einem ROM kann man also nur Daten lesen, aber keine hineinschreiben. ROMs bezeichnet man deshalb auch als Nur-Lese-Speicher, aus denen man die fest vorprogrammierten Informationen, die auch beim Abschalten des Computers nicht verloren gehen, lesen kann.

Beim RAM dagegen handelt es sich um einen Schreib-Lese-Speicher, d.h. man kann Daten hier hineinschreiben und auch wieder daraus lesen. Beim Ausschalten des Computers gehen sie allerdings verloren. Im RAM werden beispielsweise Ihre BASIC-Programme oder die Zeichen abgelegt, die Sie jeweils auf dem Bildschirm sehen.

Sämtliche Routinen und Daten, mit denen der Computer arbeitet, sind in 8-Bit-Einheiten abgelegt. Ein Bit ist die kleinste Informationseinheit, die Sie sich auch als Schalter vorstellen können, der entweder ein- oder ausgeschaltet ist. Entsprechend nimmt ein Bit den Zustand 0 (aus) oder 1 (ein) an.

Jeweils acht Bit zusammengefaßt ergeben eine Speicherzelle im Computer, die man als Byte bezeichnet. Wenn nun Ihr CPC 6128 128K RAM (JOYCE 256K RAM) enthält, so heißt dies, daß er über

128 * 1024 = 131072 Byte

bzw.

256 * 1024 = 262144 Byte

Schreib-Lesespeicher verfügt, die allerdings nicht alle für BASIC-Programme nutzbar sind. Der Buchstabe K steht hier für Kilobyte, das in der Computertechnik nicht 1000, sondern 1024 Byte umfaßt.

Da sich jedes Byte aus 8 Bit zusammensetzt, kann es somit 2^8 oder 256 verschiedene Zustände annehmen. Um dies zu verdeutlichen, wollen wir hier ein paar Beispiele zur binären Zahlendarstellung betrachten:

0000 0000

ergibt den Wert Null, da sämtliche 8 Bits nicht gesetzt sind. Dagegen ergibt

```
1111 1111
```

den Wert 255, der sich folgendermaßen errechnet (von links nach rechts):

```
  1 * 128
+ 1 * 64
+ 1 * 32
+ 1 * 16
+ 1 * 8
+ 1 * 4
+ 1 * 2
+ 1 * 1
--------
    255
```

Da hier sämtliche Bits gesetzt sind, ist 255 der größte Wert, der in einem Byte dargestellt werden kann. Darüber hinaus gibt es aber noch Zwischenwerte, wie im folgenden Beispiel, das die Zahl 114 binär darstellt:

```
0111 0010
```

oder

```
  0 * 128
+ 1 * 64
+ 1 * 32
+ 1 * 16
+ 0 * 8
+ 0 * 4
+ 1 * 2
+ 0 * 1
--------
    114
```

Will man nun größere Zahlenwerte als 255 darstellen, so benötigt man zwei oder mehr Bytes, die zu einer Einheit zusammmengefaßt werden. Zwei Bytes können somit 256 * 256 = 65536 oder drei Bytes 256 * 256 * 256 = 16777216 verschiedene Zustände annehmen. Von dieser Möglichkeit macht z.B. BASIC Gebrauch, um Fließkommazahlen intern abzulegen, wenn auch in einer etwas abgewandelten Form.

Mit den 256 verschiedenen Zuständen eines Bytes kann man aber noch andere Informationen außer Zahlen erfassen, was jedoch von Fall zu Fall vereinbart werden muß. Dies geschieht beispielsweise zur Darstellung von Buchstaben und Satzzeichen, wobei jedem Zeichen ein entsprechender binärer Wert zugeordnet ist (ASCII-Code). Auch Maschinenprogramme setzen sich aus einer Vielzahl von Binärwerten zusammen, wobei jeder Wert einer bestimmten Operation entspricht, die jeweils zwischen eins und vier Bytes benötigt. Die restlichen Bytes beinhalten dann z.B. Werte, die für diese Operation erforderlich sind.

Sie sehen, daß man in den einzelnen Bytes im Speicher eine Vielzahl von Informationen ablegen kann, wobei natürlich in jedem Fall feststehen muß, ob es sich um eine Fließkommazahl, um Textzeichen oder um Maschinenprogramme handelt. Selbst ein BASIC-Programm setzt sich aus Bytewerten zwischen 0 und 255 zusammen.

Damit Sie nun direkt nach dem Einschalten des CPC 6128 BASIC-Programme eingeben und ausführen können, müssen folgende interne Routinen ausgeführt bzw. aktiviert werden:

o Betriebssystem

o Editor

o BASIC-Interpreter

Das Betriebssystem ist der eigentliche Kern des Computers und arbeitet auf unterer Ebene. Es fragt z.B. die Tastatur ab, welche Taste Sie gerade gedrückt haben, steuert den Datentransfer zu Drucker und Floppy, sendet Videosignale an den Bildschirm oder erzeugt Töne über den Lautsprecher. Dazu gehören selbstverständlich auch die bereits erwähnten Zusatzbausteine für bestimmte Aufgaben.

Mit dem Betriebssystem allein können Sie aber noch kein BASIC-Programm schreiben und ausführen. Sie können sich höchstens eigene Maschinenprogramme schreiben, die bestimmte Anweisungen an das Betriebssystem geben. Das ist sicher eine interessante Aufgabe für den Profi, der sich mit der direkten Programmierung der Z80-CPU oder den anderen Bausteinen auskennt und umfangreiche Kenntnisse in der Assemblerprogrammierung mitbringen muß.

In den frühen Zeiten der Computertechnik bediente man sich solcher Programmiertechniken, die allerdings sehr aufwendig waren. Bald jedoch entwickelte man höhere Programmiersprachen, die das Programmieren erheblich vereinfachten. Zu diesen Programmiersprachen gehört auch das BASIC Ihres Computers.

Eine solche Programmiersprache bewirkt im Prinzip nichts anderes, als einfache Anweisungen - die, wie z.B. in BASIC, der üblichen algebraischen Schreibweise sehr nahe kommen - *in eine Vielzahl von Maschinenbefehlen* umzusetzen und diese durch das Betriebssystem ausführen zu lassen.

Kommen wir wieder auf den CPC 6128 zurück: Zunächst muß der BASIC-Programmtext, so wie Sie ihn eingeben, erfaßt und intern gespeichert werden. Dies ist die Aufgabe des Editors, der letztlich ebenfalls aus einem umfangreichen Maschinenprogramm besteht, mit dessen Hilfe Sie jede

BASIC-Zeile eintippen und die Eingabe durch Drücken der RETURN-Taste beenden können. Dabei wird der BASIC-Text in einen speziellen Code umgewandelt, der zwar noch weit vom eigentlichen Maschinencode entfernt ist, aber als BASIC-Programm abgearbeitet werden kann.

Die Abarbeitung übernimmt dann der BASIC-Interpreter, ebenfalls ein umfangreiches Maschinenprogramm, welches diesen Code liest und die zugehörigen Maschinenroutinen des Betriebssystems aufruft.

Nur durch das Zusammenwirken von Editor, Interpreter und Betriebssystem können wir auf so einfache Weise mit BASIC arbeiten. Dabei brauchen wir uns über die Arbeitsweise dieser Routinen keinerlei Gedanken zu machen. Nebenbei sei bemerkt, daß es auch Compiler gibt, die den Programmtext direkt in ausführbaren Maschinencode übersetzen. Dies hat den Vorteil, daß die Programme dann wesentlich schneller ablaufen, da ein Interpreter nicht mehr benötigt wird. Nachteilig dagegen wirkt sich ein erhöhter Speicherplatzbedarf und die Tatsache aus, daß ein compiliertes BASIC-Programm nicht mehr gelistet und korrigiert werden kann.

Nachdem wir nun die Arbeitsweise des Computers unter BASIC in groben Zügen kennengelernt haben, fällt es nicht mehr schwer, auch CP/M zu verstehen. CP/M ist nämlich ein universelles platten- oder diskettenorientiertes Betriebssystem, das an fast jeden Computer angepaßt werden kann, der eine Z80- oder 8080-CPU besitzt. Programme, die auf einem CP/M-Computer erstellt wurden, sind - von geringen Anpassungen einmal abgesehen - auf jedem anderen CP/M-Computer ebenfalls lauffähig. Es ist daher nicht verwunderlich, daß es eine Unmenge von CP/M-Software auf dem Markt gibt, die auf jedem CP/M-Computer lauffähig ist. Nicht umsonst wird behauptet, daß CP/M-Programme die größte Softwarebibliothek der Welt darstellen.

CP/M-Programme sind meist reine Maschinenprogramme, die auf dem CP/M-Betriebssystem lauffähig sind und genormte Einsprungadressen verwenden. Um mit CP/M arbeiten zu können, ist es aber nicht unbedingt notwendig, daß Sie sich in der Maschinenprogrammierung auskennen, wenn Sie auf fertige Programme zurückgreifen. Darüber hinaus werden zu CP/M noch verschiedene Dienstprogramme mitgeliefert, die das Arbeiten erleichtern.

Unter CP/M können Sie in den verschiedensten Programmiersprachen programmieren, wenn Sie sich den jeweiligen Editor und Interpreter bzw. Compiler kaufen. So gibt es auch M-BASIC (eine BASIC-Version, die dem Mallard-BASIC des JOYCE sehr ähnelt, die sich aber vom BASIC des CPC 6128 in einigen Punkten unterscheidet) oder Programmiersprachen wie z.B.

Turbo-PASCAL, FORTRAN und FORTH, um nur einige Beispiele zu nennen.

Die Renner unter CP/M sind allerdings professionelle Programme, wie WordStar, ein komfortables Textverarbeitungssystem, dBase II, ein Datenbankprogramm und MultiPlan, ein Tabellenkalkulationsprogramm. Diese Programme sind, fertig angepaßt für den CPC 6128 und JOYCE, bei Markt&Technik erhältlich.

Wenn Sie andere CP/M-Programme von anderen CP/M-Rechnern auf Ihren Computer übertragen möchten, müssen Sie diese überspielen und eventuell anpassen. Dazu benötigen Sie jedoch umfangreichere Kenntnisse, weshalb wir dieses Thema erst an späterer Stelle in diesem Buch anschneiden werden. Zunächst jedoch ist es wichtig, daß Sie mit fertig angepaßter CP/M-Software und den mitgelieferten Dienstprogrammen arbeiten können.

1.2 Geschichtliche Entwicklung

Anfangs der siebziger Jahre steckte die Mikrocomputer-Technik noch in den Kinderschuhen. Der 8080 Mikroprozessor, der Vorgänger des Z80, der sich in den Schneider-Computern befindet, war seinerzeit die neueste Errungenschaft. In der damaligen Zeit waren Computer noch relativ teuer und besaßen für unsere heutigen Begriffe nur einen kleinen RAM-Speicher.

Schon bald entstand der Wunsch, für alle Computer, die mit einem 8080 und später einem Z80 Prozessor ausgestattet waren, ein einheitliches Betriebssystem zu entwickeln, das sich mit wenig Mühe an die jeweilige Hardware (Computertyp) anpassen ließ.

Ein Angestellter der Firma Intel Corporation, namens Gary Kildall, entwickelte schließlich im Jahre 1974 die erste CP/M-Version, die aber gegenüber den heute verwendeten Versionen noch recht primitiv war. Er arbeitete seinerzeit an einer Dateiverwaltung, die er in der höheren Programmiersprache PL/M schrieb und verwendete CP/M lediglich zur Unterstützung.

Im Jahre 1975 kam die erste kommerziell angebotene CP/M-Version 1.4 auf den Markt, die zunächst aber wenig Beachtung fand. Darüber hinaus war sie nur auf den genormten 8-Zoll-IBM-Diskettenlaufwerken lauffähig und konnte nur schwer an andere Diskettenformate angepaßt werden. Die heute weit verbreiteten 5.25-Zoll- und 3-Zoll-Disketten gab es ja damals noch nicht.

Dieser Zustand änderte sich erst, als um 1979/80 die erweiterte CP/M-Version 2.2 herauskam und von der Firma Digital Research vertrieben wurde. Jetzt war CP/M ein universelles Betriebssystem, das auf die verschiedensten Diskettenformate angepaßt werden konnte. Dies hatte zur Folge, daß CP/M bald eine große Verbreitung fand. Somit konnten, von geringen Anpassungen einmal abgesehen, auf einem CP/M-Rechner entwickelte Programme leicht auf andere CP/M-fähige Computer übertragen werden. Es ist daher auch nicht verwunderlich, daß CP/M-Programme die größte Softwarebibliothek der Welt darstellen.

Obwohl auch heute noch die Version 2.2 als Standard gilt, wurde CP/M zwischenzeitlich weiter entwickelt. Daraus entstand für 8-Bit-Rechner die Version 3.0, auch CP/M Plus genannt, die mehr als 64 KByte RAM-Speicher benötigt. Mit eben dieser Version werden wir uns in diesem Buch ausführlich beschäftigen.

Auch für 16-Bit-Rechner gibt es CP/M-Versionen, die aber nicht sehr weit verbreitet sind. Für die mit einem 8086- bzw. 8088- Prozessor ausgestatteten IBM-PC-Computer und deren kompatiblen Geräte entstand CP/M 86, das sich allerdings nicht gegen MS-DOS durchsetzen konnte. Darüber hinaus sind auch Computer mit einem 68000-Prozessor (z.B. Apple-Macintosh) CP/M-fähig, wobei die Version CP/M-68K Anwendung findet.

Neben CP/M gibt es noch das Betriebssystem MP/M. MP/M ist CP/M sehr ähnlich und enthält noch einige Zusatzfunktionen. Während CP/M ein Einbenutzersystem ist, ist MP/M multitasking-fähig, d.h. es kann mehrere Aufgaben gleichzeitig erfüllen. MP/M wird überall dort eingesetzt, wo mehrere Anwender über verschiedene Konsolen (in der Regel Tastatur und Bildschirm) auf einen Computer zugreifen. Außerdem hat ein einzelner Benutzer auch die Möglichkeit, mehrere Aufgaben gleichzeitig durchführen zu lassen, wie z.B. einen Text zu editieren und einen anderen auszudrucken.

Auch für MP/M gibt es eine 16-Bit-Version, die Concurrent CP/M 86 heißt. Unter dieser Version können sogar MS-DOS-Programme eingesetzt werden.

1.3 Gerätezusammenstellung

Damit ein Computersystem unter CP/M arbeiten kann, muß es sich zumindest aus folgenden Geräte-Einheiten zusammensetzen:

o Konsole (Bildschirm und Tastatur)

o Diskettenlaufwerk

o Drucker

1.3.1 Bildschirm und Tastatur (Konsole)

Bildschirm und Tastatur werden bei CP/M meist zusammengefaßt und als Konsole bezeichnet, obwohl es sich eigentlich um zwei getrennte Geräte handelt. Die Tastatur kann sowohl der amerikanischen als auch der deutschen Norm angepaßt sein, was für die Funktionstüchtigkeit des Systems jedoch ohne Bedeutung ist.

Der CPC 6128 wird mit einer amerikanischen QWERTY-Tastatur geliefert, die keine deutschen Sonderzeichen enthält. Zur Bedienung des CP/M-Betriebssystems ist dies auch nicht unbedingt erforderlich. Wenn Sie jedoch mit einem Textverarbeitungs-Programm deutschsprachige Texte eingeben, ist eine deutsche DIN-Tastatur sehr zu empfehlen. Man kann zwar die deutschen Umlaute auch umschreiben, indem man z.B. für den Buchstaben "ä" "ae" setzt, ein solcher Text wäre aber sehr ungewöhnlich für uns, so daß Sie diese Methode nur im äußersten Notfall einsetzen sollten.

Der CPC 6128 läßt sich leicht auf den deutschen Zeichensatz umstellen. Davon macht z.B. das Textverarbeitungs-Programm WordStar Gebrauch. In seiner für den CPC 6128 angepaßten und von Markt&Technik vertriebenen Version können Sie zwischen dem amerikanischen dem deutschen Zeichensatz wählen. Dabei werden nicht nur die Tastatur entsprechend angepaßt, sondern gleichzeitig auch die deutschen Sonderzeichen auf dem Bildschirm erzeugt. Da hierbei einige Tasten ihre ursprüngliche Bedeutung verlieren, empfiehlt es sich, entsprechende Tastenaufkleber herzustellen.

Der JOYCE wird dagegen bereits werksmäßig mit einer deutschen DIN-Tastatur geliefert, so daß hier keine Umstellung erforderlich ist. Beachten Sie aber, daß bei CP/M häufig eckige Klammern benötigt werden, die auf einer deutschen Tastatur nicht vorhanden sind. Hier ist dann das Zeichen "[" durch "Ä" und das Zeichen "]" durch "Ü" zu ersetzen.

1.3.2 Diskettenlaufwerk

Zum Arbeiten mit CP/M benötigen Sie zumindest ein Diskettenlaufwerk. Obwohl dies im Prinzip für die meisten Zwecke ausreicht, ist ein zweites Laufwerk sehr zu empfehlen, da es sich damit wesentlich bequemer arbeiten läßt. Sowohl der CPC 6128 als auch der JOYCE enthalten bereits ein eingebautes Laufwerk mit der Bezeichnung "A", sehen aber die Möglichkeit vor, ein weiteres Laufwerk B anzuschließen. Darüber hinaus enthält der JOYCE noch ein virtuelles Laufwerk mit 112 KByte Speicherplatz, das die Bezeichnung "M" trägt und genauso wie ein echtes Diskettenlaufwerk zu bedienen ist. Die Daten werden hier aber nicht auf Diskette, sondern im RAM des Computers abgelegt und gehen natürlich nach beim Abschalten des Gerätes verloren. Das virtuelle Laufwerk eignet sich besonders als temporärer Zwischenspeicher, beispielsweise wenn Dateien mit nur einem echten Laufwerk kopiert werden sollen.

Beachten Sie, daß der CPC 6128 und JOYCE jeweils verschiedene Disketten-Aufzeichnungsformate verwenden. Deshalb können Sie mit dem CPC 6128 keine Disketten lesen, die im JOYCE-Format beschrieben sind. Dagegen kann der JOYCE Disketten im CPC-Format lesen und auch beschreiben. So können Sie ohne weiteres auf dem JOYCE Dateien vom einen in das andere Format übertragen, beispielsweise mit dem Kopierprogramm PIP, das wir noch näher kennenlernen werden.

Das 3-Zoll-Floppylaufwerk, das bei den Schneider-Computern Verwendung findet, gehört zu den kleinsten und kompaktesten Laufwerken, die es heute gibt. Dabei ist es genauso leistungsfähig wie seine größeren Brüder mit 8 Zoll und 5 ¼ Zoll.

Wenn Sie eine 3-Zoll-Diskette neben eine 5-¼-Zoll-Diskette legen, fällt Ihnen nicht nur der Größenunterschied auf, sondern auch die Schutzhülle. Die eigentliche Diskette besteht nämlich aus einer empfindlichen Magnetscheibe, die gegen mechanische Verletzungen unbedingt geschützt werden muß!

Die Hülle der 5-¼-Zoll-Disketten ist sehr biegsam und besitzt zwei Öffnungen für den Schreib-/Lesekopf und das Indexloch. Es verwundert daher nicht, daß der Umgang mit solchen Disketten besonderer Sorgfalt bedarf. Auch sollten die Disketten nur zum Gebrauch aus ihrer Papierhülle, die die Öffnungen schützt, herausgenommen werden.

Die für die Schneider-Floppies verwendeten 3-Zoll-Disketten befinden sich in einem stabilen Plastikgehäuse, das bei weitem nicht so empfindlich ist, wie die Hülle der 5-¼-Zoll-Disketten. Zusätzlich werden die Öffnungen für den Schreib-/Lesekopf und das Indexloch mit einer Metallklappe ver-

schlossen, die sich beim Einschieben der Diskette in das Laufwerk automatisch öffnet. An der Seite des Gehäuses befindet sich nämlich eine Führungsschiene mit einem weißen Schieber. Wenn Sie diesen Schieber (vorsichtig!) betätigen, spannen Sie eine Feder und öffnen gleichzeitig die Klappe, wobei die Magnetscheibe sichtbar wird. Beim Loslassen des Schiebers schnappt die Klappe aufgrund der Federwirkung wieder zu.

Die 3-Zoll-Disketten sind beidseitig beschreibbar. Sie können aber immer nur eine Seite, nämlich Seite "A" oder Seite "B", gleichzeitig nutzen. Das Laufwerk greift jeweils auf die Seite zu, die beim Einschieben oben auf der Diskette sichtbar ist. Beim JOYCE ist es immer die linke Seite, da hier das Laufwerk senkrecht angeordnet ist.

Das Einlegen der Diskette in das Laufwerk ist denkbar einfach. Sie schieben die Diskette so weit hinein, bis sie einrastet. Dies sollte aber nur dann geschehen, wenn sowohl der Computer, als auch die Floppy eingeschaltet sind, da es sonst unter ungünstigen Umständen zu Datenverlusten kommen kann.

Das Herausnehmen der Diskette ist genauso einfach: Sie drücken lediglich den Knopf rechts unter der Öffnung und die Diskette springt heraus.

Abgesehen von dem Schreibschutz für CP/M-Dateien, mit dem wir uns noch befassen werden, können Sie Ihre Disketten vor versehentlichem Schreiben schützen, wenn Sie links oben auf jeder Seite das kleine, mit einer roten Klappe verschlossene Loch öffnen. Dies erreichen Sie, indem Sie mit einem spitzen Stift den seitlich angebrachten roten Schieber betätigen.

Ungeachtet des Schreibschutzes sollten Sie von allen wichtigen Programmen und Dateien mindestens eine Sicherheitskopie anfertigen, die sich nach Möglichkeit auf einer anderen Diskette befinden sollte. Trotz der hohen Aufzeichnungssicherheit ist es dennoch nicht völlig auszuschließen, daß Schreib- oder Lesefehler auftreten. Doch wahrscheinlicher als rein technische Fehler ist das "menschliche Versagen", wodurch unbeabsichtigt eine Diskette neu formatiert oder wichtige Dateien gelöscht oder überschrieben werden. Auch in einem solchen Fall kann man dann auf die Sicherheitskopie zurückgreifen.

Im weiteren Verlauf dieses Buches werden wir uns mit dem Kopieren von Dateien und ganzen Disketten noch näher beschäftigen.

Einige Fremdfirmen bieten 5-¼-Zoll-Laufwerke für Schneider-Computer an, die entweder nur als Zweitlaufwerk (Laufwerk B) eingesetzt werden

können oder zusammen mit einem eigenen Betriebssystem geliefert werden und somit die Schneider-Laufwerke vollständig ersetzen.

Es stellt sich allerdings die Frage, welchen Nutzen diese Laufwerke bringen. Der Einsatz von 5-¼-Zoll-Disketten allein bringt keinen Vorteil, wie wir bereits gesehen haben. Solche Laufwerke sind nur dann sinnvoll, wenn Sie Daten mit einem anderen Computer austauschen möchten, dessen Floppy ein IBM-kompatibles Disketten-Aufzeichnungsformat verwendet. Neben dem IBM PC gibt es noch eine Reihe anderer Computer, die mit diesem Format arbeiten oder es zumindest lesen können. Die CPC-Floppy arbeitet normalerweise zwar nicht im IBM-Format, sie kann aber Disketten in diesem Format lesen und beschreiben.

In absehbarer Zeit werden sicher auch Plattenlaufwerke (Harddisc) für Schneider-Computer auf den Markt kommen, die anstelle der Diskettenlaufwerke eingesetzt werden können. Im Gegensatz zu einer Diskette enthalten sie eine Festplatte, die meist nicht austauschbar ist. Harddiscs haben eine wesentlich größere Speicherkapazität als Disketten, die meist im Rahmen zwischen 10 und 20 Megabyte (1 MByte = 1024 KByte) liegt. Dafür sind sie auch wesentlich teurer.

1.3.3 Drucker

Jedes CP/M-System sollte auch einen Drucker enthalten. Der Drucker NLQ-401 von Schneider ist ein leistungsfähiger und preisgünstiger Matrixdrucker für den CPC 6128, mit dem Sie auch Korrespondenzschrift (NLQ = Near Letter Quality) erzeugen können. Darüber hinaus kann er mit Hilfe von DIP-Schaltern auch auf den deutschen Zeichensatz umgeschaltet werden, was sich besonders für eine deutsche Textverarbeitung vorteilhaft auswirkt. Beim JOYCE dagegen ist ein ähnlicher Drucker bereits im Lieferumfang mit enthalten.

Matrixdrucker besitzen einzelne Nadeln, die für jedes Zeichen ein Punktraster auf das Papier drucken. In der NLQ-Betriebsart werden die einzelnen Zeichen nach einem bestimmten Schema zweimal überdruckt, wodurch ein sehr sauberes Schriftbild entsteht, das dem eines Typenraddruckers (siehe unten) sehr nahe kommt.

Außer dem NLQ 401 können Sie aber auch jeden anderen Drucker an den CPC 6128 anschließen, der mit einer Centronics-Schnittstelle ausgerüstet ist. Dies ist bei den meisten handelsüblichen Druckern der Fall, die zudem auch den deutschen Zeichensatz enthalten. Der CPC 6128 ist bereits standardmäßig mit einer Centronics-Schnittstelle versehen, während für den JOYCE ein entsprechendes Interface erhältlich ist. Dieses Interface besitzt

außer einer Centronics- noch eine RS232-Schnittstelle, die zur Datenfernübertragung erforderlich ist.

Ein besonders schönes Druckbild erzeugen die Typenraddrucker, bei denen keine Punktraster, wie bei Matrixdruckern, sondern Typen, die auf einem rotierenden Rad angeordnet sind, für die einzelnen Zeichen gedruckt werden. Ein Typenraddrucker arbeitet aber wesentlich langsamer als ein Matrixdrucker. Sie sollten sich jedoch ernsthaft die Frage stellen, ob sich ein Typenraddrucker für Ihre Zwecke wirklich lohnt, denn die NLQ-Matrixdrucker erzeugen bereits ein sehr schönes Schriftbild, dessen Qualität derjenigen eines Typenraddruckers schon sehr nahe kommt.

Als weiteres gibt es noch Tintenstrahldrucker, die ähnlich wie Matrixdrucker arbeiten. Das Zeichenraster wird hier aber nicht von Nadeln, sondern von feinen Farbdüsen erzeugt, die kleine Farbtupfer auf das Papier spritzen.

Die neueste Errungenschaft sind die Laserdrucker, die die Druckfarbe mittels eines Laserstrahls auf das Papier übertragen und dort einbrennen. Diese Drucker arbeiten zwar extrem schnell, dafür ist ihr Anschaffungspreis aber noch sehr hoch.

1.4 Wichtige Grundbegriffe für Anfänger

Wir befassen uns jetzt mit wichtigen Grundlagen, die Sie unbedingt durchlesen sollten, bevor Sie mit CP/M arbeiten. Dabei spielt es keine Rolle, ob Sie einen CPC 6128 oder JOYCE besitzen, denn das Arbeiten mit CP/M Plus ist prinzipiell für beide Rechner gleich. Etwaige gerätespezifische Unterschiede werden jedoch erläutert.

1.4.1 Der CP/M-Start

Der CP/M-Start, d.h. das Booten bzw. Laden des CP/M-Betriebssystems, ist für den CPC 6128 und den JOYCE unterschiedlich. Während Sie auf dem CPC 6128 direkt nach dem Einschalten BASIC-Programme schreiben oder ausführen können, enthält der JOYCE nur einen Bootstrap-Lader, der das gesamte Betriebssystem, um welches es sich dabei auch immer handeln mag, von Diskette laden muß. Bevor dies nicht geschehen ist, kann man mit dem JOYCE absolut nichts anfangen.

Beginnen wir mit dem CPC 6128. Achten Sie darauf, daß alle drei Steckverbindungen zwischen der Konsole und dem Bildschirm richtig hergestellt sind. Falls Sie ein zweites (externes) Laufwerk verwenden, verbinden Sie dieses noch vor dem Einschalten mit einem entsprechenden Flachkabel, das

Sie in die dafür vorgesehene Buchse des Grundgerätes einstecken. Jetzt schalten Sie zuerst das Zweitlaufwerk (falls vorhanden) und dann den Computer ein.

Dem CPC 6128 liegen beim Kauf zwei Systemdisketten mit insgesamt vier Seiten bei, deren Nummer jeweils auf dem Etikett der Diskette aufgedruckt ist. Legen Sie jetzt Seite 1 in das eingebaute Laufwerk und geben Sie dann

|CPM <RETURN>

ein. Durch diesen Vorgang wird CP/M Plus geladen und initialisiert. Statt der RETURN-Taste könnten Sie auch die ENTER-Taste verwenden, denn die Funktion beider ist vollkommen identisch. Im folgenden sprechen wir aber immer nur von der RETURN-Taste.

Jetzt erscheint die Einschaltmeldung

```
CP/M Plus Amstrad Consumer Electronics plc
v 1.0, 61K TPA, 1 disc drive
```

und darunter das Bereitschaftszeichen

```
A>
```

Dies bedeutet, daß der Computer nun zur Entgegennahme von CP/M-Befehlen bereit ist. Ist kein externes Laufwerk angeschlossen, erscheint in der unteren rechten Ecke zusätzlich noch der Hinweis

```
Drive is A:
```

Er gibt an, daß das Laufwerk A das Bezugslaufwerk ist.

Kommen wir jetzt zum JOYCE, der ebenfalls ein eingebautes Laufwerk besitzt. Stellen Sie auch hier zunächst sämtliche Steckverbindungen zwischen dem Bildschirm, der Tastatur und dem Drucker her. Schalten Sie nun die gesamte Anlage mit dem Schalter ein, der unterhalb des Bildschirms angebracht ist. Zunächst finden Sie einen hellen Bildschirm und ein leise surrendes Laufwerk vor. Auch dem JOYCE liegen zwei Systemdisketten mit den Seiten 1 bis 4 bei. Alles, was Sie nun tun müssen, um CP/M zu starten, ist, nach dem Einschalten Seite 2 in das Laufwerk zu schieben. Jetzt wird das CP/M-Betriebssystem automatisch geladen, ohne daß Sie eine weitere Taste drücken müssen.

Beim JOYCE erscheint eine ähnliche Einschaltmeldung wie beim CPC 6128:

```
CP/M Plus Amstrad Consumer Electronics plc
v 1.2, 61K TPA, 1 Laufwerk, 112K Laufwerk M:
```

Darunter erscheint, ebenso wie beim CPC 6128, das Bereitschaftszeichen

```
A>
```

und Sie können jetzt CP/M-Befehle eingeben. Die Einschaltmeldung gibt auch an, daß wir auf dem JOYCE ein virtuelles Laufwerk "M" mit 112 KByte zur Verfügung haben, das lediglich aus RAM-Speicher besteht.

Falls der JOYCE kein zweites eingebautes Laufwerk enthält, erscheint in der rechten unteren Ecke die Meldung:

```
Laufwerk ist A:
```

Vielleicht ist Ihnen aufgefallen, daß der JOYCE teilweise deutsche Meldungen ausgibt, während sie beim CPC 6128 englisch sind. Dies ist ein Vorteil, welcher den JOYCE neben seiner DIN-Tastatur als Bürocomputer für den deutschen Sprachgebrauch auszeichnet.

1.4.2 Die drei Grundelemente von CP/M

Eingangs haben wir schon erfahren, daß CP/M ein universelles Betriebssystem ist, das an fast jeden Computer mit einem 8080- oder Z80-Mikroprozessor angepaßt werden kann.

Wir wissen auch schon, daß ein Computer aus flüchtigem RAM-Speicher und verschiedenen ROMs sowie anderen Bausteinen besteht. Der Computer, einschließlich sämtlicher Bauteile, wird als Hardware bezeichnet. Programme, gleich welcher Art, die man von Diskette nachladen muß und die im RAM-Speicher abgelegt werden, nennt man Software.

Die Hardware des Computers enthält somit auch dessen Betriebssystem, das allerdings nicht mit dem CP/M-Betriebssystem identisch ist. Weiter oben hatten wir der Einfachheit halber nur von einem Betriebssystem gesprochen, um den Aufbau eines Computersystems zu erklären.

Das CP/M-Betriebssystem dagegen besteht aus den drei Teilen CCP (Command Control Processor), BDOS (Basic Disc Operation System) und BIOS (Basic Input Output System). Das Wort "Basic" hat hier jedoch nichts mit der Programmiersprache BASIC zu tun.

Beginnen wir mit dem BDOS, dem Kernstück des CP/M-Betriebssystems. Das BDOS steuert sämtliche Diskettenoperationen, es schreibt Dateien auf Diskette, liest sie wieder und verwaltet das Disketten-Inhaltsverzeichnis (Directory). Darüber hinaus besitzt es auch Routinen zur Abfrage der Tastatur und zur Ausgabe von Zeichenketten (Strings) auf dem Bildschirm oder auf dem Drucker.

Das BDOS besteht aus geräteunabhängigen Maschinenroutinen, die für alle CP/M-Computer gleich sind. Lediglich der Speicherbereich, in dem das BDOS abgelegt ist, kann von Gerät zu Gerät unterschiedlich sein.

Das BDOS besitzt die vorteilhafte Eigenschaft, daß CP/M-Programme, die auf einem Computer entwickelt wurden, auch auf einem anderen CP/M-Computer lauffähig sind. Dies wird durch genormte Funktionsaufrufe erreicht, wobei für jede Funktion eine Konstante in das C-Register des Prozessors geladen und dann das Unterprogramm ab Adresse 5 (0005H) aufgerufen wird.

Falls Sie sich bereits in der Assemblerprogrammierung auskennen, dürfte es Ihnen nicht schwer fallen, diesen Vorgang zu verstehen. Betrachten wir ein kleines Beispiel:

Wir wollen eine Zeichenkette auf dem Bildschirm ausgeben. Dazu laden wir das C-Register mit der Konstante 9 und rufen das Unterprogramm ab der Speicherzelle 5 auf. So einfach ist das.

Für diejenigen Leser, die sich bisher nur mit BASIC befaßt haben, hier ein kleines BASIC-Programm, das die Funktionsweise der BDOS-Aufrufe simuliert.

```
10 REM BDOS-Simulation
20 INPUT "Funktionsnummer"; a
30 GOSUB 60
40 GOTO 10
50 :
60 ON a GOTO 100, 200, 300
70 :
100 REM Stringausgabe
        (Anweisungen)
150 RETURN
190 :
200 REM Abfrage Tastatur
        (Anweisungen)
250 RETURN
290 :
300 REM Datei auf Diskette schreiben
        (Anweisungen)
350 RETURN
```

Nach Eingabe der Funktionsnummer, die wir hier in der Variablen "a" ablegen, wird das Unterprogramm in Zeile 60 aufgerufen. Dortsteht die BASIC-Anweisung

```
ON a GOTO 100, 200, 300
```

die Sie vielleicht schon aus dem Handbuch oder einem BASIC-Lehrbuch kennen. Erhält dabei die Variable "a" den Wert "1", wird die Routine ab

Zeile 100 aufgerufen, beim Wert "2" ist es die Routine ab Zeile 200 und beim Wert "3" die entsprechende ab Zeile 300.

Die Zuordnung eines Wertes zur Variablen "a" entspricht dem Laden des C-Registers mit einer Funktions-Konstanten und Abarbeiten von Zeile 60, dem jeweiligen BDOS-Aufruf in Adresse 5. BDOS verzweigt dann intern in die Routine, die mit der Funktionsnummer aufgerufen wurde.

In unserem BASIC-Programm haben wir nur drei verschiedene BDOS-Aufrufe simuliert; in Wirklichkeit kann das BDOS aber über 50 verschiedene Funktionen ausführen.

Das BDOS steht in ständigem Kontakt mit dem CCP, den man auch als Bedienungsprozessor bezeichnen kann. Ebenso wie das BDOS ist auch der CCP systemunabhängig und steuert den Dialog mit dem Benutzer. So gibt z.B. der CCP das Anforderungszeichen A> aus und wartet, daß Sie einen Befehl eingeben, den er dann analysiert und ausführt.

Eine Reihe von residenten Befehlen, ist bereits im CCP enthalten. Im Gegensatz dazu gibt es auch nicht residente Befehle, zu deren Ausführung erst ein Maschinenprogramm von Diskette nachgeladen werden muß. Mehr hierüber in Kapitel 2.

Kommen wir schließlich zum BIOS, welches das Bindeglied zwischen dem BDOS und der jeweiligen Hardware darstellt. Während CCP und BDOS auf jedem CP/M-Computer identisch sind, ist dies beim BIOS nicht der Fall, denn es muß jeweils angepaßt bzw. neu geschrieben werden.

Um die BIOS-Anpassung brauchen Sie sich bei Ihrem CPC-Computer glücklicherweise nicht mehr zu kümmern, denn diese Arbeit haben Ihnen bereits die Konstrukteure des Computers abgenommen. Beim CPC 6128 ist das BIOS ein fester Bestandteil des Floppy-ROMs und verwendet teilweise die gleichen Routinen, die auch unter AMSDOS, dem Diskettenbetriebssystem für BASIC, Anwendung finden. Auch ist die Dateiverwaltung unter AMSDOS derjenigen unter CP/M weitestgehend angepaßt, weshalb Sie einige Dateien auch zwischen BASIC und CP/M austauschen können. Doch hierüber später mehr.

Ähnlich wie das BDOS enthält auch das BIOS eine Tabelle mit konstanten Einsprungadressen. Die aufgerufenen BIOS-Routinen befinden sich aber auf einer weit niedrigeren Maschinenebene als die des BDOS. So ist das BDOS beispielsweise in der Lage, eine ganze Zeichenkette auf dem Bildschirm auszugeben. Dabei ruft es wiederholt eine BIOS-Routine auf, die diese Kette jeweils zeichenweise an die Hardware weitergibt, was in diesem Fall der Bildschirm ist.

Neben den drei Hauptteilen von CP/M - CCP, BDOS und BIOS - dürfen wir aber nicht vergessen, daß ein CP/M-System auch einen Arbeitsspeicher benötigt. Er wird in der Fachsprache als TPA (Transient Program Area) bezeichnet und muß einen zusammenhängenden Speicherbereich umfassen. Er beginnt bei der Adresse 100H (256 dez.) und enthält im oberen Bereich die drei Elemente CCP, BDOS und BIOS. Je größer der TPA ist, desto umfangreichere Programme können auf dem System laufen. Auf dem CPC stehen unter CP/M 2.2 etwa 41 KByte und unter CP/M Plus (nur CPC 6128) 61 KByte an TPA zur Verfügung.

1.5 CP/M 2.2 und CP/M Plus

Falls Sie bereits auf einem anderen Computer mit CP/M 2.2 gearbeitet haben, werden Sie sicher die Unterschiede zu CP/M Plus interessieren, die wir hier kurz vorstellen wollen.

Der Hauptunterschied zwischen beiden Versionen ist der benötigte RAM-Speicher. CP/M 2.2 ist für maximal 64 KByte Speicher ausgelegt, denn mehr kann die Z80-CPU nicht adressieren. 64 KByte entsprechen genau einer Speicherbank.

CP/M Plus dagegen kann durch sogenanntes Banking auch auf mehrere Speicherbänke zugreifen. Unter Banking versteht man die Umschaltung zwischen mehreren Speicherbänken zu je 64 KByte.

Der CPC 6128 verfügt über zwei solcher Speicherbänke und der JOYCE sogar über vier, die allerdings nur teilweise für CP/M Plus reserviert sind. Wenn Sie den CPC 6128 jedoch unter CP/M 2.2 betreiben, benötigen Sie nur eine Speicherbank, ähnlich wie beim CPC 464 und 664. Deshalb können diese beiden Geräte normalerweise nicht unter CP/M Plus betrieben werden. Es gibt jedoch noch eine ungebankte CP/M-Plus-Version, für die nur eine Speicherbank erforderlich ist. Sie steht allerdings nicht für Schneider-Computer zur Verfügung und würde außerdem einen Teil des wertvollen TPA beanspruchen.

Da sowohl der CPC 6128 als auch der JOYCE in mehrere Bänke aufgeteilt sind, kann die Bank 0 als Systembank und die Bank 1 als TPA dienen. Dies bedeutet aber nicht, daß Sie unter CP/M Plus volle 64 KByte TPA zur Verfügung haben, denn ein Teil der Bank 1 wird für Teile des BDOS und BIOS benötigt. Immerhin bleiben für den Anwender noch 61 KByte verfügbar.

Die Bank 0 enthält überhaupt keinen TPA und ist für die Hauptteile des CCP, BDOS und BIOS reserviert. Der überwiegende Teil jedoch dient zur vorübergehenden Ablage von Diskettensektoren.

An dieser Stelle werden Sie sich vielleicht fragen, warum für den TPA nur eine Speicherbank zur Verfügung steht, wobei nicht allzuviel Platz mehr als in einem gut ausgelegten CP/M 2.2-System vorhanden ist. Auch die Ablage von Diskettensektoren in dem riesigen Speicher kommt Ihnen vielleicht überflüssig vor.

Wir werden jedoch gleich sehen, daß diese Anordnung erhebliche Vorteile mit sich bringt, die nicht nur in der Größe des Speicherplatzes liegen.

Nehmen wir einmal an, Sie bearbeiten eine relative Datei, die aus zahlreichen Datensätzen besteht. Steht, wie unter CP/M 2.2, nur eine Speicherbank zur Verfügung, müssen Sie zum Lesen oder Schreiben der einzelnen Datensätze jedesmal auf die Diskette zugreifen. Diese Zugriffe benötigen nicht nur extrem viel Zeit, sondern beanspruchen außerdem auch Laufwerk und Diskette.

Der Speicherbereich, in dem unter CP/M Plus Diskettensektoren abgelegt werden, ist so groß, daß er manche Datei vollständig aufnehmen kann, zumindest aber einen großen Teil davon. Jetzt ist es nicht mehr notwendig, bei jedem einzelnen Zugriff das Floppy-Laufwerk anzuwerfen, um einen Datensatz zu lesen oder zu schreiben. Aus diesem Grund können die Floppy-Operationen wesentlich schneller abgewickelt werden als unter CP/M 2.2.

CP/M Plus kann auch mehr als zwei Speicherbänke verwalten, wobei alle weiteren Speicherbänke ausschließlich zur Ablage von Diskettensektoren bzw. Dateien dienen. Dies kann sogar so weit führen, daß der Inhalt einer ganzen Diskette im RAM gespeichert wird und somit überhaupt nicht mehr auf die Diskette zugegriffen werden muß. Mit zwei Speicherbänken ist aber schon ein sehr komfortables Arbeiten möglich.

In diesem Zusammenhang spricht man auch von einer RAM-Floppy oder RAM-Disc, einem RAM-Speicher, der ein Floppylaufwerk teilweise oder ganz ersetzt, zumindest solange, wie die betreffende Datei in Arbeit ist. Der JOYCE geht sogar noch weiter, denn nahezu zwei Bänke (112 KByte) können hier als virtuelles Laufwerk M angesprochen werden.

Auch unter BASIC kann man die zweite Speicherbank des CPC 6128 als RAM-Disc nutzen, wozu das auf der Systemdiskette befindliche Programm BANKMAN dient. Mit dessen Hilfe können Sie entweder vier Bildschirminhalte oder eine relative Datei in der zweiten Speicherbank ablegen, die

Sie vor der Bearbeitung von Diskette laden und im Anschluß daran wieder als Ganzes abspeichern. Vielleicht hilft Ihnen dieses Beispiel, den Nutzen einer zweiten Speicherbank noch besser zu verstehen.

Aber damit nicht genug: In den folgenden Kapiteln befassen wir uns noch ausführlich mit den zahlreichen Dienstprogrammen, die nur unter CP/M Plus zur Verfügung stehen. Sie haben die Möglichkeit, Disketten und Dateien mit einem Schutzwort vor unberechtigtem Zugriff zu schützen oder mit einem Datums- und Zeiteintrag zu versehen. Auch in der Bedienung ist CP/M Plus wesentlich komfortabler als CP/M 2.2. Dies sehen wir an dem folgenden Beispiel:

Wenn Sie unter CP/M 2.2 einen Diskettenwechsel vornehmen, müssen Sie anschließend unbedingt CTRL-C eingeben (gleichzeitiges Drücken der CTRL- und der C-Taste), um einen Warmstart auszulösen. Dabei werden nicht nur die sytemunabhängigen Teile BDOS und CCP, sondern auch noch für den Zugriff wichtige Daten von der Diskette gelesen. Unter CP/M Plus dagegen können Sie vollkommen auf CTRL-C verzichten, da die Diskettenparameter bei jedem Zugriff automatisch gelesen werden.

Auf dem JOYCE werden Sie allerdings vergeblich eine CTRL-Taste suchen, denn ihre Aufgabe übernimmt hier die ALT-Taste.

2 Allgemeine Grundlagen

Nachdem wir im ersten Kapitel wichtige Grundbegriffe über CP/M kennengelernt haben, wollen wir jetzt einen Schritt weiter gehen und die ersten praktischen Gehversuche unternehmen. Wir wissen bereits, daß wir das CP/M-Betriebssystem zuerst laden und initialisieren müssen, bevor wir damit arbeiten können. Diesen Vorgang wollen wir hier kurz rekapitulieren. Beim CPC 6128 legen Sie nach dem Einschalten von Floppy und Computer Seite 1 der Systemdisketten in das Laufwerk A und geben dann

|CPM <RETURN>

ein. Falls Sie nur mit einem Laufwerk arbeiten, ist dies immer das Bezugslaufwerk bzw. Laufwerk A. Es erscheint die Einschaltmeldung und das Anforderungszeichen "A>", so daß Sie nun mit CP/M arbeiten können.

Beim JOYCE schalten Sie die gesamte Anlage ein und legen dann Seite 2 der Systemdisketten ins Laufwerk ein. Sollte Ihr JOYCE mit zwei Laufwerken ausgestattet sein, so benutzen Sie das obere. CP/M Plus wird jetzt automatisch geladen, ohne daß Sie sonst irgend etwas dazu tun müssen.

2.1 Einige Tippversuche

Der CCP, der den Dialog mit dem Benutzer herstellt, ist anders zu bedienen als der Editor, der unter BASIC zur Verfügung steht. In diesem Zusammenhang benötigen wir einige Steuerzeichen, die meist mit der CTRL-Taste (JOYCE ALT-Taste) erzeugt werden. Lassen Sie uns deshalb ein paar Übungen durchführen.

Wir beginnen mit einem ganz einfachen Befehl, der das Inhaltsverzeichnis (Directory) der Systemdiskette auflistet. Dies funktioniert ähnlich wie unter BASIC und AMSDOS, wo wir hierzu

|dir

eingegeben haben. Unter CP/M geschieht dies mit

A>*dir <RETURN>*

oder

A>*DIR <RETURN>*

Der Vollständigkeit halber haben wir hier das Anforderungszeichen A> mit aufgeführt, um die gesamte Befehlszeile darzustellen. Sie dürfen es aber keinesfalls nochmals eingeben, da sonst ein Fehler auftritt. Sie müssen lediglich die drei Zeichen "D", "I" und "R" eingeben und anschließend die RETURN-Taste drücken. In der Regel spielt es bei CP/M keine Rolle, ob Sie die Befehlszeile in Groß- oder Kleinbuchstaben eingeben, intern wird sie ohnehin in Großschrift umgewandelt.

Wenn Sie bisher alles richtig gemacht haben, erscheint das Directory wie folgt auf dem Bildschirm:

```
dir
A: C10CPM3 EMS : BANKMAN BAS : PROFILE  ENG : SUBMIT   COM
A: SETKEYS COM : KEYS    CCP : LANGUAGE COM : SET24X80 COM
A: PALETTE COM : SETSIO  COM : SETLST   COM : DISCKIT3 COM
A: DATE    COM : DEVICE  COM : DIR      COM : ED       COM
A: ERASE   COM : GET     COM : PIP      COM : PUT      COM
A: RENAME  COM : SHOW    COM : TYPE     COM : SET      COM
A: SETDEF  COM : AMSDOS  COM : BANKMAN  BIN : KEYS     WP
A>
```

Mit seinem Aufbau werden wir uns an später noch befassen. Jetzt ist es nur wichtig, daß Sie die Befehlsübergabe richtig beherrschen.

Beachten Sie, daß nach Ausgabe des Directories wieder das Anforderungszeichen A> und der Cursor erscheinen. CP/M ist jetzt zur Übernahme weiterer Befehle bereit.

Bisher sind wir davon ausgegangen, daß Sie alles richtig eingegeben haben. Was geschieht aber, wenn Sie sich vertippen? Lassen Sie uns dies anhand dieses einfachen Befehls einmal ausprobieren, und geben Sie dazu folgende fehlerhafte Anweisung ein:

A>*dirr <RETURN>*

Diesen Befehl versteht CP/M nicht und gibt deshalb die Meldung aus:

```
A>dirr
DIRR?

A>
```

Die fehlerhafte Anweisung erscheint in Großschrift mit angefügtem Fragezeichen und das Anforderungszeichen ist wieder sichtbar, d.h. CP/M kann einen neuen Befehl entgegennehmen.

Dies heißt aber noch lange nicht, daß es überhaupt soweit kommen muß, insbesondere dann nicht, wenn Sie den Fehler noch vor dem Drücken der RETURN-Taste bemerken. In diesem Fall können Sie die DEL-Taste

(JOYCE: DEL-Taste mit Linkspfeil) drücken, wodurch das letzte Zeichen gelöscht wird und der Cursor um eine Stelle nach links rückt. Da der Befehl jetzt richtig geschrieben ist, kann er nun auch durch Drücken der RETURN-Taste ausgeführt werden.

Betrachten wir ein weiteres Beispiel:

A>*fir*

Hier wurde nicht das letzte, sondern bereits das erste Zeichen falsch eingegeben. Um diesen Fehler zu korrigieren, haben wir zwei Möglichkeiten:

Zunächst können wir dreimal hintereinander die DEL-Taste drücken, wodurch die gesamte Zeile gelöscht wird. Da beim CPC 6128 und JOYCE sämtliche Tasten mit einer Autorepeat-Funktion ausgestattet sind, brauchen wir die DEL-Taste nur lange genug zu drücken, bis alle drei Zeichen verschwunden sind. Dabei werden Sie dann feststellen, daß das Anforderungszeichen nicht gelöscht werden kann.

Es gibt aber noch eine andere Funktion, die die gesamte Befehlszeile auf einmal löscht. Halten Sie dazu die CTRL-Taste gedrückt und drücken Sie anschließend die X-Taste. Einen solchen kombinierten Tastendruck beschreibt man mit

CTRL-X

oder kürzer

^X

In manchen Fällen erscheint die verkürzte Schreibweise auch auf dem Bildschirm, da die mit der CTRL-Taste erzeugten Steuerzeichen sonst nicht dargestellt werden können. Aus drucktechnischen Gründen wird der auf dem Bildschirm erscheinende "Pfeil nach oben" durch "^" ersetzt.

Da der JOYCE keine CTRL-Taste besitzt, benutzen Sie statt dessen die ALT-Taste, die die gleiche Funktion ausübt. Im folgenden werden wir aber nur noch von der CTRL-Taste sprechen, da diese Bezeichnung für CP/M üblich ist und die meisten CP/M-Computer diese Taste enthalten.

Nachdem Sie nun die Zeile gelöscht haben, geben Sie sie nochmals richtig ein und drücken die RETURN-Taste. Beachten Sie, daß Sie eine fehlerhafte Befehlszeile immer soweit löschen müssen, bis alle falschen Zeichen entfernt sind. Von dieser Stelle an geben Sie dann die richtigen Zeichen ein. Es ist also nicht in jedem Fall erforderlich, die gesamte Zeile zu löschen.

Außer CTRL-X gibt es noch eine Reihe anderer CTRL-Steuerzeichen, von denen wir die wichtigsten hier behandeln werden. Darüber hinaus befindet sich im Anhang eine Zusammenfassung sämtlicher Steuerzeichen.

Doch nun noch einmal zurück zur DEL-Taste. Statt DEL könnten wir auch CTRL-H eingeben, was die gleiche Wirkung hat. Ähnliches gilt auch für die RETURN-Taste (ENTER), die wir durch CTRL-M ersetzen können.

Falls Sie einmal eine sehr lange Befehlszeile eingeben müssen, die nicht in eine Bildschirmzeile paßt, können Sie sich folgendermaßen behelfen: Nach Abschluß einer jeden Bildschirmzeile drücken Sie CTRL-E (nicht RETURN!), worauf Sie in der nächsten Zeile weiterschreiben können. Erst wenn Sie Ihren Befehl komplett eingegeben haben, drücken Sie die RETURN-Taste.

Als kleines Demonstrationsbeispiel wollen wir einmal den DIR-Befehl auf drei Bildschirmzeilen verteilen, wobei jede Zeile nur ein Zeichen enthält:

A>*D <CTRL-E>*
I <CTRL-E>
R <RETURN>

Auch in diesem Fall wird das Directory genauso wie vorher aufgelistet. Übrigens wollen wir von nun an sämtliche CP/M-Befehle groß schreiben, um sie besser hervorzuheben. Dies ist jedoch, wie wir oben bereits gesehen haben, keinesfalls notwendig.

Wir werden in diesem Buch noch viele Befehle kennenlernen, die etwas auf dem Bildschirm auflisten, wie es auch beim DIR-Befehl der Fall ist. Häufig ist das Listing so umfangreich, daß es nicht auf den Bildschirm paßt und sich infolgedessen nach oben schiebt. Diesen Vorgang nennt man auch Scrollen. Hier können wir mit CTRL-S den Listvorgang unterbrechen und ihn mit CTRL-Q fortsetzen. Versuchen Sie dies einmal, während Sie das Directory listen.

Unter CP/M können Sie von der hervorragenden Eigenschaft Gebrauch machen, daß alles, was auf dem Bildschirm gelistet, auch gleichzeitig über den Drucker ausgegeben wird. Achten Sie aber unbedingt darauf, daß Ihr Drucker auch angeschlossen und eingeschaltet ist. CPC-Computer haben nämlich die unangenehme Eigenschaft, in einer Endlosschleife zu hängen, wenn der Drucker angesprochen, aber nicht angeschlossen ist.

Ist der Drucker betriebsbereit, können Sie ihn mit CTRL-P der Bildschirmausgabe parallel schalten. Probieren Sie das einmal aus und listen Sie dann ganz normal das Directory. Sie erhalten den Inhalt Ihrer Diskette

somit schwarz auf weiß ausgedruckt. Durch eine erneute Eingabe von CTRL-P schalten Sie anschließend den Drucker wieder ab.

Abschließend wollen wir auch in diesem Zusammenhang CTRL-C nicht vergessen, das ein gerade ablaufendes Programm unterbricht und einen Warmstart auslöst. Während bei CP/M 2.2 auch nach jedem Diskettenwechsel CTRL-C gedrückt werden mußte, kann dies bei CP/M Plus entfallen, da hier das BDOS diese Funktion automatisch ausführt.

2.2 Etwas über Dateien

Ebenso wie bei BASIC, gibt es auch bei CP/M Dateien. Allgemein ausgedrückt ist eine Datei eine Ansammlung von Daten, die auf Diskette oder einem anderen Datenträger abgelegt sind. Jede Datei enthält einen Namen, der in das Inhaltsverzeichnis (Directory) der Diskette eingetragen wird.

Nun gibt es allerdings eine Vielzahl von verschiedenen Dateien, welche die unterschiedlichsten Arten von Informationen enthalten können. So gibt es z.B. Programmdateien, die ein ausführbares Programm enthalten, Textdateien, die von einem Textverarbeitungssystem erzeugt werden oder reine Datendateien, in denen beispielsweise ein Adressenverzeichnis abgelegt ist. Selbst wenn Sie Ihre Farbgraphiken auf Diskette schreiben, legen Sie eine Datei an, die in diesem Fall den Inhalt des Bildschirm-RAMs speichert.

In diesem Abschnitt erhalten Sie nun einen ersten Einblick in die verschiedensten Dateiarten, die unter CP/M eine Rolle spielen. Nachdem wir im letzen Abschnitt schon einmal das Directory der Systemdiskette für den CPC 6128, Seite 1, aufgelistet haben, wollen wir dies jetzt nochmals wiederholen und uns die Einträge darin einmal genauer ansehen.

Von nun an verzichten wir darauf, das Anforderungszeichen A> mit anzugeben und beschränken uns lediglich auf die einzugebende Befehlszeile.

Hier noch einmal das Directory:

DIR

```
A: C10CPM3  EMS : BANKMAN BAS : PROFILE  ENG : SUBMIT   COM
A: SETKEYS  COM : KEYS    CCP : LANGUAGE COM : SET24X80 COM
A: PALETTE  COM : SETSIO  COM : SETLST   COM : DISCKIT3 COM
A: DATE     COM : DEVICE  COM : DIR      COM : ED       COM
A: ERASE    COM : GET     COM : PIP      COM : PUT      COM
A: RENAME   COM : SHOW    COM : TYPE     COM : SET      COM
A: SETDEF   COM : AMSDOS  COM : BANKMAN  BIN : KEYS      WP
```

Mit dem DIR-Befehl werden sämtliche Dateinamen in der Reihenfolge ausgegeben, in der sie im Directory eingetragen sind. Dabei erscheinen

jeweils vier Namen nebeneinander in einer Zeile. Am Zeilenanfang ist das Laufwerk aufgeführt, von dem das Directory gelistet wurde. In unserem Fall ist es Laufwerk A.

Falls Sie ein zweites Laufwerk B angeschlossen haben, können Sie auch das Directory von dessen Diskette listen, was mit dem Befehl

DIR B:

geschieht. Achten Sie darauf, daß zwischen den Bezeichnungen DIR und B: ein Leerzeichen frei bleibt, da sonst CP/M den Befehl falsch interpretiert. Diese Schreibweise gilt übrigens für die meisten CP/M-Befehle, falls sie sich auf ein anderes als das Bezugslaufwerk beziehen.

Normalerweise ist Laufwerk A immer das Bezugslaufwerk. Sie können allerdings auch Laufwerk B dazu bestimmen, indem Sie

B:

eingeben. Wenn Sie jetzt den DIR-Befehl erteilen, erhalten Sie das Directory von Laufwerk B. Das Inhaltsverzeichnis von "A" erscheint in diesem Fall entsprechend mit

DIR A:

Wollen Sie später wieder "A" zum Bezugslaufwerk bestimmen, geben Sie lediglich

A:

ein. Für den Normalfall wollen wir aber Laufwerk A als Bezugslaufwerk beibehalten und nur in Ausnahmefällen auf "B" zurückgreifen. Außerdem wird CP/M Plus immer von Laufwerk A aus gestartet (gebootet). Dazu muß die Diskette, jedenfalls beim CPC 6128, im Systemformat formatiert sein und zudem die Datei C10CPM3.EMS (JOYCE: J12DCPM3.EMS) enthalten, in der das gesamte CP/M Plus-Betriebssystem abgelegt ist. Mit diesen Dingen werden wir uns aber an anderer Stelle noch ausführlich befassen.

Betrachten wir nun wieder das aufgelistete Directory, stellen wir fest, daß jeder Dateiname noch einen Zusatz enthält, der als Dateityp oder Extension bezeichnet wird. Wie Sie sehen, ist der überwiegende Teil der Dateinamen mit dem Zusatz (Extension) "COM" versehen.

COM steht für englisch "Command" und kennzeichnet unter CP/M eine Befehlsdatei. Solche Dateien haben die Eigenschaft, daß sie nach dem Aufruf

sofort ausgeführt werden. Dazu müssen Sie nur deren Namen (ohne COM) als Befehlszeile eingeben und anschließend die RETURN-Taste drücken.

Auf einer Diskette können verschiedene Dateiarten gespeichert sein, die durch die jeweilige Extension gekennzeichnet sind. So finden Sie auf Ihren Systemdisketten auch einige Dateien mit der Extension BAS vor, die ausschließlich von BASIC aus ausgeführt werden können. Deshalb können Sie auch Ihre eigenen BASIC-Programme auf Disketten ablegen, die gleichzeitig noch CP/M-Programme enthalten. Das Disketten-Aufzeichnungsformat ist nämlich für sämtliche Dateiarten identisch und auch für das Disketten-Betriebssystem (DOS) ist es ohne Bedeutung, mit welcher Extension die Dateinamen versehen sind.

Unter gewissen Umständen können und müssen BASIC-Programme sogar mit CP/M-Dienstprogrammen bearbeitet werden. Dies gilt insbesondere für das CP/M-Programm DISCKIT3.COM (JOYCE DISCKIT.COM) zum Kopieren und Formatieren von Disketten, aber auch für PIP.COM zum Kopieren einzelner Dateien. In der Regel spielt es für diese Programme keine Rolle, ob sie BASIC-Dateien, die unter AMSDOS arbeiten, oder reine CP/M-Dateien oder sonstige Dateien kopieren.

Dateinamen dürfen maximal acht Zeichen umfassen, an die sich noch die Extension von drei Zeichen anschließt. Im Gegensatz zur Auflistung des Directories mit dem DIR-Befehl, muß meistens zwischen Dateinamen und Extension ein Punkt gesetzt werden, besonders wenn die Extension mit in der Befehlszeile auftaucht. Abgesehen von der Ausführung der COM-Dateien ist diese Schreibweise sogar meist zwingend. Betrachten wir ein Beispiel:

Auf der Systemdiskette befindet sich das Programm DISCKIT3 mit der Extension COM, also eine reine CP/M-Befehlsdatei zum Kopieren und Formatieren von Disketten. Im Directory-Listing ist sie aufgeführt mit

```
DISCKIT3 COM
```

wobei DISCKIT3 die maximale Länge von acht Zeichen umfaßt. Zwischen DISCKIT3 und der Extension befindet sich ein Leerzeichen. Wäre der Dateiname weniger als acht Zeichen lang, würde sich der Abstand zur Extension entsprechend vergrößern. So umfaßt der Name PIP beispielsweise nur drei Zeichen, weshalb er wie folgt mit dem DIR-Befehl ausgegeben wird:

```
PIP      COM
DISCKIT3 COM
```

Zum Vergleich haben wir nochmals den Namen DISCKIT3 darunter gesetzt.Da es sich in beiden Fällen um COM-Dateien handelt, werden sie mit

PIP <RETURN>

bzw.

DISCKIT3 <RETURN>

aufgerufen und ausgeführt. Von diesem Sonderfall einmal abgesehen, werden beide Namen zusammen mit ihrer Extension als PIP.COM bzw. DISCKIT3.COM geschrieben, wobei zwischen Dateinamen und Extension lediglich ein Punkt steht. Diese Schreibweise wollen wir auch im vorliegenden Buch beibehalten.

Nachfolgend einige Beispiele für generell zulässige Dateinamen:

```
TEST.COM
BUCH.TXT
WS.COM
BIBLTHEK.DAT
LAGER.BAK
PROBE.BAS
ERW.BIN
TITEL.
```

Nicht alle hier verwendeten Extensionen haben eine bestimmte Bedeutung und können teilweise auch frei gewählt bzw. weggelassen werden. Dies ist besonders bei reinen Daten- oder Textdateien häufig der Fall.

In der Praxis sind jedoch einige Extensionen für bestimmte Zwecke reserviert, die Sie bei der Wahl der Dateinamen berücksichtigen sollten:

COM	Befehlsdatei
BAK	Backup-(Sicherungs-)Datei
SUB	Stapelverarbeitungs-Datei
ASM	Datei mit Assembler-Quelltext
BAS	BASIC-Programm
BIN	Binärdatei unter BASIC bzw. AMSDOS
PRN	Datei für Listenausdruck assemblierter Maschinenprogramme
HEX	Datei im INTEL-HEX-Format
$$$	Zwischendatei (nur für bestimmte Anwendungen)

Viele käuflichen BASIC- und CP/M-Programme verwenden außerdem weitere Extensionen, die gegebenenfalls zu beachten sind.

Der DIR-Befehl eignet sich aber nicht nur zum Auflisten des gesamten Directories, sondern kann auch zur Abfrage einzelner oder mehrdeutiger Dateinamen verwendet werden.

Angenommen, Sie möchten feststellen, ob auf Ihrer Diskette die Datei DISCKIT3.COM vorhanden ist, ohne erst das gesamte Directory durchsuchen zu müssen, dann geben Sie lediglich ein:

DIR DISCKIT3.COM

Dies ist ein Anwendungsfall, in dem der Dateiname mit der durch einen Punkt abgetrennten Extension anzugeben ist. Ist nun DISCKIT3.COM auf der Diskette enthalten, erscheint:

```
A: DISCKIT3 COM
```

d.h. die Schreibweise ist genauso wie beim einfachen DIR-Befehl. Wurde die gewünschte Datei dagegen nicht gefunden, erscheint die Meldung:

```
NO FILE
```

In diesem Beispiel handelt es sich um einen eindeutigen Dateinamen, der vorgegeben wurde. Es besteht aber auch die Möglichkeit, mehrdeutige Dateinamen anzugeben, wozu die Zeichen "*" und "?" dienen. Hier ein paar weitere Beispiele:

Wie wir bereits wissen, können sich auf ein und derselben Diskette gleichzeitig CP/M- und BASIC-Programme befinden. Mit Hilfe von

*DIR *.BAS*

können wir nun feststellen, welche BASIC-Dateien darunter sind und erhalten z.B.

```
A: SPIELE   BAS : MONSTER  BAS
```

Selbstverständlich funktioniert dies genauso mit allen anderen Extensionen, so daß wir beispielsweise auch sämtliche COM-Dateien auflisten könnten.

Nun möchten wir wissen, welche COM-Dateien sich auf der Diskette befinden, die mit dem Buchstaben D beginnen. Dazu geben wir

DIR D???????.COM

oder

DIR D.COM*

ein und erhalten die Auflistung

```
A: DATE     COM : DUMP     COM : DISCKIT3 COM
```

Wir sehen also, daß ein Fragezeichen einen einzelnen Buchstaben ersetzt. Ein Sternchen dagegen steht entweder für den gesamten Dateinamen bzw. die gesamte Extension oder ersetzt die restlichen Zeichen des Dateinamens, wenn die ersten fest vorgegeben sind. Um dies zu verdeutlichen, wollen wir nochmals das gesamte Directory ausgeben, diesmal allerdings mit Hilfe mehrdeutiger Dateinamen. Sämtliche nachfolgend aufgeführten Befehle haben alle dieselbe Wirkung:

DIR

*DIR *.**

*DIR ????????.**

*DIR *.???*

DIR ????????.???

Natürlich wird man in der Praxis hierfür nur DIR verwenden. Es gibt jedoch viele Befehle, die auf jeden Fall einen ein- oder mehrdeutigen Dateinamen erfordern.

Wenn Sie den DIR-Befehl in seiner bisher beschiebenen Form verwenden, können Sie lediglich die Namen von Dateien oder Dateigruppen auf der Diskette erfahren. Dies reicht zwar in vielen Fällen aus, nicht aber, wenn Sie nähere Angaben über die noch freie Diskettenkapazität oder den Umfang von einzelnen Dateien benötigen.

CP/M Plus enthält noch einen erweiterten DIR-Befehl, der einen genaueren Aufschluß über die Beschaffenheit von Dateien gibt. Er funktioniert aber nur, wenn die Datei DIR.COM ausgeführt wird, die sich mit auf der Sytemdiskette befindet. Durch Eingabe von

DIR [FULL]

erhalten wir nämlich folgende Darstellung:

```
Scanning Directory...

Sorting Directory...

Directory For Drive A: User 0

     Name      Bytes   Recs   Attributes      Name      Bytes   Recs  Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
AMSDOS  COM      1k      8 Dir RW       BANKMAN  BAS      1k      7 Dir RW
BANKMAN BIN      2k     12 Dir RW       C10CPM3  EMS     25k    200 Dir RW
DATE    COM      3k     23 Dir RW       DEVICE   COM      8k     58 Dir RW
DIR     COM     15k    114 Dir RW       DISCKIT3 COM      6k     48 Dir RW
ED      COM     10k     73 Dir RW       ERASE    COM      4k     29 Dir RW
GET     COM      7k     51 Dir RW       KEYS     CCP      1k      3 Dir RW
KEYS    WP       1k      3 Dir RW       LANGUAGE COM      1k      8 Dir RW
PALETTE COM      1k      8 Dir RW       PIP      COM      9k     68 Dir RW
PROFILE ENG      1k      1 Dir RW       PUT      COM      7k     55 Dir RW
RENAME  COM      3k     23 Dir RW       SET      COM     11k     81 Dir RW
SET24X80 COM     1k      8 Dir RW       SETDEF   COM      4k     32 Dir RW
SETKEYS COM      2k     16 Dir RW       SETLST   COM      2k     16 Dir RW
SETSIO  COM      2k     16 Dir RW       SHOW     COM      9k     66 Dir RW
SUBMIT  COM      6k     42 Dir RW       TYPE     COM      3k     24 Dir RW

Total Bytes     =    146k  Total Records =    1093  Files Found =   28
Total 1k Blocks =    146   Used/Max Dir Entries For Drive A:   29/  64
```

Nun erhalten wir das Directory in alphabetischer Reihenfolge sortiert, wobei gleichzeitig auch der Speicherplatz angegebenen ist, den jede Datei auf der Diskette belegt. Dieser erscheint sowohl in Kilobyte (K) als auch in Records. Ein Record ist eine CP/M-Aufzeichnungseinheit und umfaßt 128 Byte, während ein Kilobyte 1024 Byte enthält. Doch hierüber später mehr.

Die beiden unteren Zeilen enthalten weitere wichtige Informationen. In unserem Beispiel sind 146K (bzw. 146 Blöcke mit 1 KByte) auf der Diskette belegt und sämtliche Dateien umfassen insgesamt 1093 Records. Das Directory verwaltet derzeit 28 Dateien (Files), wobei 29 der 64 möglichen Eintragungen belegt sind. Daß hier 28 Dateien 29 Eintragungen benötigen, hat seinen bestimmten Grund, den wir im Zusammenhang mit den Extents (Kapitel 2.4) noch kennenlernen werden. Mit allen weiteren in dieser Auflistung enthaltenen Informationen werden wir uns in Kapitel 3 noch intensiv beschäftigen.

Leider gibt diese Directory-Auflistung keinen Aufschluß darüber, wieviel Speicherplatz wir noch für weitere Dateien auf der Diskette zur Verfügung haben. Dazu benötigen wir nämlich das CP/M-Programm SHOW, das sich ebenfalls mit auf der Systemdiskette befindet. Indem wir nun

SHOW

eingeben, erhalten wir den Speicherplatz, der auf den Disketten aller angeschlossenen Laufwerke noch freie ist, wie z.B.

```
A: RW, Space: 23k
B: RW, Space: 161k
```

Uns stehen demgemäß noch 23 KByte in Laufwerk A und 161 KByte in Laufwerk B für weitere Dateien zur Verfügung. RW gibt an, daß es sich hierbei um Schreib-/Lesespeicher handelt, der mit keinem besonderen Schreibschutz versehen ist. Mit

SHOW A:

bzw.

SHOW B:

bekommen wir diese Angaben auch für jedes Laufwerk einzeln.

Da es sich bei dem erweiterten DIR-Befehl und dem SHOW-Befehl um auszuführende CP/M-Befehlsdateien bzw. nicht residente Befehle handelt (siehe unten), müssen sie jeweils auf der Diskette vorhanden sein. Wenn Sie mit zwei Laufwerken arbeiten, legen Sie am besten die Systemdiskette, die diese Dateien enthält, in Laufwerk A und die zu untersuchende Diskette in Laufwerk B und geben dann

DIR B:[FULL]

bzw:

SHOW B:

ein. Aber selbst wenn Sie nur ein Laufwerk zur Verfügung haben, können Sie genauso verfahren, denn CP/M unterstützt ein zweites Laufwerk B, selbst wenn nur ein Laufwerk angeschlossen ist. Das Laufwerk erhält dann einmal die Bezeichnung "A" und einmal die Bezeichnung "B", wobei Sie jeweils bei der Umschaltung zum Diskettenwechsel aufgefordert werden. Hierbei handelt es sich um einen wesentlichen Vorteil von CP/M Plus gegenüber CP/M 2.2, wo einige Programme, die zwei Laufwerke benötigten, einfach nicht einsatzfähig waren und man sich in einem solchen Fall anderweitig behelfen mußte. Dafür sprechen z.B. die vielen Kopier- und Verifizier-Programme, die sich auf der CP/M 2.2-Diskette des CPC 464 und 664 befinden, während bei zwei Laufwerken das universelle Kopierprogramm PIP (siehe unten und Kap. 5) hierfür vollauf genügt. Ein häufiger Diskettenwechsel unter CP/M Plus ist zwar manchmal lästig, bietet aber immer noch einschneidende Vorteile gegenüber den Möglichkeiten, die unter CP/M 2.2 zur Verfügung stehen.

Hier noch ein Hinweis an alle JOYCE-Benutzer: Den Befehl

DIR [FULL]

und andere, die eckige Klammern verwenden, können Sie in dieser Form nicht eingeben, denn der JOYCE ist bereits werksmäßig mit einer deutschen DIN-Tastatur und einem deutschen Zeichensatz ausgestattet, der diese Zeichen nicht enthält. Die ASCII-Code-Zeichen, die nach der amerikanischen Norm den eckigen Klammern entsprechen, sind beim JOYCE dem Buchstaben Ä anstelle von [und dem Buchstaben Ü anstelle von] (Ä und Ü sind Großbuchstaben) zugeordnet. Wenn Sie nun das erweiterte Directory auflisten wollen, müssen Sie demgemäß

DIR ÄFULLÜ

eingeben. Entsprechendes gilt auch für alle anderen Befehle mit eckigen Klammern. Dies betrifft auch den CPC 6128, wenn Sie ihn mit LANGUAGE 2 auf den deutschen Zeichensatz umgeschaltet haben. Diese Schreibweise wirkt zwar nicht gerade übersichtlich, sie ist aber leider nicht zu umgehen. Im weiteren Verlauf dieses Buches verwenden wir aber nur eckige Klammern, die der üblichen CP/M-Schreibweise entsprechen.

2.3 Das Dienstprogramm DISCKIT3

Dieses Dienstprogramm benötigen wir zum Formatieren, Kopieren (Backup) und Verifizieren von Disketten. Neben PIP gehört es zu den wichtigsten CP/M-Programmen, die Sie selbst dann benötigen, wenn Sie mit BASIC bzw. AMSDOS arbeiten.

Die nachfolgende Beschreibung gilt in erster Linie für den CPC 6128. Für den JOYCE befindet sich auf der Systemdiskette das Programm DISCKIT, das eine ähnliche Funktion hat. Wir werden darauf später noch gesondert eingehen.

Geben Sie jetzt einmal

DISCKIT3 <RETURN>

ein, worauf DISCKIT3 von der Diskette geladen und gestartet wird.

DISCKIT3 fragt zunächst ab, wieviele Laufwerke angeschlossen sind. Haben Sie nur das eingebaute Laufwerk A in Betrieb, erscheint

```
One drive found
```

Ist dagegen ein zusätzliches Laufwerk B angeschlossen, lautet die Meldung:

```
Two drives found
```

Im unteren Bildschirm erscheint nun folgendes Auswahlmenü:

```
Copy                7          (Diskette kopieren)
Format              4          (Diskette formatieren)
Verify              1          (Diskette verifizieren)
Exit from program   0          (DISCKIT3 verlassen)
```

Zur Auswahl des gewünschten Menüpunktes müssen Sie die entsprechende Funktionstaste im Zifferntastenblock drücken.

Bevor wir überhaupt etwas auf eine Diskette schreiben können, müssen wir sie formatieren. Dabei werden auf die Diskette Spuren und Sektoren aufgetragen, in denen später Daten oder Programme abgelegt werden können.

Die Disketten erhalten 40 Spuren (Spur 0 bis 39), von denen jede in 9 Sektoren zu je 512 Byte unterteilt ist. Die gesamte für den Anwender verfügbare Diskettenkapazität beträgt demnach

40 * 9 * 512 = 184320 Byte bzw. 180 KByte

Nachdem Sie DISCKIT3 geladen haben, drücken Sie die Taste f4 zum formatieren. Nehmen Sie jetzt die Systemdiskette aus dem Laufwerk und legen Sie statt dessen die zu formatierende Diskette ein. Achten Sie unbedingt darauf, daß Sie eine fabrikneue Diskette einlegen oder eine solche, auf deren Inhalt Sie verzichten können. Durch die Formatierung gehen nämlich auf der Diskette vorhandene Daten unwiderruflich verloren!

Sie sehen nun ein weiteres Menü auf dem Bildschirm, in dem Sie zur Art der Formatierung gefragt werden:

```
System format       9          (Systemformat)
Data format         6          (Datenformat)
Vendor format       3          (Vendorformat)
Exit menu           .          (Menü verlassen)
```

Falls Sie auf der zu formatierenden Diskette nur unter AMSDOS arbeiten möchten, wählen Sie jetzt das Datenformat (Taste f6) aus. Soll Ihre Diskette CP/M-Dateien erhalten oder sowohl CP/M- als auch AMSDOS-Dateien, müssen Sie Taste f9 (Systemformat) drücken. Dabei werden die beiden äußersten Spuren der Diskette (0 und 1) mit dem CP/M-Betriebssystem belegt und stehen deshalb für den Anwender nicht zur Verfügung. Entsprechend verringert sich auch die verfügbare Diskettenkapazität geringfügig auf 171 KByte.

Das Vendorformat ist mit dem Systemformat identisch, mit der Ausnahme, daß die Systemspuren für CP/M zwar reserviert, aber nicht beschrieben

werden. Dieses Format ist nur für den Verkauf von CP/M-Software interessant, wobei zwar die Software, nicht aber das CP/M-Betriebssystem mit verkauft wird.

Nachdem Sie die gewünschte Formatierungsart ausgewählt haben, erscheint folgende Meldung auf dem Bildschirm:

```
Y       Format as Data
           Any other key to exit menu
```

Drücken Sie jetzt Y, falls Sie eine Datendiskette formatieren möchten. Dies ist eine letzte Sicherheitsabfrage vor dem Formatieren. Falls Sie ein anderes Format gewählt haben, erscheint eine betreffende ähnliche Meldung. Wenn Sie statt Y eine andere Taste drücken, findet keine Formatierung statt,und der Computer kehrt ins Menü zurück.

Falls zwei Laufwerke angeschlossen sind, erfolgt vor dem Formatieren noch zusätzlich die Abfrage, ob die zu formatierende Diskette in Laufwerk A oder B liegt.

Kommen wir jetzt zum Kopieren einer Diskette (Taste f7). Dabei muß die Diskette, auf die kopiert werden soll (Zieldiskette), schon formatiert sein.

Sie brauchen sich beim Kopieren nicht darum zu kümmern, ob die zu kopierende Diskette (Quelldiskette) im System-, Daten- oder Vendorformat formatiert ist, da dies der Computer automatisch erkennt.

Falls Sie nur mit einem Laufwerk kopieren, erfolgt zunächst die Sicherheitsabfrage:

```
Y       Copy
           Any other Key to exit menu
```

Wenn Sie diese mit Y (Ja) beantworten, kann der Kopiervorgang beginnen. Sie sollten zuvor jedoch Ihre Systemdiskette aus dem Laufwerk nehmen und statt dessen die Diskette einlegen, die kopiert werden soll. Nun erscheinen abwechselnd die Meldungen:

```
Insert disc to WRITE              (Zieldiskette einlegen)
Press any key to continue         (Weiter - beliebige Taste drücken)
```

und

```
Insert disc to READ               (Quelldiskette einlegen)
Press any Key to continue         (Weiter - beliebige Taste drücken)
```

Wenn Sie die Anweisungen befolgen und den Diskettenwechsel immer richtig vornehmen, erscheint nach einigen Durchgängen die Meldung:

```
Copy completed                  (Kopiervorgang beendet)
Remove disc                     (Diskette herausnehmen)
Press any key to continue       (Weiter - beliebige Taste drücken)
```

Damit ist der Kopiervorgang beendet.

Falls sie mit zwei Laufwerken kopieren, müssen Sie Quell- und Zieldiskette nicht ständig wechseln. Zunächst erscheint das Zusatzmenü:

```
Write to A:     9       (auf Laufwerk A kopieren)
Write to B:     6       (auf Laufwerk B kopieren)
Exit menu       3       (Menü verlassen)
```

Sie legen jetzt Quell- und Zieldiskette in das richtige Laufwerk ein und drücken die entsprechende Taste. Es ist allerdings bequemer und schneller, von Laufwerk A auf Laufwerk B zu kopieren.

Der letzte Menüpunkt von DISCKIT3 ist Verify (Taste f1). Hierbei wird der Zustand der zu überprüfenden Diskette untersucht und alle Fehler werden angezeigt.

Verify arbeitet ebenfalls wahlweise mit einem oder zwei Laufwerken. Beachten Sie alle Anweisungen, die auf dem Bildschirm erscheinen, um den Test durchzuführen.

Kommen wir nun zum JOYCE, auf dessen Systemdisketten sich ein ähnliches Programm befindet. Es heißt hier nicht DISCKIT3, sondern lediglich DISCKIT und wird mit

DISCKIT <RETURN>

geladen und gestartet. Da sich dieses Programm in einigen Punkten von der CPC 6128-Version DISCKIT3 unterscheidet, wollen wir es hier kurz betrachten.

Wenn Sie DISCKIT laden, fällt Ihnen auf, daß hier sämtliche Anweisungen in deutscher Sprache erscheinen, sodaß Sie bei der Anwendung keine englischen Sprachkenntnisse benötigen. Zur Bedienung selbst ist nicht mehr viel hinzuzufügen, da sie derjenigen von DISCKIT3 sehr ähnlich ist. Wir lernen hier lediglich die EXIT-Taste kennen, die sich unter der rechten SHIFT-Taste befindet und zum Verlassen des Programms dient.

Wenn Sie mit DISCKIT Disketten formatieren, besteht jedoch ein wesentlicher Unterschied, denn DISCKIT kann nur in einem Format formatieren: Es unterscheidet sich von den drei Formaten des CPC 6128 und kann nur vom JOYCE gelesen werden. Dieses Format verwendet eine Systemspur und ist dem Systemformat des CPC 6128 sehr ähnlich.

Auf dem JOYCE formatierte Disketten können also nicht auf dem CPC 6128 gelesen werden, dagegen kann der JOYCE Disketten im CPC-Format lesen und beschreiben, was für den Datenaustausch zwischen beiden Computern sicherlich sehr vorteilhaft ist. Übrigens verwenden auch der CPC 464 und 664 das gleiche Format wie der CPC 6128.

2.4 Etwas über den Diskettenaufbau

Nachfolgend wollen wir uns einmal etwas näher mit dem internen Diskettenaufbau beschäftigen. Wenn Sie eine fabrikneue Diskette kaufen, enthält diese lediglich eine Magnetschicht, auf die noch nichts aufgetragen ist. Wollen Sie nun Daten auf die Diskette schreiben, so muß sie zuerst formatiert werden Dabei wird sie in eine bestimmte Anzahl von Spuren (Tracks) eingeteilt, die wiederum in einzelne Sektoren unterteilt werden.

Stellen Sie sich einmal eine runde Torte vor, auf die eine Reihe von Ringen aus Sahne oder Zuckerguß gespritzt ist. Schneiden wir nun die Torte in einzelne, gleichmäßige Stücke, so enthält jedes Stück immer noch einen Teil von jedem Ring.

Diesen bildlichen Vergleich können wir auch auf die Diskette übertragen. Bei der Formatierung erhält sie natürlich statt der Zuckerringe Spuren und statt der Teilringe auf den Tortenstücken Sektoren.

Beim CPC 6128 und JOYCE sind auf einer formatierten Diskette 40 Spuren mit je 9 Sektoren aufgetragen, von denen jeder 512 Byte an Informationen aufnehmen kann. Insgesamt ergibt dies eine Speicherkapazität von 180 KByte, wovon allerdings ein kleiner Teil für die Systemspuren und das Directory abgeht, die für den Anwender nicht nutzbar sind.

Nun verwenden bedauerlicherweise die meisten Computerhersteller ihr eigenes Diskettenformat mit einer unterschiedlichen Anzahl von Spuren, Sektoren und Byte pro Sektor. Da CP/M aber auf sämtlichen Systemen lauffähig sein soll, ist es die Aufgabe des BIOS, die jeweilige Anpassung vorzunehmen.

Das CP/M-Betriebssystem arbeitet allerdings nicht mit den physikalischen Sektoren, sondern überläßt dies dem jeweiligen BIOS. Es kann nur in sogenannten logischen Records und Blöcken "denken". Ein Record ist eine Aufzeichnungseinheit von 128 Byte, wogegen ein Block meist 1024 Byte (1 KByte) umfaßt, wie es auch beim CPC 6128 und JOYCE der Fall ist.

Wenn Sie eine Datei auf Diskette schreiben, belegt sie, je nach Umfang, eine bestimmte Anzahl von Records, wie wir es in der Directory-Auf-

listung im Abschnitt 2.2 gesehen haben. Nun kann das Directory Dateien jedoch nur in Blöcken zu je 1024 Bytes verwalten, weshalb dort auch die Anzahl der Blöcke mit angegeben ist.

Zwischen Records und Blöcken besteht demgemäß ein logischer Zusammenhang, denn ein Block enthält exakt acht Records. Für jede angefangenen acht Records muß also ein weiterer Block auf der Diskette reserviert werden, der im Directory eingetragen wird. Wenn Sie in der Auflistung die jeweilige Recordanzahl durch acht dividieren, können Sie dies leicht nachprüfen.

Selbst wenn eine Datei nur aus drei Bytes besteht, belegt sie einen Record, für den ein ganzer Block reserviert werden muß.

Das Directory kann maximal 64 Einträge enthalten, von denen jeder 16 KByte, also 16 Blöcke verwalten kann. Diese 16 Blöcke bilden somit eine neue Einheit, die wir als Extent bezeichnen. Umfaßt nun eine Datei mehr als 16 KByte, so benötigt sie für jede weiteren angefangenen 16 KByte einen neuen Extent oder Erweiterungseintrag im Directory.

Als wir in Abschnitt 2.2 das erweiterte Directory der Systemdiskette auflisteten, wurden 29 Directory-Einträge angezeigt, obwohl die Diskette nur 28 Dateien enthält. Bei genauerer Betrachtung stellen wir aber fest, daß eine Datei, nämlich C10CPM3.EMS, 25 KByte oder 2 Extents umfaßt. Sie benötigt deshalb einen zweiten Directory-Eintrag, der jedoch nicht gesondert in der Auflistung erscheint.

2.5 Dateien kopieren

Mit DISCKIT3 bzw. DISCKIT konnten wir immer nur die gesamte Diskette kopieren. Dieser Vorgang ist sicherlich zur Herstellung von Sicherheitskopien sehr nützlich. So sollten Sie sich von Ihren Systemdisketten unbedingt eine Kopie herstellen, mit der Sie arbeiten. Das Orginal bewahren Sie an sicherer Stelle auf. Obwohl die Schneider-Disketten und -Laufwerke eine hohe Datensicherheit aufweisen, ist es dennoch nicht völlig auszuschließen, daß die Diskette beschädigt oder sogar zerstört wird. In einem solchen Fall sind Sie dann froh, wenn Sie noch eine unzerstörte Kopie besitzen.

Aber auch von allen anderen wichtigen Dateien sollten Sie mindestens eine Sicherheitskopie herstellen, möglichst auf einer anderen Diskette. Dabei ist es oft von Vorteil, nur eine Datei und nicht die gesamte Diskette zu kopieren. Dies geschieht am besten mit dem CP/M-Dienstprogramm PIP, das wir hier kurz betrachten und in Kapitel 5 noch ausführlich behandeln wer-

den. Dabei kommt uns wieder die Eigenschaft von CP/M Plus zugute, ein Laufwerk abwechselnd mit "A" und "B" zu adressieren, falls kein zweites Laufwerk vorhanden ist.

Zunächst laden und starten Sie PIP von Ihrer Systemdiskette, indem Sie

PIP <RETURN>

eingeben. Daraufhin erscheint:

```
CP/M 3 PIP VERSION 3.0
*
```

Der Cursor steht jetzt hinter dem Sternchen, so daß Sie einzelne Kopieranweisungen erteilen können. Betrachten wir dies an einem Beispiel, in dem wir eine Datei BUCH.TXT von Laufwerk A nach Laufwerk B kopieren:

B:=A:BUCH.TXT <RETURN>

Der Dateiname ist hier vollständig anzugeben, also einschließlich der Extension. Bevor Sie mit dem Kopieren beginnen, vergessen Sie aber nicht, die Quelldiskette, von der die Datei BUCH.TXT kopiert werden soll, in Laufwerk A und die Zieldiskette, auf die kopiert werden soll, in Laufwerk B einzulegen. Falls Sie nur mit einem Laufwerk arbeiten, legen Sie zunächst die Quelldiskette ein und führen den Befehl aus. Achten Sie aber unbedingt darauf, daß in der unteren rechten Bildschirmecke der Hinweis

```
Drive is A:
```

bzw. beim JOYCE

```
Laufwerk ist A:
```

erscheint, damit Sie auch sichergehen, daß die Quelldiskette in Laufwerk A liegt. Sollte als Bereitschaftszeichen zwar

```
A>
```

erscheinen, obwohl rechts unten Laufwerk B angezeigt ist, ist dies nicht weiter tragisch, denn Sie werden vor Ausführung der Operation noch rechtzeitig darauf hingewiesen, die Diskette zu wechseln und eine beliebige Taste zu drücken. Lautet das Bereitschaftszeichen aber

```
B>
```

dann ist "B" das Bezugslaufwerk und Sie müssen erst mit

A:

Laufwerk A dazu bestimmen. Auch in diesem Fall gilt das eben Gesagte.

Zunächst liest PIP die zu kopierende Datei in den Arbeitsspeicher und fordert dann zum Diskettenwechsel auf. Nun ersetzen Sie die Quell- durch die Zieldiskette und drücken eine beliebige Taste, worauf die Datei auf die Zieldiskette geschrieben wird.

Nach Beendigung der Operation erscheint wieder das Sternchen. Sie können nun entweder eine neue Kopieranweisung erteilen oder PIP verlassen, indem Sie lediglich die RETURN-Taste drücken.

2.6 Residente und nichtresidente Befehle

Unter residenten Befehlen versteht man all diejenigen Befehle, die fest ins CP/M-Betriebssystem eingebaut sind und nicht erst von der Diskette nachgeladen werden müssen.

Während man unter CP/M 2.2 noch klar zwischen residenten und nichtresidenten Befehlen unterscheiden konnte, ist dies bei CP/M Plus nur bedingt möglich. Betrachten wir einmal den DIR-Befehl. Wenden wir ihn ohne Option an, oder nur mit einem ein- oder mehrdeutigen Dateinamen, ist er ein residenter Befehl, der im Betriebssystem enthalten und nicht erst von Diskette geladen werden muß.

Benutzen wir dagegen DIR [FULL], um das erweiterte Directory aufzulisten, so liegt ein nichtresidenter Befehl vor, da hierbei die Datei DIR.COM auf der Systemdiskette ausgeführt wird. Ähnliches gilt auch für die noch zu besprechenden Befehle REN [RENAME] und ERA [ERASE], je nachdem, ob Optionen verwendet werden oder nicht. SHOW dagegen ist in jedem Fall ein nichtresidenter Befehl, der von Diskette zu laden ist.

Hier eine Aufstellung der residenten und der wichtigsten nichtresidenten Befehle, die in jedem CP/M Plus-System vorhanden sind - ohne Rücksicht darauf, ob sie bereits besprochen wurden oder nicht. Die Erläuterung "teilweise" besagt, daß der betreffende Befehl je nach Optionen sowohl als residenter wie auch als nichtresidenter Befehl eingesetzt werden kann. Für einige residente Befehle ist sowohl die ausführliche wie auch die Kurzform zulässig, die beide nebeneinander angegeben sind.

Residente Befehle:

DIR		Listet Directory	(teilweise)
DIRSYS	DIRS	Listet Systemdateien in Directory	
ERASE	ERA	Löscht Dateien	(teilweise)
RENAME	REN	Nennt Dateien um	(teilweise)
TYPE	TYP	Listet ASCII-Datei	(teilweise)
USER	USE	Weist Benutzerbereich zu	

Nichtresidente Befehle:

COPYSYS	Erstellt Systemdiskette (nicht CPC und JOYCE)
DATE	Datum/Uhrzeit einstellen/ändern
DEVICE	Zuordnung logischer zu physikal.Geräteeinheiten
DUMP	Listet Maschinenprogramm
ED	Ruft den Texteditor auf
GET	Terminaleingabe über Diskettendatei
HELP	Dokumentation sämtlicher CP/M-Plus-Befehle
HEXCOM	Umwandlung von Intel-HEX- in ausführbare COM-Datei
INITDIR	Präpariert Directory für Datums- und Zeiteinträge
LINK	Bildet COM-Datei aus verschiebbarem REL-Modul
MAC	Makro-Assembler
PIP	Universelles Kommunikations- und Kopierprogramm
PUT	Schreibt Terminalausgabe in Diskettendatei
RMAC	Bildet verschiebbare Moduln aus Assembler-Quelltext
SET	Setzt Dateiattribute, Zeitmarken u.a.
SETDEF	Setzt Systemparameter und Suchpfad
SHOW	Anzeige diverser Diskettenparameter
SID	Debugger
SUBMIT	Dient zur Stapelverarbeitung
XREF	Erstellt Referenztabelle für Assemblerprogramm

2.7 Dateien umbenennen

Dateien können mit dem REN-Befehl einen neuen Namen erhalten. Dabei wird der alte Name lediglich durch den neuen im Directory ersetzt. Hier die allgemeine Schreibweise:

REN neuer Dateiname=alter Dateiname

Sowohl der neue als auch der alte Dateiname müssen jeweils einschließlich Extension angegeben werden. Hier ein Beispiel:

REN NEUDAT.TXT=ALTDAT.TXT

Hierbei wird die Datei ALTDAT.TXT in NEUDAT.TXT umbenannt. Soll die Umbenennung auf Laufwerk B stattfinden, lautet der Befehl entsprechend:

REN B:NEUDAT.TXT=ALTDAT.TXT

Achten Sie darauf, daß kein Leerzeichen in der Befehlszeile auftreten darf, außer direkt hinter REN.

Das folgende Beispiel benutzt einen mehrdeutigen Dateinamen und benötigt deshalb die Datei RENAME.COM auf Diskette (nichtresidenter Befehl). Wir wollen jetzt sämtlichen BAK-Dateien die Extension TXT verleihen, damit sie wieder der Textverarbeitung zur Verfügung stehen:

*REN *.TXT=*.BAK*

Achten Sie aber darauf, daß sich noch keine andere Datei unter dem neuen Namen auf der Diskette befindet. Ist dies nämlich der Fall, erscheint die Meldung:

```
FILE EXISTS
```

und die Umbenennung findet nicht statt.

Eine etwas komfortablere Eingabe der Dateinamen ist durch Ausführung der Datei RENAME.COM möglich, die Sie mit

RENAME

von Diskette laden und starten. Das Programm fragt dann nach dem neuen und dem alten Dateinamen.

2.8 Dateien löschen

Zum Löschen von Dateien dient der ERA-Befehl. Hinter ERA muß ein ein- oder mehrdeutiger Dateiname aufgeführt sein. Hier einige Beispiele:

ERA PIP.COM

löscht die Datei PIP.COM,

ERA B:PROBE.TXT

löscht die Datei PROBE.TXT auf Laufwerk B,

*ERA *.COM*

löscht sämtliche COM-Dateien,

ERA AB.**

löscht sämtliche Dateien, deren Namen mit AB beginnen, und

*ERA *.**

löscht sämtliche Dateien auf der Diskette. Da der ERA-Befehl bei falscher Anwendung fatale Auswirkungen haben kann, findet bei Angabe von mehrdeutigen Dateinamen vorsichtshalber noch eine Sicherheitsabfrage statt.

Sie können Dateien auch mit dem nichtresidenten Befehl löschen, indem Sie mit

ERASE

die Datei ERASE.COM von Diskette laden und ausführen. Dieses Programm fragt dann gesondert nach dem Namen der zu löschenden Datei.

2.9 Mehre Befehle gleichzeitig

Bei CP/M Plus haben Sie die Möglichkeit, in einer Befehlszeile mehrere Befehle gleichzeitig anzugeben, die dann der Reihenfolge nach ausgeführt werden. Die einzelnen Befehle müssen lediglich durch Ausrufungszeichen getrennt sein.

Betrachten wir hierzu folgendes Beispiel:

*DIR ! ERA *.BAK ! SHOW*

Nach der Auflistung des Directories werden sämtliche BAK-Dateien gelöscht und schließlich mit SHOW der freie Diskettenspeicher angezeigt, der jetzt zur Verfügung steht.

Darüber hinaus besteht noch die Möglichkeit, mehrere Befehle in einer SUBMIT-Datei auf Diskette abzulegen. Durch einen einmaligen Aufruf dieser Datei werden dann sämtliche Befehle der Reihe nach ausgeführt. Mit dieser Technik werden wir uns in Kapitel 6 noch ausführlich beschäftigen.

2.10 Das Arbeiten mit gekaufter Software

Wenn Sie fertige CP/M-Software kaufen, die bereits für den CPC 6128 oder JOYCE angepaßt wurde, ist diese im allgemeinen sofort einsatzfertig. Wenn nicht, liegt meistens eine Beschreibung bei, was Sie noch tun müssen. In der Regel handelt es sich dabei um Dinge wie Auswahl der Bildschirmfarben, Änderung der Tastaturbelegung oder die Einbeziehung von Sonderzeichen. Zu diesem Zweck befindet sich normalerweise ein zusätzliches Programm auf der Diskette, das einen Namen wie CONFIG.COM oder ähnlich trägt. Wenn Sie dieses Programm ausführen, benötigen Sie keine

besonderen Kenntnisse, denn sämtliche Instruktionen werden meist ausführlich auf dem Bildschirm oder im beiliegenden Handbuch erklärt.

Bevor Sie nun mit der Software arbeiten, empfiehlt es sich, eine Arbeitskopie herzustellen, wie oben beschrieben, und die Orginaldiskette an einem sicheren Ort aufzubewahren. Fertige Software wird meist auf Disketten geliefert, die weder mit dem BOOT-Sektor noch mit dem CP/M-Betriebssystem versehen sind.

Zur Herstellung der Arbeitskopie gehen Sie am besten folgendermaßen vor: Als erstes formatieren Sie eine Diskette. Während es beim JOYCE nur ein Format gibt, wählen Sie beim CPC 6128 das Systemformat, wobei gleichzeitig der BOOT-Sektor mit aufgetragen wird. Dies reicht allerdings zum Starten des CP/M-Betriebssystems noch nicht aus, denn Sie benötigen hierzu noch die Datei C10CPM3.EMS für den CPC 6128 bzw. J12DCPM3.EMS für den JOYCE, die Sie mit PIP auf Ihre Arbeitsdiskette übertragen. Schließlich kopieren Sie, ebenfalls mit PIP, sämtliche Dateien der Orginaldiskette, zumindestens diejenigen, die Sie zum Arbeiten benötigen.

Das Arbeiten mit einer solchen Kopie ist besonders bequem, denn Sie können von ihr direkt nach dem Einschalten des Computers zunächst das CP/M-Betriebssystem und im Anschluß daran die Anwendersoftware laden, ohne daß Sie auf die Systemdiskette zurückgreifen müssen. Oft empfiehlt es sich, noch einige häufig benötigte Dienstprogramme, wie z.B. PIP oder SHOW, mit auf die Arbeitsdiskette zu kopieren. Beachten Sie diesbezüglich die Hinweise des Software-Herstellers.

3 Spezielles über CP/M Plus

Das Kapitel, das Sie jetzt lesen, bildet das Kernstück des gesamten Buches. Es beschreibt alle wesentlichen Befehle und Anwendungsmöglichkeiten, die CP/M Plus gegenüber der Vorgängerversion 2.2 aufweist. Wir beschäftigen uns hier unter anderem mit Timestamps (Zeiteinträgen im Directory) und Passwords (Schutzwörtern), aber auch mit gerätespezifischen Funktionen, wie der Änderung der Bildschirmfarben, der Tastaturbelegung und des Zeichensatzes.

Diejenigen Leser, die bereits mit CP/M 2.2 gearbeitet haben, vermissen hier vielleicht das Pogramm STAT. Mit STAT erhält man unter CP/M 2.2 wichtige Informationen über das verwendete CP/M-System sowie über Disketten und Dateien. Darüber hinaus dient es zum Setzen und Aufheben des Schreibschutzes und anderer Dateiattribute sowie zur Zuordnung logischer zu physikalischen Geräteeinheiten.

Die Aufgabe von STAT übernehmen unter CP/M Plus mehrere Programme, wie z.B. SET, SHOW und DEVICE. Dazu gehört auch der erweiterte DIR-Befehl, den wir im letzten Kapitel schon kurz kennengelernt haben. All diese Programme bieten aber noch viele zusätzliche Möglichkeiten, die diejenigen von STAT weit übertreffen.

Wenn Sie die ersten beiden Kapitel dieses Buches gut durchgearbeitet haben, müßten Sie jetzt in der Lage sein, die wichtigsten CP/M-Befehle anzuwenden und fertige Software einzusetzen. An dieser Stelle wird das Arbeiten mit CP/M Plus aber erst richtig interessant, denn wir lernen hier viele zusätzliche Befehle kennen, die über dieses Grundwissen weit hinausgehen.

3.1 Datum und Uhrzeit

Eine wichtige Eigenschaft von CP/M Plus, wie auch einiger weiter entwickelter Betriebssysteme, ist das Einstellen und Abfragen von Datum und Uhrzeit. Wie wir bereits wissen, enthält der Computer eine Z80A-CPU mit 4 MHz Taktfrequenz, die 4 Millionen Schwingungen pro Sekunde erzeugt. Ohne diese Einrichtung könnte ein Computer überhaupt nicht funktionieren. Mit Hilfe dieser Schwingungsintervalle kann man eine elektronische Uhr steuern, die solange läuft, wie der Computer eingeschaltet bleibt.

Auf den Systemdisketten befindet sich das Programm DATE, das wir zum Stellen und Abfragen der internen Uhr benötigen. Achten Sie darauf, daß die Zeiteingabe und -abfrage folgendes Format verwendet:

mm/tt/jj hh:mm:ss

Tag und Monat sind gegenüber der sonst üblichen Schreibweise vertauscht. Soll beispielsweise der 7. Dezember 1985, 22 Uhr, 23 Minuten, 35 Sekunden dargestellt werden, so erscheint dies als:

```
12/07/85 22:23:35
```

Dabei ist also nicht der 12. Juli gemeint! Mit dem gleichen Datum wollen wir nun die Uhr des Computers stellen, wozu wir

DATE 12/07/85 22:23:35

eingeben. Damit wir die Uhr auch richtig stellen können, erscheint zunächst:

```
Strike key to set time
```

d.h. wir geben die richtige Zeit vor und drücken erst dann eine beliebige Taste, wenn dieser Zeitpunkt erreicht ist.

Wollen wir später die Uhrzeit abfragen, geben wir lediglich

DATE

ein, worauf beispielsweise

```
Sat 12/07/85 22:44:13
```

erscheint. Wir sehen, daß der Computer hier gleichzeitig noch den Wochentag errechnet. Mit

DATE CONTINUOUS

oder abgekürzt:

DATE C

erhalten wir eine Digitaluhr, d.h. Datum und Uhrzeit werden fortlaufend auf dem Bildschirm ausgegeben. Durch Drücken einer beliebigen Taste können wir diesen Vorgang abbrechen und es folgt wieder das Anforderungszeichen.

Die interne Uhr dient aber nicht nur zur reinen Zeitabfrage, die wir eben kennengelernt haben. Ihre eigentliche Aufgabe besteht bei CP/M Plus

darin, das Directory-Label und einzelne Dateien mit Timestamps (Zeiteintragungen des letzten Zugriffs bzw. Aktualisierung) zu versehen. In den nächsten beiden Abschnitten lernen wir diese Möglichkeiten näher kennen.

3.2 Directory-Label und -Timestamps

Unter CP/M-Plus haben Sie die Möglichkeit, das Disketten-Directory mit einer Kennzeichnung (Label) zu versehen. Darunter kann man auch einen Namen verstehen, den man einer Diskette, ähnlich wie einer einzelnen Datei erteilt. Mit einem solchen Label ist es dann einfacher, die Diskette zu identifizieren.

Wenn Sie auf einer Diskette verschiedene Textdateien ablegen, können Sie sie beispielsweise mit dem Label TEXT.TXT kennzeichnen. Falls Sie eine Diskette mit Lagerdaten verwenden, würde hier der Name LAGER.DAT gut passen.

Wie Sie vielleicht schon bemerkt haben, unterliegen Diskettenlabels den gleichen Schreibregeln wie Dateinamen, d.h. Sie wählen einen Namen mit maximal acht und eine Extension mit maximal drei Zeichen.

Wenn Sie das Directory mit einem Label versehen, ist es gleichzeitig auch für Timestamps eingerichtet. Dies sind Datums- und Zeiteinträge, die angeben, zu welchem Zeitpunkt das Label erstellt oder geändert wurde.

Damit das Directory-Label nicht von Unbefugten geändert werden kann, können Sie es zusätzlich mit einem Password schützen, das vor jedem Zugriff auf das Label anzugeben ist.

Achtung! Das Password für das Directory-Label und die Timestamps für das Label sind nicht mit Passwords und Timestamps für Dateien zu verwechseln, mit denen wir uns im nächsten Abschnitt noch ausführlich beschäftigen werden. Im Augeblick behandeln wir ausschließlich das Directory-Label.

Um die folgenden Vorgänge besser zu verstehen, legen Sie sich am besten eine Diskette zurecht, mit der Sie die beschriebenen Anweisungen nachvollziehen können. Verwenden Sie dazu entweder eine fabrikneue Diskette oder eine solche, auf deren Inhalt Sie unbedingt verzichten können. Diese Diskette formatieren Sie jetzt ganz normal mit DISCKIT3 (CPC 6128) bzw. mit DISCKIT (JOYCE).

Jetzt führen Sie den Befehl

SHOW B:[DIR]

aus, mit dem Sie die freien Einträge im Directory abfragen können. Dazu legen Sie die Systemdiskette mit der Datei SHOW.COM in Laufwerk A und die frisch formatierte Diskette in Laufwerk B ein. Falls Sie nur ein Laufwerk verwenden, behalten Sie diese Regelung bei, indem Sie jeweils die Disketten wechseln, wenn Sie dazu aufgefordert werden.

Wir wissen bereits, daß das Directory für Schneider-Disketten Platz für maximal 64 Einträge bietet. Deshalb erhalten wir mit dem oben genannten Befehl

```
B: Number of free directory entries:       64
```

Bevor wir jetzt das Directory mit einem Label versehen, stellen Sie zunächst mit dem DATE-Befehl (siehe oben) Datum und Uhrzeit ein. Übrigens sollte dies immer dann geschehen, wenn Sie mit Disketten oder Dateien arbeiten, die Timestamps verwenden. Nun können wir den Namen (Label), der beispielsweise PROBE.DAT lautet, mit Hilfe des folgenden Befehls in das Directory eintragen:

SET B:[NAME=PROBE.DAT]

Nach erfolgreicher Operation erscheint:

```
Directory      Passwds  Stamp    Stamp    Stamp
Label          Reqd     Create   Access   Update
-------------  -------  -------  -------  -------
PROBE    .DAT    off      off      off      off
```

Die vier rechten Spalten für Passwords und Stamps interessieren uns im Augenblick noch nicht, denn sie betreffen ausschließlich Dateien, nicht aber das Directory-Label. Jetzt ordnen wir dem Label ein Password zu, das wir hier mit "GEHEIM" bezeichnen wollen. Es wäre aber auch jedes andere Wort mit maximal acht Zeichen Länge zulässig. Nun setzen wir das Password und geben dazu

SET B:[PASSWORD=GEHEIM]

ein. Wieder erscheinen die gleichen Meldungen wie oben und darunter die Bestätigung:

```
Password = GEHEIM
```

Das Password sollten wir uns gut merken, denn es dient ja zum Schutz des Labels. Wenn wir es vergessen haben, besteht keine Möglichkeit, es abzufragen, jedenfalls nicht mit normalen Mitteln. Eine Diskette, deren Directory-Label mit einem Password versehen ist, ist nicht vor Schreib- und Lesezugriffen geschützt, sondern lediglich vor einer unbefugten Änderung des Labels, das somit zur ständigen Identifikation der Diskette zur Verfü-

gung steht. Darüber hinaus wird es für Timestamps und Passwords von Dateien benötigt, wie wir im nächsten Abschnitt noch sehen werden. Mit dem folgenden Befehl können wir das Label abfragen, ungeachtet dessen, ob es geschützt ist oder nicht:

SHOW B:[LABEL]

Nun erscheinen folgende Angaben auf dem Bildschirm:

```
Directory      Passwds  Stamp    Stamp
Label          Reqd     Create   Update   Label Created     Label Updated
-------------  -------  -------  -------  ----------------  ----------------
PROBE    .DAT    off      off      off     12/15/85 18:40    12/15/85 18:40
```

Auch hier betreffen die Angaben in der zweiten bis vierten Spalte wieder nicht das Directory-Label, sondern nur Dateien. Links erscheint nun das Label PROBE.DAT, während die beiden rechten Spalten Datum und Uhrzeit für den Zeitpunkt angeben, an dem das Label erstellt (created) bzw. letztmals geändert (updated) wurde. Beide Zeiteinträge sind hier noch identisch. Wenn wir jetzt jedoch das Label ändern, erscheint bei einer erneuten Ausführung des Befehls in der letzten Spalte der Zeitpunkt der Änderung.

Wir wollen nun das Label von PROBE.DAT in TEXT.DAT umbenennen und geben deshalb erneut den Befehl

SET B:[NAME=TEXT.DAT]

ein. Dies funktioniert dieses Mal jedoch nicht so problemlos wie beim ersten Mal, da wir ja zwischenzeitlich das Label mit einem Password versehen haben. Und prompt fragt der Computer:

```
Directory Label
Password ?
```

Hoffentlich haben Sie das Password nicht vergessen! Wir erinnern uns jedoch, daß wir es mit "GEHEIM" bezeichnet haben, und geben es nun ein. Aber ganz so einfach scheint die Sache nicht zu sein, denn wenn wir das Wort eintippen, erscheint es nicht auf dem Bildschirm und selbst der Cursor bewegt sich nicht. Sehr schnell erscheint dann die Meldung:

```
ERROR: Wrong Password
Directory Label
```

Wir haben entweder ein falsches Password eingegeben oder das richtige falsch geschrieben. Daß die Eingabe hier nicht auf dem Bildschirm erscheint, ist kein Fehler im Betriebssystem, sondern eine zusätzliche Sicherheitsmaßnahme. Es könnte ja vorkommen, daß uns jemand über die Schul-

tern schaut, wenn wir das Password eingeben. Wir müssen es also blind eintippen, ohne jegliche Kontrollmöglichkeit, ob wir es auch richtig geschrieben haben.

Nachdem wir nun endlich das Password "GEHEIM" richtig eingegeben haben, erhält auch das Directory das neue Label TEXT.DAT. Mit dem Befehl

SHOW B:[LABEL]

überprüfen wir dies nochmals und stellen fest, daß der Zeitpunkt für die Änderung (Update) in der rechten Spalte angezeigt erscheint.

Eingangs hatten wir schon einmal den Befehl

SHOW [DIR]

ausgeführt und festgestellt, daß sämtliche 64 Directory-Einträge nicht belegt waren, da wir ja auf unserer Diskette noch keine Dateien gespeichert hatten. Da dies auch jetzt noch nicht der Fall ist, wiederholen wir den Befehl nochmals zur Kontrolle und stellen fest, daß nur noch 63 freie Plätze vorhanden sind. Dies hat durchaus seine Richtigkeit, denn das Label belegt einen Eintrag im Directory, den wir allerdings nicht mit dem DIR-Befehl auflisten können. Sicherheit kostet also Platz im Directory.

Selbstverständlich können Sie ein Directory auch nachträglich mit einem Label bzw. Password versehen, wenn auf der Diskette bereits Dateien abgelegt sind. In diesem Abschnitt wurde aber bewußt darauf verzichtet, um die Vorgehensweise besser darzustellen.

3.3 Passwords und Timestamps für Dateien

Im vorangehenden Abschnitt haben wir uns mit dem Directory-Label beschäftigt, das mit Zeitangaben versehen und mit einem Password geschützt werden kann. Nun werden wir uns mit einzelnen Dateien befassen.

Obwohl Passwords und Timestamps für das Directory-Label und für einzelne Dateien unabhängig voneinander erstellt werden, besteht dennoch ein gewisser Zusammenhang. Dateien können nämlich nur dann mit einem Password oder Zeiteintrag versehen werden, wenn vorher zumindest das Directory-Label definiert wurde. Dies allein bietet in der Praxis aber noch keinen ausreichenden Schutz vor unberechtigtem Zugriff auf Dateien. Der Password-Schutz für Dateien kann nämlich wahlweise an- und abgeschaltet werden, ohne daß das Datei-Password eingegeben werden muß. Dadurch wird letztlich der gesamte Schutzmechanismus überflüssig.

Glücklicherweise wird aber beim An- und Abschalten dieses Datei-Password-Schutzes das Password des Directory-Labels abgefragt, falls es erteilt wurde. Deshalb ist ein ausreichender Schutz erst durch Vorgabe eines Passwords sowohl für das Directory-Label, als auch für die betreffende Datei gegeben.

Sollen Dateien darüber hinaus noch mit Timestamps versehen werden, ist zunächst das Programm INITDIR auszuführen, wodurch im Directory Platz zur Ablage der Timestamps geschaffen wird. Diese Timestamps sind aber nicht mit denjenigen des Directory-Labels identisch. Datei-Timestamps sind jedoch nicht von der Erteilung eines Directory-Labels oder Datei-Passwords abhängig und funktionieren auch ohne Schutz. Als einzige Voraussetzung muß jedoch ein Directory-Label vorhanden sein, das, wie bereits gesagt, nicht geschützt sein muß.

Diese Vorgänge mögen Ihnen zunächst noch etwas kompliziert und undurchsichtig erscheinen, was in der Tat auch zutrifft, solange man sich noch nicht genau damit auskennt. Deshalb wollen wir wieder auf unsere Probediskette zurückgreifen, die wir im letzten Abschnitt angelegt haben. Sie enthält bereits das Directory-Label TEXT.DAT, das mit dem Password "GEHEIM" geschützt ist.

Kopieren Sie jetzt mit Hilfe von PIP eine Datei auf diese Diskette. Dabei spielt es keine Rolle, ob es sich um eine COM- oder sonstige Datei handelt. In unserem Beispiel nennen wir sie BRIEF.TXT.

Neben einem Password-Schutz wollen wir sie auch mit Timestamps versehen. Dazu legen wir unsere Diskette wieder in Laufwerk B und die Systemdiskette mit INITDIR.COM in Laufwerk A. Falls Sie nur mit einem Laufwerk arbeiten, gilt das gleiche, wie im letzten Abschnitt beschrieben.

Wir wissen bereits, daß INITDIR dazu benötigt wird, um Platz für die Timestamps im Directory zu reservieren. Dabei wird das Directory vollkommen neu organisiert, wobei für jeweils drei Einträge ein weiterer reserviert wird, in dem genügend Platz für sämtliche Timestamps von drei Dateien vorhanden ist.

Da das Directory maximal 64 Einträge aufnehmen kann, sind auf unserer Diskette jetzt immerhin noch 62 nicht belegt. Einen Eintrag beansprucht das Directory-Label und einen weiteren unsere Datei BRIEF.TXT, sofern sie nicht mehr als ein Extent beansprucht, d.h. nicht mehr als 16K belegt. Falls Sie sich hierüber nicht sicher sind, überprüfen Sie diese Angaben nochmals mit Hilfe von

SHOW B:[DIR]

Jetzt geben wir

INITDIR B:

ein, worauf die Neuordnung des Directories stattfindet. Auf dem Bildschirm erscheint zunächst

```
INITDIR WILL ACTIVATE TIME STAMPS FOR SPECIFIED DRIVE
Do you want to re-format the directory on drive: B  (Y/N)?
```

Dies ist eine letzte Sicherheitsabfrage. Wenn Sie jetzt Y eingeben, findet die Neuordnung statt, N dagegen bricht den Vorgang ab.

Wir geben Y ein und stoßen sogleich auf eine Password-Abfrage, denn unsere Diskette hat ja ein password-geschütztes Directory-Label:

```
Directory is password protected.
Password, please  >GEHEIM
```

Dieses Mal geben wir unser Password in offener Form ein, d.h. es erscheint auf dem Bildschirm. Diese Sicherheitsfrage würde entfallen, wenn wir das Directory zuerst mit INITDIR neu geordnet und erst dann mit einem Password versehen hätten. Timestamps, und somit die Ausführung von INITDIR, sind nämlich unabhängig von Passwords zulässig.

Jetzt geben wir erneut

SHOW B:[DIR]

ein und erhalten

```
B: Number of time/date directory entries:   16
B: Number of free directory entries     :   46
```

Die erste Zeile gibt an, daß für die Timestamps insgesamt 16 der 64 Directory-Einträge reserviert sind. Ziehen wir diesen Wert von dem ursprünglichen von 62 ab, so verbleiben noch 46 freie Directory-Einträge, wie in der zweiten Zeile angegeben. Das zur Ablage von Dateinamen reservierte Directory ist also um ein Viertel kleiner geworden.

Jetzt ist zwar das Directory zur Aufnahme von Timestamps vorbereitet, jedoch werden diese noch nicht automatisch eingetragen. Dazu ist nämlich eine spezielle Initialisierung notwendig, und zwar für folgende drei Zeiteinträge getrennt:

```
[CREATE=ON/OFF]      (Erstellen ein/aus)
[ACCESS=ON/OFF]      (Lese-Zugriff ein/aus)
[UPDATE=ON/OFF]      (Aktualisieren/ändern ein/aus)
```

ON aktiviert und OFF deaktiviert also den jeweiligen Timestamp. Wenn Sie CREATE und ACCESS gleichzeitig aktivieren, ist immer nur ACCESS wirksam, da beide Arten sich aus Platzgründen den Speicherplatz im Directory teilen müssen.

Um den optimalen Nutzen aus den Timestamps zu ziehen, werden wir jetzt mit Hilfe des SET-Befehls UPDATE und ACCESS aktivieren:

SET B:[ACCESS=ON,UPDATE=ON]

Vergessen Sie aber nicht, vorher die Computeruhr mit DATE zu stellen. Wenn Sie dies versäumen, beginnt die Zählung immer beim 15. Dezember 1982,dem Zeitpunkt, an dem vermutlich das Programm DATE geschrieben wurde.

Versuchen Sie jetzt einmal, die Datei BRIEF.TXT irgendwie zu aktualisieren und merken Sie sich die Uhrzeit. Sollte dies im Augenblick ohne größeren Aufwand nicht möglich sein, führen Sie einfach folgenden Befehl aus:

PIP B:BRIEF.TXT=B:BRIEF.TXT

Hiermit wird die Datei BRIEF.TXT in sich selbst kopiert, d.h. von PIP neu geschrieben. Bei diesem Vorgang wird aber die Zeitmarke UPDATE gesetzt.

Nach einigen Minuten versuchen Sie, einen beliebigen Lesezugriff auf die Datei vorzunehmen, z.B. mit dem Befehl

TYPE B:BRIEF.TXT

der sie auf dem Bildschirm listet. Sollten Sie eine COM-Datei gespeichert haben, können Sie TYPE nicht ausführen. Statt dessen führen Sie einfach die COM-Datei aus, wobei sie ja immerhin gelesen wird. Merken Sie sich auch hier den Zeitpunkt, der in der Marke ACCESS gespeichert wird. Schließlich geben Sie

DIR B:[FULL]

ein und erhalten das Directory auf eine ganz neue Weise aufgelistet:

```
Scanning Directory...

Sorting  Directory...


Directory For Drive B:  User  0

    Name      Bytes   Recs   Attributes   Prot      Update Access
------------ ------ ------ ------------ ------ ------------- -------------

BRIEF    TXT     1k      5 Dir RW       None   12/10/85 22:18  12/10/85 22:27

Total Bytes     =      1k  Total Records =      5  Files Found =     1
Total 1k Blocks =      1   Used/Max Dir  Entries for Drive B:  3/   64
```

Da Sie sich die Zeiten gemerkt haben, können Sie nun kontrollieren, ob sie auch richtig eingetragen sind.

Kommen wir nun zum Password-Schutz für Dateien. Als Beispiel wählen wir das Password SUSI, falls die Datei BRIEF.TXT einen Text mit dem Liebesbrief an die Freundin Susi enthält, den nicht jeder zu lesen braucht. Zuvor jedoch müssen wir den Password-Schutz für Dateien aktivieren, indem wir

SET B:[PROTECT=ON]

eingeben. Dabei werden wir zunächst einmal nach dem Password des Directory-Labels gefragt:

```
Directory Label
Password ?
```

Hier müssen wir das Password "GEHEIM" eingeben, dieses Mal wieder blind. Dann erscheint:

```
Directory     Passwds  Stamp    Stamp    Stamp
Label         Reqd     Create   Access   Update
------------  -------  -------  -------  -------
TEXT    .DAT   on       off      on       on
```

Dieser Auflistung entnehmen wir außer dem Directory-Label noch den Hinweis, daß das Directory jetzt sowohl für Datei-Passwords als auch für die Datei-Timestamps Access und Update aktiviert ist.

Die gleichen Angaben erhalten wir übrigens auch, wenn wir den Befehl SHOW [LABEL] (siehe oben) ausführen. Hier sind lediglich noch die Timestamps für das Directory-Label mit angegeben.

Schließlich müssen wir noch das Password eingeben, das auch in diesem Fall maximal acht Zeichen umfassen darf. Mit unserem Password "SUSI" geschieht dies folgendermaßen:

SET B:BRIEF.TXT [PASSWORD=SUSI]

Trat dabei kein Fehler auf, erscheint:

```
B:BRIEF   .TXT  Protection = READ, Password = SUSI
```

READ beinhaltet hier nicht nur einen Leseschutz für die Datei infolge des Passwords, sondern auch einen Schutz vor sämtlichen Zugriffen, einschließlich des Löschens mit ERA und des Umbenennens mit REN.

Nachdem wir nun die Datei mit einem Password versehen haben, wollen wir nochmals mit

SHOW B:[DIR]

feststellen, wieviele freie Directory-Einträge noch zur Verfügung stehen. Nach Ausführung von INITDIR waren es noch 46, jetzt sind jedoch nur noch 45 Plätze frei. Wir sehen also, daß ein Datei-Password einen ganzen Directory-Eintrag beansprucht.

Wie können wir nun auf eine password-geschützte Datei zugreifen? Dazu gibt es zwei Möglichkeiten: Entweder man gibt im Anschluß an den Dateinamen ein Semikolon und dann das Password ein oder das Password wird später wie gewohnt angefordert und ist dann blind einzutippen. Wenn wir nun unsere Textdatei BRIEF.TXT mit TYPE auflisten wollen, geschieht dies folgendermaßen:

TYPE B:BRIEF.TXT;SUSI <RETURN>

Vergessen wir das Password, müssen wir es anschließend blind eingeben. Dies ist jedoch beim Aufruf von COM-Dateien in der Regel nicht möglich, weshalb wir in einem solchen Fall auf die erste Möglichkeit zurückgreifen müssen, wie das folgende Beispiel zeigt:

BILANZ;MEIER <RETURN>

Hier wird die Datei BILANZ.COM unter Angabe des Passwords MEIER geladen und ausgeführt.

Fassen wir abschließend noch einmal zusammen:

Eine Diskette, auf der Dateien Passwords und/oder Timestamps erhalten sollen, ist zunächst mit einem Directory-Label zu versehen. Dieses Label selbst kann mit einem gesonderten Password geschützt werden, muß aber nicht. Es kann aber ohne weitere Vorbereitung (Label-)Timestamps aufnehmen, sofern die interne Uhr mit DATE gestellt ist.

Werden nun Dateien auf einer solchen Diskette abgelegt, können diese wahlweise mit einem Password vor unberechtigtem Zugriff gesichert werden. Dieser Schutz ist jedoch nur dann sinnvoll und voll wirksam, wenn gleichzeitig das Directory-Label mit einem Password geschützt ist. Der Password-Schutz für Dateien ist aber gesondert zu aktivieren.

Unabhängig davon können Dateien mit Timestamps versehen werden. Es spielt also keine Rolle, ob ein Password-Schutz für das Directory-Label oder für einzelne Dateinamen vorhanden ist oder nicht. Jedoch ist in jedem Fall das Directory zuvor mit INITDIR neu zu organisieren, um Platz zur Aufnahme der Timestamps zu schaffen. Damit diese richtig eingetragen werden, ist auch hier zunächst die interne Uhr zu stellen und die verschiedenen Timestamps sind gesondert zu aktivieren.

Eine mit Password geschützte Datei wird aufgerufen, indem zunächst ein Semikolon und dann das Password an den Dateinamen angefügt wird. Dies gilt besonders für COM-Dateien. Bei anderen Dateien besteht meist alternativ die Möglichkeit, das Password nach dem Aufruf blind einzugeben.

Wir haben hier die wichtigsten Befehle kennengelernt, die zum Arbeiten mit Passwords und Timestamps erforderlich sind. Es gibt jedoch noch einige weitere Befehle (z.B. zum Aufheben eines Passwords), die in Kapitel 10 ausführlich behandelt werden.

3.4 Schreibschutz für Laufwerk

CP/M Plus bietet die Möglichkeit, ein einzelnes Laufwerk mit einem Schreibschutz zu versehen, so daß Dateien nur gelesen werden können. Für Laufwerk A geschieht dies mit

SET A:[RO]

und für Laufwerk B mit

SET B:[RO]

Das betreffende Laufwerk ist hier unbedingt mit anzugeben. Es genügt also nicht, sich auf das Bezugslaufwerk zu stützen.

Dieser Befehl nimmt keinen Eintrag auf der Diskette, sondern nur im CP/M-Betriebssystem selbst vor. Der Schreibschutz wird deshalb durch Ausschalten des Computers, durch einen Warmstart infolge von CTRL-C oder durch folgenden Befehl wieder aufgehoben:

SET A:[RW]

bzw.

SET B:[RW]

Ein derartiger Schutz bietet also keine Sicherheit gegen unberechtigten Schreibzugriff auf die Diskette und ist nur zeitweilig wirksam. Er ist nur dann zu empfehlen, wenn versehentliches Überschreiben der in dem Laufwerk gespeicherten Dateien vermieden werden soll.

Achtung! Der hier beschriebene Schreibschutz für Laufwerke ist keinesfalls mit dem mechanischen Disketten-Schreibschutz zu verwechseln, der durch Betätigung des Schiebers bzw. der roten Klappe am Diskettengehäuse erreicht wird. Eine mechanisch gesicherte Diskette kann unter keinen Umständen beschrieben werden, selbst wenn Laufwerk und Dateien nicht mit einem Schreibschutz versehen sind.

3.5 Dateiattribute

Außer einem Laufwerk können auch einzelne Dateien vor Schreibzugriffen geschützt werden. Unabhängig davon können sie zu Systemdateien bestimmt werden, so daß sie mit dem einfachen DIR-Befehl nicht mehr aufzulisten sind. Ansonsten sind sie aber ganz normal aufzurufen und auszuführen.

Beides ist durch Setzen von Dateiattribute zu erreichen, wobei ein entsprechender Vermerk im Directory der Diskette angebracht wird. Dateiattribute sind unabhängig von eventuell gesetzten Timestamps oder Passwords.

Hier einige Beispiele:

SET B:MITGLIED.DAT [RO]

versieht in Laufwerk B die Datei MITGLIED.DAT mit einem Schreibschutz. Infolge von

SET B:MITGLIED.DAT [RW]

wird der Schreibschutz wieder aufgehoben.

Soll ein Dateiname nicht mit dem einfachen DIR-Befehl auflistbar sein, versehen Sie ihn mit dem Systemattribut wie in diesem Beispiel:

SET B:TEST.COM [SYS]

Der Befehl

SET B:TEST.COM [DIR]

ersetzt das Systemattribut durch das Directory-Attribut und hebt somit den Listschutz wieder auf.

Einen allzu großen Schutz bietet das Systemattribut allerdings nicht. Die Dateien sind nämlich nur bedingt kopiergeschützt (siehe PIP) und ansonsten wie normale Dateien zu behandeln. Selbst wenn sie mit dem einfachen DIR-Befehl nicht auflistbar sind, so ist dies doch durch den Spezialbefehl

DIRSYS

oder den Befehl

DIR [FULL]

durchaus möglich, wobei im letzten Fall das Attribut sogar noch mit angezeigt wird.

Dateiattribute können auch für mehrdeutige Dateinamen vorgegeben werden, um mit einem Befehl ganze Dateigruppen zu schützen.

3.6 Nützliches über den SET-Befehl

Wie wir in den vorangehenden Abschnitten gesehen haben, erfüllt der SET-Befehl eine vielfältige Aufgabe. So dient er beispielsweise zum Setzen und Aufheben von Passwords, zum Aktivieren der Timestamps oder zur Einrichtung bzw. Aufhebung eines Schreibschutzes für Dateien und Laufwerke.

SET ist also der universelle CP/M Plus-Befehl für die verschiedensten Schutzeinrichtungen. Diese sind zwar einzeln wirksam, können aber auch kombiniert werden. So kann eine Datei beispielsweise gleichzeitig mit Timestamps versehen sein, mit einem Password gesichert sein, einen Schreibschutz enthalten und mit dem Systemattribut gekennzeichnet sein. Sämtliche Schutzmaßnahmen können aber auch separat erteilt werden.

Die Option des SET-Befehls, die in eckigen Klammern anzugeben ist, kann aber auch mehrere Anweisungen gleichzeitig enthalten, so daß SET nur einmal auszuführen ist. Hier ein Beispiel:

SET B:BRIEF.TXT [RO,SYS,PROTECT=ON,PASSWORD=SUSI]

Durch diesen Befehl haben wir die Datei BRIEF.TXT in Laufwerk B schreibgeschützt, zur Systemdatei erklärt, für den Password-Schutz initialisiert und schließlich mit dem Password SUSI versehen. Dasselbe hätten wir selbstverständlich auch erreicht, wenn wir den SET-Befehl viermal hintereinander mit verschiedenen Optionen ausgeführt hätten, nämlich mit:

SET B:BRIEF.TXT [RO]
SET B:BRIEF.TXT [SYS]
SET B:BRIEF.TXT [PROTECT=ON]
SET B:BRIEF.TXT [PASSWORD=SUSI]

3.7 Benutzerbereiche

Unter CP/M kann man mit maximal 16 verschiedenen Benutzerbereichen arbeiten, die die Kennzeichnung 0 bis 15 erhalten. Benutzerbereiche sind immer dann nützlich, wenn mehrere Personen mit der gleichen Diskette arbeiten und jede ihren eigenen Bereich zugewiesen bekommt. Wenn wir beispielsweise

USER 4

eingeben, können wir nur auf solche Dateien zugreifen, die unter der Benutzernummer "4" abgelegt sind. Ein Zugriff auf Dateien anderer Benutzerbereiche ist also nicht möglich.

Der vorgegebene Benutzerbereich (außer "0") ist immer vor dem Anforderungszeichen angegeben. In unserem Beispiel erscheint also:

```
4A>
```

Listen wir jetzt das Directory auf, erscheinen nur solche Dateien, die im Benutzerbereich 4 abgelegt sind.

Wenn wir den Befehl

SHOW [USER]

ausführen, erhalten wir folgende Angaben über die Benutzerbereiche:

```
A: Active User : 4
A: Active Files: 1  4  7  12
A: # of files  : 25

A: Number of free directory entries:     39
```

In diesem Beispiel ist gerade der Benutzerbereich 4 eingestellt. Darüber hinaus sind in den Bereichen 1, 4, 7 und 12 Dateien abgelegt. Die Anzahl

der insgesamt gespeicherten Dateien beträgt 25, wobei noch 39 freie Directory-Einträge vorhanden sind.

Standardmäßig ist die Benutzernummer 0 voreingestellt. Die Verwendung einzelner Benutzerbereiche bringt auf dem CPC 6128 und JOYCE allerdings kaum Vorteile. Anders ist dies allerdings bei Mehrplatzsystemen, wie z.B. MP/M, wo mehrere Benutzer an verschiedenen Terminals arbeiten, denen jeweils eine Benutzernummer zugeordnet ist.

3.8 Mehr über DIR und SHOW

Die Befehle DIR und SHOW sind uns bereits häufiger begegnet. Mit ihren verschiedenen Zusatzangaben (Optionen) sind sie als die Standardbefehle unter CP/M Plus zu bezeichnen, die Angaben über Dateien, Laufwerke und das Betriebssystem abfragen.

In Kapitel 10 sind beide Befehle, einschließlich sämtlicher Optionen,ausführlich beschrieben. Trotzdem wollen wir die wichtigsten dieser Zusatzangeben, die wir noch nicht kennengelernt haben, an dieser Stelle kurz betrachten.

Beginnen wir mit dem DIR-Befehl, der das Directory auflistet. In seiner einfachen Form ist er uns längst bekannt, was wir hier nicht zu wiederholen brauchen. Viel interessanter ist jedoch der Befehl

DIR [FULL]

den wir zwar schon kennen, aber noch nicht vollständig behandelt haben. Falls das Directory für Timestamps aktiviert ist, erzeugt er eine Auflistung wie im folgenden Beispiel:

```
Scanning Directory...

Sorting  Directory...


Directory For Drive B:  User  0

    Name      Bytes   Recs   Attributes   Prot      Update Access
------------ ------ ------ ------------ ------ ------------- -------------

SPIEL    BAS     3k     18 Dir RW Arcv  Read   12/02/85 22:18  12/10/85 11:03
MUSTER   ASM     1k      5 Dir RO       Read   12/02/85 22:07  12/07/85 08:15
BILANZ   COM     1k      1 Dir RW       Read                   12/02/85 22:45
LAGER    DAT     1k      3 Dir RW       None
BIBLTHEK DAT     2k     12 Sys RO       None                   12/02/85 22:47

Total Bytes     =      8k  Total Records =      39  Files Found =    5
Total 1k Blocks =      8   Used/Max Dir Entries For Drive B:   11/  64
```

Neben dem Dateinamen erscheinen hier Angaben über den belegten Speicherplatz auf der Diskette in KByte und Records, über verschiedene Dateiattribute, den Passord-Schutz sowie den Zeitpunkt der letzten Aktualisierung (Update) und des letzten Zugriffs (Access).

Die Timestamps (Update u. Access) sind hier nur angezeigt, falls auf die betreffenden Dateien schon ein Zugriff stattfand, seit das Programm INITDIR ausgeführt und die Zeiterfassung aktiviert wurde. Hier erfuhren lediglich die ersten beiden Dateien eine Aktualisierung, während auf alle, außer LAGER.DAT, zumindest einmal ein Lesezugriff vorgenommen wurde.

Kommen wir jetzt zu den Attributen. "Dir" kennzeichnet eine Directory- und "Sys" eine Systemdatei (siehe Kapitel 3.5). Darüber hinaus sind die Dateien MUSTER.ASM und BIBLTHEK.DAT schreibgeschützt (RO), während auf die restlichen auch Schreibzugriffe vorgenommen werden können. Die Datei SPIEL.BAS ist außerdem noch mit dem Arcv-(Archive-)Attribut gekennzeichnet, das bei PIP-Operationen eine Rolle spielt und in Kapitel 10 erklärt wird.

Die Spalte "Prot" gibt Aufschluß über den Password-Schutz. Er ist bei "Read" gegeben, bei "None" dagegen nicht. Bei "Read" ist der Password-Schutz für alle Zugriffsarten wirksam, jedoch können Dateien auch nur zum Schreiben oder zum Löschen mit einem Password geschützt sein, wobei in dieser Spalte "Write" bzw. "Delete" erscheint. Mehr hierüber erfahren Sie in Kapitel 10.

Wollen Sie das erweiterte Directory sämtlicher angeschlossenen Laufwerke auf einmal auflisten, geben Sie ein:

DIR [DRIVE=ALL]

Dieser Befehl ist jedoch nur für solche Laufwerke wirksam, auf die seit dem Booten des CP/M-Betriebssystems schon einmal zugegriffen wurde. Dabei werden Sie jedoch feststellen, daß der Listvorgang immer dann unterbrochen wird und die Meldung

```
Press RETURN to Continue
```

erscheint, wenn der Bildschirm vollgeschrieben ist. Dies geschieht aber nicht nur beim DIR-Befehl, sondern auch bei einigen anderen Befehlen, wie beispielsweise bei TYPE. In manchen Fällen ist das ständige Drücken der RETURN-Taste sicher lästig, um den Listvorgang fortzusetzen. Durch den Zusatz NOPAGE ist dies jedoch zu umgehen. Geben Sie deshalb

DIR [DRIVE=ALL,NOPAGE]

ein und die Directories sämtlicher Laufwerke erscheinen ohne Unterbrechung.

Manchmal benötigen wir nicht sämtliche Angaben, die uns der erweiterte DIR-Befehl liefert, jedoch genügt uns auch der einfache Befehl nicht, da er keine Auskunft über den belegten Speicherplatz einzelner Dateien auf der Diskette gibt. Hier ist der Befehl

DIR [SIZE]

nützlich, der beispielsweise für die Systemdiskette des CPC 6128, Seite 1, folgende Angaben liefert:

```
Scanning Directory...

Sorting  Directory...

Directory For Drive A:  User  0

A: AMSDOS   COM    1k : BANKMAN  BAS    1k : BANKMAN  BIN    2k
A: C10CPM3  EMS   25k : DATE     COM    3k : DEVICE   COM    8k
A: DIR      COM   15k : DISCKIT3 COM    6k : ED       COM   10k
A: ERASE    COM    4k : GET      COM    7k : KEYS     CCP    1k
A: KEYS     WP     1k : LANGUAGE COM    1k : PALETTE  COM    1k
A: PIP      COM    9k : PROFILE  ENG    1k : PUT      COM    7k
A: RENAME   COM    3k : SET      COM   11k : SET24X80 COM    1k
A: SETDEF   COM    4k : SETKEYS  COM    2k : SETLST   COM    2k
A: SETSIO   COM    2k : SHOW     COM    9k : SUBMIT   COM    6k
A: TYPE     COM    3k

Total Bytes     =    146k  Total Records =    1093  Files Found =   28
Total 1k Blocks =    146   Used/Max Dir Entries For Drive A:   29/  64
```

Das Directory erscheint hier dreispaltig, in komprimierter Form, einschließlich des jeweiligen Dateiumfangs in KByte.

Auch beim DIR-Befehl lassen sich verschiedene Optionen kombinieren, wie im folgenden Beispiel:

DIR [DRIVE=ALL,SIZE,NOPAGE]

Hier wird das Directory sämtlicher angeschlossenen Laufwerke aufgelistet, allerdings nicht in der vollständig erweiterten Form, sondern nur mit Angabe der Dateigrößen. Außerdem wird der Listvorgang ohne Unterbrechung fortgesetzt, wenn der Bildschirm vollgeschrieben ist.

Kommen wir nun zum SHOW-Befehl. Wir haben ihn bereits in Form von

SHOW

(ohne Zusatz) kennengelernt, um Angaben über den noch freien Diskettenspeicher aller angeschlossenen Laufwerke zu erhalten. Erscheint hier z.B.:

```
A: RW, Space:     54k
B: RO, Space:     29k
```

so können auf Laufwerk A noch 54 KByte und 29 KByte auf Laufwerk B belegt werden. Während Laufwerk B mit einem temporären Schreibschutz (RO) versehen ist, können auf Laufwerk A sowohl Schreib- als auch Lesezugriffe vorgenommen werden.

Auch kennen wir bereits den Befehl, der die noch freien Directory-Einträge angibt:

SHOW [DIR]

Falls das Directory mit Hilfe von INITDIR zur Aufnahme von Timestamps vorbereitet wurde, erscheint zusätzlich die Anzahl der für diesen Zweck reservierten Einträge (siehe Kapitel 3.3).

Ebenfalls in Berührung kamen wir bereits mit dem Befehl

SHOW [LABEL]

der folgende Angaben liefert:

```
Directory     Passwds  Stamp    Stamp
Label         Reqd     Access   Update   Label Created    Label Updated
------------  -------  -------  -------  ---------------  ---------------
PROBE   .DAT    on       off      off     12/01/85 12:15   12/15/85 18:40
```

In diesem Beispiel erhalten wir das Directory-Label PROBE.DAT und den Zeitpunkt, an dem es erstellt und zum letzten Mal geändert wurde. Die restlichen Angaben betreffen die Dateien auf der Diskette, für die hier der Password-Schutz aktiviert ist, nicht jedoch die Eintragung von Timestamps.

Neu ist der folgende SHOW-Befehl, mit dem wir nähere Einzelheiten über ein Diskettenlaufwerk in Erfahrung bringen. Wir rufen ihn mit

SHOW [DRIVE]

für das Bezugslaufwerk oder mit

SHOW B:[DRIVE]

für Laufwerk B auf und erhalten für ein 3-Zoll-Laufwerk des CPC 6128 folgende Angaben:

```
    A: Drive Characteristics
1,368: 128 Byte Record Capacity
  171: Kilobyte Drive  Capacity
   64: 32  Byte Directory Entries
   64: Checked  Directory Entries
  128: Records / Directory Entry
    8: Records / Block
   36: Records / Track
    2: Reserved Tracks
  512: Bytes / Physical Record
```

Beginnen wir von oben nach unten. In diesem Beispiel wurden die Eigenschaften von Laufwerk A aufgelistet, welches am Anfang der ersten Zeile angegeben ist. Für Laufwerk B würden wir aber die gleichen Angaben erhalten, sofern es sich hierbei ebenfalls um ein 3-Zoll-Laufwerk für Schneider-Computer handelt. Die zweite Zeile gibt an, daß das Laufwerk eine Kapazität von 1368 Records zu je 128 Bytes umfaßt. Dabei sind allerdings die Systemspuren nicht mit enthalten. Insgesamt können wir also

1368 * 128 = 175104 Byte

auf der Diskette mit Dateien belegen, was

175104 / 1024 = 171 KByte

entspricht. Dabei sind wir auch schon bei der dritten Zeile, die diesen Wert angibt.

Die nächsten beiden Zeilen machen Angaben über das Directory. Wir erfahren, daß ein Directory-Eintrag intern 32 Bytes beansprucht und wir maximal 64 Einträge vornehmen können. Sämtliche 64 Einträge können wir auch mit einem Schreibschutz versehen (RO), was durch den Zusatz "Checked" angegeben wird.

Als weiteres erfahren wir, daß ein Directory-Eintrag (Extent) maximal 128 Records verwalten kann. Wenn wir dies nachrechnen, erhalten wir

128 * 128 = 16384 Byte = 16 KByte

pro Extent bzw. Eintrag. Darüber hinaus befinden sich 8 Records in einem Block (1024 Byte) und 36 Records auf einer Spur (Track). Außerdem sind zwei Systemspuren reserviert, wobei es sich um die beiden äußeren Spuren 0 und 1 handelt, die den Boot-Sektor aufnehmen und deshalb nicht zum Speichern von Dateien zur Verfügung stehen. Schließlich gibt die unterste Zeile an, daß ein physikalischer Record, den wir normalerweise aber als Sektor bezeichnen, auf der Diskette 512 Bytes umfaßt.

Das Laufwerk des JOYCE weist Daten auf, die sich geringfügig von dem des CPC 6128 unterscheiden:

```
   A: Drive Characteristics
1,400: 128 Byte Record Capacity
  175: Kilobyte Drive  Capacity
   64: 32  Byte Directory Entries
   64: Checked  Directory Entries
  128: Records / Directory Entry
    8: Records / Block
   36: Records / Track
    1: Reserved Tracks
  512: Bytes / Physical Record
```

Dieses Laufwerk verwaltet nur eine Systemspur, weshalb sich die freie Kapazität auf 175 KByte bzw. 1400 Records erhöht.

3.9 Gerätezuordnung

CP/M Plus verfügt über verschiedene logische Ein- und Ausgabekanäle, denen physikalische Geräte zugeordnet sind. Auf der Systemdiskette befindet sich das Programm DEVICE, das diese Zuordnung abfragt oder neu vornimmt. Geben Sie jetzt einmal

DEVICE

ein, worauf Sie folgende Angaben erhalten:

```
Physical Devices:
I=Input,O=Output,S=Serial,X=Xon-Xoff
CRT    NONE  IO     LPT    NONE  O

Current Assignments:
CONIN:  = CRT
CONOUT: = CRT
AUXIN:  = Null Device
AUXOUT: = Null Device
LST:    = LPT

Enter new assignment or hit RETURN
```

Wir erfahren, daß dem System derzeit die physikalischen Geräte CRT und LPT zugeordnet sind. CRT steht hier für das Terminal, wobei die Tastatur ein Eingabe- und der Bildschirm ein Ausgabegerät darstellt. LPT dagegen stellt nichts anderes als den Drucker dar.

Unter "Current Assignments" ist nun die Zuordnung der logischen Ein- und Ausgabekanäle zu diesen Geräten angegeben. CP/M Plus verfügt über insgesamt fünf solcher Kanäle, die hier allesamt aufgeführt sind. CONIN ist der Eingabekanal für das Terminal, der hier CRT (Tastatur) zugeordnet ist. Anders verhält es sich beim nächsten Kanal CONOUT, dem Ausgabekanal für das Terminal. Unter CRT ist hier also der Bildschirm zu verstehen.

Die nächsten beiden Kanäle AUXIN und AUXOUT sind keinem Gerät zugeordnet und deshalb mit "Null Device" gekennzeichnet. Sie können vom Benutzer frei definiert werden und beispielsweise als Ein- und Ausgabekanal für eine serielle Schnittstelle dienen, über die der Computer mit einem Modem oder Akustikkoppler verbunden ist. Der LST-Kanal ist schließlich dem Drucker zugeordnet und steuert die Centronics-Schnittstelle an.

Diese Gerätezuordnungen sind zwar standardmäßig voreingestellt, können aber durchaus geändert werden. Deshalb fragt die letzte Zeile, ob eine Änderung vorgenommen werden soll; ansonsten ist das Programm durch Drükken der RETURN-Taste wieder zu verlassen.

Für den Normalfall kann diese standardmäßige Belegung ohne weiteres beibehalten werden. Es gibt aber Fälle, in denen eine Änderung angebracht ist. Meistens handelt es sich dabei um die bereits erwähnte serielle Schnittstelle, die außer zur Datenfernübertragung auch zur Ansteuerung einiger Drucker benötigt wird.

Wenn Sie jetzt beispielsweise

CONOUT: = LPT

eingeben, wird alles, was normalerweise auf dem Bildschirm erscheinen müßte, über den Drucker ausgegeben. Sie können auch den LST-Kanal ausschalten, indem Sie

LST: = Null Device

vorgeben. Diese Möglichkeit ist durchaus denkbar, wenn Sie ohne Drucker arbeiten müssen. Drücken Sie dann nämlich versehentlich CTRL-P, hängt der Rechner in einer Warteschleife, die nur schwer zu verlassen ist, wenn kein Drucker angeschlossen ist. Dieses Mißgeschick passiert jedoch nicht, wenn der LST-Kanal nicht aktiviert ist.

Dem CPC 6128 und JOYCE können andere physikalische Geräte jedoch nur dann zugeordnet werden, wenn eine entsprechende Änderung im BIOS vorgenommen wird. Das BIOS enthält nämlich eine Tabelle, in der sämtliche Gerätebezeichnungen eingetragen sind. Standardmäßig sind nur die beiden Geräte CRT und LPT vorgesehen und selbst eine serielle Schnittstelle muß erst softwarewäßig angepaßt werden. Diese Aufgabe übernehmen aber meist die mitgelieferten Terminalprogramme, die auch einen Gerätenamen für die Schnittstelle in der Tabelle implementieren.

3.10 Zeichensatz und Tastaturbelegung

Der CPC 6128 und JOYCE sind mit verschiedenen internationalen Zeichensätzen ausgestattet, die jedoch gesondert aktiviert werden müssen. Häufig besteht nämlich der Wunsch, unter CP/M Plus auch mit dem deutschen Zeichensatz zu arbeiten. Während er beim JOYCE bereits voreingestellt ist, muß er beim CPC 6128 erst eingerichtet werden.

Auf der Systemdiskette befindet sich die Datei LANGUAGE.COM, mit der man acht verschiedene Zeichensätze einstellen kann. Wenn Sie jetzt

LANGUAGE 2

eingeben, wird der deutsche Zeichensatz aktiviert und infolge von

LANGUAGE 0

erscheint wieder der amerikanische. Achten Sie aber darauf, daß die deutschen Sonderzeichen einige amerikanische Zeichen ersetzen. Darunter fallen auch die eckigen Klammern, die für viele CP/M Plus-Befehle benötigt werden. Hier gilt dann das gleiche wie für den JOYCE, daß Sie nämlich beispielsweise anstelle von

DIR [FULL]

DIR ÄFULLÜ

eingeben müssen, was schon einige Umstellung erfordert.

Bei Verwendung des deutschen Zeichensatzes ist es sinnvoll, auch die Tastatur entsprechend anzupassen. Natürlich besteht die Möglichkeit, auf dem CPC 6128 eine vollständige DIN-Tastatur zu erzeugen. Dafür ist aber eine große Anzahl von Tasten neu zu definieren, von denen die meisten nur relativ selten benutzte Zeichen enthalten. Darüber hinaus müßten Sie sich auch eine große Anzahl entsprechender Aufkleber herstellen, um die Tasten nicht zu verwecheln.

Wir wollen einen Zwischenweg gehen und nur einige Tasten umbelegen, deren Zeichen hier ohnehin nicht benutzt werden. Selbst wenn die Tastatur dann nicht hundertprozentig der DIN-Norm entspricht, kann man doch schon recht gut mit ihr arbeiten. Hier die einzelnen Änderungen:

Y- und Z-Taste vertauschen
ü, Ü auf Klammeraffen-Taste (@ und |)
ö, Ö auf Taste für eckige Klammer auf ([)
ä, Ä auf Taste für eckige Klammer zu (])
ß auf geshiftete Pfeiltaste (anstelle des £-Zeichens)
§ auf die Schrägstrich-Taste (\) rechts unten

In diesem Zusammmenhang wollen wir auch gleichzeitig die Belegung von Funktionstasten kennenlernen. Als Beispiel dafür belegen wir die Taste f0 mit dem DIR (FULL)-Befehl.

Die Umbelegung der Tastatur nimmt das Programm SETKEYS vor, das eine ASCII-Textdatei mit den entsprechenden Änderungen einliest. Diese Datei, die wir hier DEUTSCH.KEY nennen wollen, sieht beispielsweise so aus:

```
71 N "^'121'"  y statt z
71 S "^' 89'"  Y statt Z
71 C "^' 25'"  CTRL-Y statt CTRL-Z
43 N "^'122'"  z statt y
43 S "^' 90'"  Z statt Y
43 C "^' 26'"  CTRL-Z statt CTRL-Y
22 N "^' 64'"  Paragraph statt \
24 S "^'126'"  scharfes s statt £
26 N "^'125'"  ue statt @
26 S "^' 93'"  UE statt |
17 N "^'124'"  oe statt [
17 S "^' 92'"  OE statt {
19 N "^'123'"  ae statt ]
19 S "^' 91'"  AE statt }
15 N "^'128'" Erweiterungs-Zeichen 128 fuer Taste f0
E 128 "DIR [full] ^M" String fuer Erweiterungs-Zeichen 128
```

Die Datei können Sie mit jedem Texteditor, wie z.B. ED (siehe Kapitel 4) erstellen. Falls Sie jedoch kein Textverarbeitungsprogramm besitzen und die umständliche Anwendung von ED nicht gewohnt sind, können Sie diese Datei auch mit folgendem BASIC-Programm erzeugen, in dem die einzelnen Textzeilen als DATAs angegeben sind:

```
100 OPENOUT "deutsch.key"
110 FOR i=1 TO 16
120 READ a$
140 PRINT #9, a$
150 NEXT
160 CLOSEOUT
170 END
1000 DATA  71 N "^'121'"  y statt z
1010 DATA  71 S "^' 89'"  Y statt Z
```

```
1020 DATA  71 C "^' 25'"  CTRL-Y statt CTRL-Z
1030 DATA  43 N "^'122'"  z statt y
1040 DATA  43 S "^' 90'"  Z statt Y
1050 DATA  43 C "^' 26'"  CTRL-Z statt CTRL-Y
1060 DATA  22 N "^' 64'"  Paragraph statt \
1070 DATA  24 S "^'126'"  scharfes s statt £
1080 DATA  26 N "^'125'"  ue statt @
1090 DATA  26 S "^' 93'"  UE statt |
1100 DATA  17 N "^'124'"  oe statt [
1110 DATA  17 S "^' 92'"  OE statt {
1120 DATA  19 N "^'123'"  ae statt ]
1130 DATA  19 S "^' 91'"  AE statt }
1140 DATA  15 N "^'128'" Erweiterungs-Zeichen 128 fuer Taste f0
1150 DATA  E 128 "DIR [full] ^M" String fuer Erweiterungs-Zeichen 128
```

Ist die Datei auf Diskette geschrieben, erfolgt die Umschaltung mit

SETKEYS DEUTSCH.KEY

Hier noch ein paar Worte zum Aufbau der Datei: Zuerst ist die Nummer der betreffenden Taste anzugeben. In Ihrem Handbuch befindet sich eine Skizze, die die Nummern sämtlicher Tasten enthält. Nach einem Leerzeichen folgt N, S oder C, wodurch die Taste in Normal-, SHIFT- oder CTRL-Stellung angegeben wird. Anschließend steht die neue Belegung in Anführungszeichen. Betrachten wir dazu die erste Zeile als Beispiel:

Taste 71 entspricht normalerweise dem Buchstaben Z, auf einer deutschen Tastatur jedoch dem Zeichen Y. In Normalstellung soll sie jetzt den Kleinbuchstaben y erzeugen, der dem ASCII-Code 121 (s. Anhang) zugeordnet ist. Hinter der Zuordnung kann die Zeile einen beliebigen Text zur Erläuterung enthalten, ähnlich wie eine REM-Zeile in BASIC.

Neben der hier verwendeten Form könnte die erste Zeile auch folgendermaßen geschrieben werden:

```
71 N "^'#79'"   ASCII-Code hexadezimal oder
71 N "y"        lediglich Angabe des Zeichens y
```

In der zweiten Zeile könnte entsprechend das Zeichen Groß-Y definiert werden. Die dritte Zeile steht jedoch für CTRL-Y, die wir auch so hätten schreiben können:

```
71 C "^Y"
```

Kommen wir jetzt zu den letzen beiden Zeilen, die die Funktionstaste f0 belegen. Dieser Taste, mit der Nummer 15, wird zunächst ein Erweiterungszeichen zwischen 128 und 159 (hier 128) zugeordnet. Anschließend weist die nächste Zeile, die mit einem "E" beginnen muß, dem Erweiterungszeichen den String "DIR [FULL] ^M" zu. "^M" steht hier für Carriage-Return und hat die gleiche Wirkung wie das Drücken der RETURN-

Taste. Würden wir ^M fortlassen, müßten wir nach dem Drücken der Taste f0 zusätzlich noch die RETURN-Taste drücken, um das Directory aufzulisten.

Auf der Systemdiskette des CPC 6128 befinden sich noch zwei weitere Textdateien, die zur Tastaturbelegung dienen. Wenn Sie die Datei KEYS.CCP in Form von

SETKEYS KEYS.CCP

aufrufen, sind einige Tasten aktiviert, die das Arbeiten mit der CP/M-Befehlszeile erleichtern. Die zweite Datei KEYS.WP richtet einige Tastenfunktionen ein, die sich zur Textverarbeitung besonders gut eignen.

Es wäre besonders vorteilhaft, wenn Sie die hier vorgestellte Datei DEUTSCH.KEY mit KEYS.CCP oder KEYS.WP kombinieren, indem Sie beide Dateien einfach aneinanderhängen und als eine Datei ausführen. Das Aneinanderhängen von Textdateien ist unter CP/M sehr einfach mit PIP (siehe Kapitel 5) zu realisieren.

3.11 Bildschirmfarben

Kommen wir nun zum Befehl PALETTE, mit dem sich unter CP/M Plus verschiedene Bildschirmfarben erzeugen lassen. Er wird in der Form

PALETTE Hintergrundfarbe, Schriftfarbe

ausgeführt. Wenn Sie CP/M Plus auf dem CPC 6128 aufrufen, erscheint leuchtendweiße Schrift (Farbe Nr. 63) auf blauem Hintergrund (Farbe Nr. 2). Diese Kombination entspricht dem Befehl

PALETTE 2, 63

Wenn Sie beispielsweise

PALETTE 8, 60

eingeben, erhalten Sie gelbe Schrift auf rotem Grund.

Hier die einzelnen Farbnummern, die sich von denen in BASIC jedoch unterscheiden:

Nr.	Farbe	Nr.	Farbe	Nr.	Farbe
0	Schwarz	32	Grün	48	Hellgrün
2	Blau	34	Blaugrün	50	Seegrün
3	Hellblau	35	Himmelblau	51	Helles Blaugrün
8	Rot	40	Gelb	56	Limonengrün
10	Magenta	42	Weiß	58	Pastellgrün
11	Hellviolett	43	Pastellblau	59	Pastellblaugrün
12	Hellrot	44	Orange	60	Hellgelb
14	Purpur	46	Rosa	62	Pastellgelb
15	Helles Magenta	47	Pastellmagenta	63	Leuchtendweiß

3.12 HELP

Auf Ihren Systemdisketten befinden sich die beiden Dateien HELP.COM und HELP.HLP. Sie enthalten Kurzinformationen über CP/M Plus, die Sie abrufen können, wenn Sie einmal nicht mehr weiter wissen. Die in HELP gegebenen Erläuterungen sind jedoch leider nur in englischer Sprache abgefaßt und können natürlich keine ausführliche Anleitung ersetzen. Durch Aufruf von

HELP

(ohne Zusatz) erscheint eine Auflistung von Stichworten auf dem Bildschirm, die meistens einzelnen Befehlen zugeordnet sind. Durch Eingabe eines dieser Begriffe erhalten Sie dann die dazugehörigen Informationen. Wenn Sie bespielsweise Angaben über den DIR-Befehl benötigen, geben Sie einfach

DIR

ein. Es besteht auch die Möglichkeit, HELP unter Angabe eines Begriffs aufzurufen. In unserem Beispiel geschieht dies mit

HELP DIR

Die meisten Begriffe sind nach ihrem Aufruf in weitere Unterbegriffe untergliedert. Im Falle von DIR erscheint dann nach Erläuterung des Hauptbegriffs

```
ENTER .subtopic FOR INFORMATION OF THE FOLLOWING SUBTOPICS:

BUILT-IN      WITHOPTIONS
```

Wenn Sie jetzt beispielsweise das Stichwort BUILT-IN interessiert, geben Sie es mit einem vorangestellten Punkt ein:

.BUILT-IN

Sie erhalten dann Angaben über die residenten DIR-Befehle. Von dieser Ebene aus können Sie in der gleichen Weise Begriffe der darunter liegenden Ebene aufrufen, falls welche angegeben sind. In unserem Fall erscheint noch das Stichwort EXAMPLES.

Von jeder Ebene aus können Sie jeden verfügbaren Unterbegriff auf einer untergeordneten Ebene aufrufen. Wenn uns beispielsweise nur die residenten Befehle von DIR interessieren, können wir dies gleich beim Aufruf von HELP mit angeben:

HELP DIR BUILT-IN EXAMPLES

Befinden wir uns aber schon in der Ebene von DIR, geben wir nur noch

.BUILT-IN EXAMPLES

(mit vorangestelltem Punkt!) ein.

4 Der Editor

4.1 Was ist ein Editor?

Den Begriff Editor haben wir bereits im ersten Kapitel kennengelernt. Allgemein handelt es sich dabei um ein internes Maschinenprogramm, das dem Benutzer gestattet, Texte in den Computer einzugeben oder vorhandene Texte zu korrigieren.

Was ist in diesem Zusammenhang mit einem Text gemeint? Sicher werden Sie jetzt antworten, daß z.B. die Zeilen, die Sie in diesem Buch lesen, einen Text darstellen. Damit haben Sie vollkommen recht, aber lassen Sie uns den Begriff Text einmal etwas genauer definieren.

Ein Text ist eine beliebige Aneinanderreihung von Text- oder ASCII-Zeichen, im Gegensatz zu jedem anderen Code, wie beispielsweise dem internen BASIC- oder Maschinencode. Dem Computer ist es letztlich egal, ob Sie in seinem Speicher Textzeichen oder etwas anderes ablegen, es kommt nur darauf an, nach welcher Definition er die gespeicherten Informationen handhabt. So gibt es beispielsweise den Z80-Maschinencode, der ganz bestimmten Regeln unterliegt, damit ihn die CPU richtig abarbeitet. In einem Maschinenprogramm ist also festgelegt, oder die Vereinbarung getroffen, daß es sich hierbei um Maschineninstruktionen handelt. Entsprechende Regeln, wie der Code intern im Speicher abgelegt wird, gelten auch für BASIC-Programme. Hier wurde nämlich die Vereinbarung getroffen, daß es sich um einen Code handelt, den der BASIC-Interpreter "versteht" und abarbeiten kann.

Ganz gleich, ob Sie nun ein BASIC- oder Maschinenprogramm schreiben oder nur normale Texte eingeben, Sie benötigen in jedem Fall einen Editor, mit dem Sie die entsprechenden Informationen in einer allgemein lesbaren Form eingeben bzw. bearbeiten können. Bei einem Text, wie im vorliegenden Buch, ist dies noch am einfachsten zu verstehen, denn für jedes Textzeichen gibt es einen äquivalenten ASCII-Code, der im Anhang in einer Tabelle vollständig aufgeführt ist. So entspricht z.B. das Zeichen "A" dem ASCII-Code 65 (dezimal), das Zeichen "B" dem Code 66 (dezimal) usw. Ähnliches gilt auch für die Kleinbuchstaben, die mit dem Wert 97 beginnen. Auch den Ziffern und Satzzeichen, wie Komma oder Punkt, ist ein ASCII-Wert zugeordnet.

Wenn Sie z.B. das Wort "Hallo!" mit dem Computer erfassen, und Sie sehen sich die entsprechenden Speicherzellen an, so finden Sie folgenden ASCII-Code vor:

```
48   61   6C   6C   6F  21     (Hexadezimale Schreibweise)
72   97  108  108  111  33     (Dezimale Schreibweise)
 H    a    l    l    o   !     (Entsprechende Textzeichen)
```

Ob Sie nun die hexadezimale (sedezimale) oder die dezimale Darstellung erhalten, hängt davon ab, auf welche Weise Sie die Speicherzellen auflisten. Wenn Sie dies von BASIC aus mit dem PEEK-Befehl vornehmen, erhalten Sie in der Regel dezimale Zahlenwerte, während ein Debugger meist mit hexadezimaler Darstellung arbeitet. Ein Debugger ist übrigens ein Hilfsprogramm zum Auflisten und Abändern von Speicherinhalten. Wir werden uns damit noch ausführlich in Kapitel 7 befassen.

Wenn nun der Computer auf diese Zeichen stößt und "weiß", daß er einen Text ausgeben soll, erscheint entsprechend das Wort "Hallo!". Weiß er es aber nicht, und wurde fälschlicherweise die Vereinbarung getroffen, daß es sich bei diesen Informationen um einen Maschinencode handelt, so hat dies wahrscheinlich verheerende Folgen.

Ein Editor erfaßt also immer Textzeichen und legt sie zunächst in Form von ASCII-Codes im Speicher ab. Natürlich stellt sich dann die Frage, was mit diesen Codes weiter geschieht. Der BASIC-Editor z.B. wandelt die Textzeichen in den internen BASIC-Code um, der dann vom Interpreter abgearbeitet werden kann.

Selbst wenn wir ein Maschinenprogramm schreiben, kann dies nur auf Umwegen über einen Editor, gleich welcher Art, geschehen, da wir den Maschinencode in lesbarer Form editieren müssen, bevor er in die Speicherzellen eingeschrieben wird.

Wollen wir nun umgekehrt das Wort "Hallo!" nicht im Klartext, sondern in Form von einzelnen ASCII-Codes mit einem Debugger in den Speicher schreiben, müssen wir selbst diese Hexcodes editieren. Wenn Sie anstelle des Zeichens "H" den ASCII-Code 48 eingeben, handelt es sich dabei immer noch um die beiden lesbaren Zeichen "4" und "8", die ebenfalls einen lesbaren ASCII-Code besitzen, der erst in das entsprechende Bitmuster umgewandelt werden muß.

Kurzum, was wir auch immer in lesbarer Form in den Computer ein- oder darauf ausgeben, wir benötigen auf jeden Fall eine Art Editor, der die Informationen textmäßig erfaßt bzw. wiedergibt. So spielt es letztlich keine Rolle, was für einen Text Sie mit dem Editor schreiben. Die ASCII-Codes

werden - von einem reinen Schrifttext einmal abgesehen - erst nach dem Editieren weiterverarbeitet bzw. in einen anderen Code umgewandelt.

Auch CP/M enthält einen Editor, mit dem Sie beliebige Texte erfassen können; wir werden uns nachfolgend damit beschäftigen. Allerdings ist dieser Editor, im Gegensatz zu professionellen Textverarbeitungssystemen, nicht gerade bedienerfreundlich und stammt noch aus den frühesten Zeiten von CP/M. Sollten Sie aber ein richtiges Textverarbeitungsprogramm wie etwa WordStar zur Verfügung haben, können Sie damit sämtliche Aufgaben, die sonst der CP/M-Editor übernimmt, wesentlich komfortabler ausführen. Auch WordStar ist letztlich nur ein Texteditor, der Texte erfaßt und in ASCII-Code umwandelt. Wenn Sie dieses Programm allerdings nicht zur reinen Texterfassung benutzen, sondern beispielsweise zur Assemblerprogrammierung, sollten Sie auf jeden Fall besondere Steuercodes vermeiden, die z.B. zum Randausgleich, Fettdruck usw. dienen und nur von dem jeweiligen Textverarbeitungssystem richtig interpretiert werden.

4.2 Das Dienstprogramm ED

ED ist eine COM-Datei und enthält den CP/M-Texteditor. ED muß immer im Zusammenhang mit dem Dateinamen aufgerufen werden, unter dem die zu bearbeitende Datei auf Diskette abgelegt ist bzw. abgelegt werden soll.

Um mit ED arbeiten zu können, legen wir die Systemdiskette, die die Datei ED.COM enthält, in Laufwerk A und die Diskette für die zu bearbeitende Textdatei in Laufwerk B. Dies gilt auch, wenn Sie nur ein Laufwerk zur Verfügung haben. Sie setzen es dann abwechselnd als Laufwerk A und B ein und wechseln die Disketten, wenn Sie dazu aufgefordert werden.

Wir wissen bereits, daß ED nicht gerade den komfortabelsten Texteditor darstellt und nur zur Erfassung kleinerer Texte zu empfehlen ist. Nachfolgend werden wir anhand von Beispielen schrittweise lernen, mit ED umzugehen.

4.2.1 Textdatei anlegen

Wir wollen jetzt eine Textdatei mit nur wenigen Zeilen erstellen und unter dem Namen PROBE.TXT auf Diskette ablegen. Dazu geben wir bei einem Laufwerk

ED PROBE.TXT

und bei zwei Laufwerken

ED B:PROBE.TXT

ein. Da auf der Diskette noch keine Datei unter dem Namen PROBE.TXT vorhanden ist, erscheint die Meldung:

```
NEW FILE
    : *
```

und der Cursor steht hinter dem Sternchen. Dies zeigt an, daß der Editor nun bereit ist, Befehle entgegenzunehmen.

Daraufhin geben wir den Buchstaben i ein und drücken die RETURN-Taste. ED befindet sich jetzt im Insert- oder Einfügemodus, und auf dem Bildschirm erscheint:

```
NEW FILE
    : *i
   1:
```

Bei den meisten Befehlen spielt es keine Rolle, ob sie in Groß- oder Kleinbuchstaben eingegeben werden. Beim Einfügen mit "i" müssen wir aber aufpassen, denn ein kleines "i" läßt die Eingabe von Groß- und Kleinschrift zu, während ein großes "I" sämtliche Zeichen in Großschrift umwandelt. Ähnliches gilt auch für den S-(Ersetze-)Befehl (siehe unten).

Die Ziffer 1 mit dem Doppelpunkt steht bereits für die erste Textzeile, die wir nun in Groß- und Kleinschrift eingeben können, da wir "i" klein gewählt haben. In diese Zeile schreiben wir:

Dies ist Zeile 1.

Falls Sie sich vertippen sollten, können Sie durch wiederholtes Drücken von CTRL-H bis zum fehlerhaften Zeichen zurückgehen und ab dieser Stelle den Text neu eingeben. Soll die gesamte Zeile gelöscht werden, können Sie dazu auch CTRL-X eingeben. Verwenden Sie aber auf gar keinen Fall die DEL- oder CLR-Taste, denn diese Tasten sorgen hier für unliebsame Überraschungen.

Wenn Sie die erste Zeile richtig eingegeben haben, drücken Sie die RETURN-Taste, worauf die zweite Textzeile mit der Ziffer 2 erscheint. Wir geben daraufhin den Text

Dies ist Zeile 2.

wieder auf die gleiche Weise wie zuvor ein. Genauso verfahren wir auch mit den Zeilen 3, 4 und 5, wobei der Text jeweils die Nummer der Zeile angeben soll. Haben wir die fünfte Zeile eingegeben, steht der Cursor auf der sechsten. Hier brechen wir die Eingabe ab und geben CTRL-Z ein.

Dieses Zeichen hat den ASCII-Code 26 bzw. 1AH und spielt in der Textverarbeitung eine wichtige Rolle, denn es kennzeichnet das Textende.

Unter der sechsten Zeile erscheint ein Sternchen, so daß wir einen neuen Befehl eingeben können. Durch das Drücken der E-Taste speichern wir den Text auf Diskette ab, worauf nach Beendigung das Anforderungszeichen A> erscheint. Insgesamt muß dann auf dem Bildschirm folgendes stehen:

```
NEW FILE
    : *i                    (Einfügemodus setzen)
   1: Dies ist Zeile 1.
   2: Dies ist Zeile 2.
   3: Dies ist Zeile 3.
   4: Dies ist Zeile 4.
   5: Dies ist Zeile 5.
   6:                       (Hier haben wir CTRL-Z eingegeben)
    : *E                    (Datei sichern und ED verlassen)
```

Jetzt wollen wir einmal überprüfen, ob der Text auch richtig auf Diskette geschrieben ist. Dazu verwenden wir den residenten Befehl TYPE, der eine ASCII-Datei liest und listet.

Achtung! Verwenden Sie TYPE immer nur für Text- oder ASCII-Dateien. Enthält nämlich die Datei einen anderen als den ASCII-Code, kann es zu merkwürdigen Bildschirm-Reaktionen kommen, wenn nicht darstellbare Zeichen ausgegeben werden.

Geben Sie nun

TYPE PROBE.TXT

bei einem Laufwerk und

TYPE B:PROBE.TXT

bei zwei Laufwerken ein und Sie erhalten:

```
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.
```

Tatsächlich handelt es sich dabei um den eingegebenen Text, jedoch ohne Zeilennummern, die nicht mit abgespeichert werden. Die einzelnen Zeilen befinden sich nämlich kontinuierlich oder sequentiell auf der Diskette, wobei an jedem Zeilenende ein Carriage-Return- und ein Line-Feed-Zeichen ausgegeben wird.

Spaßeshalber wollen wir uns jetzt einmal das Directory ansehen und geben dazu

*DIR PROBE.**

bzw.

*DIR B:PROBE.**

ein. Und siehe da, die Textdatei ist nicht nur einmal, sondern sogar doppelt gespeichert, denn wir erhalten:

```
A: PROBE    BAK : PROBE    TXT
```

ED legt nämlich automatisch eine Sicherheits- oder Backup-Kopie mit gleichem Dateinamen unter der Extension BAK an. Die BAK-Datei ist zwar in unserem Fall noch leer, da wir den Text gerade neu angelegt haben. Wenn wir die Datei aber später ändern oder fortschreiben, wird die alte Originaldatei zur BAK-Datei umbenannt, während die neue Datei die richtige Extension erhält. Auf diese Weise können wir immer noch auf eine Sicherheitskopie mit dem letzten Bearbeitungsstand zurückgreifen, falls der neu bearbeiteten Datei etwas zustoßen sollte.

4.2.2 Zeilen einfügen/löschen

Jetzt wollen wir daran gehen, unsere Textdatei PROBE.TXT zu verändern, indem wir Zeilen einfügen und löschen. Wenn wir dies beherrschen, besitzen wir schon einen guten Grundstock, beliebige Texte mit ED zu bearbeiten.

Um diesen Vorgang zu verstehen, müssen wir wissen, daß ED intern einen Textpuffer anlegt, der den zu bearbeitenden Text aufnimmt. Die Größe dieses Puffers ist jedoch begrenzt, so daß bei umfangreichen Dateien immer nur ein Teil in ihm abgelegt werden kann. Da es für solche Dateien bessere Editiermöglichkeiten als mit ED gibt und unser Beispieltext nur wenige Zeilen umfaßt, gehen wir im Augenblick davon aus, daß der gesamte Text aus der Originaldatei in den Puffer geladen wird.

Während Sie mit ED arbeiten, wird eine neue Datei auf der Diskette angelegt, die man als Zwischendatei bezeichnet und die mit der Extension $$$ *gekennzeichnet ist. Die Originaldatei bleibt dabei völlig unberührt und* wird nach Abschluß der Textverarbeitung in eine BAK-Datei umbenannt, während die Zwischendatei den eigentlichen Dateinamen erhält.

Auf unsere Datei PROBE.TXT bezogen bedeutet dies, daß während der Bearbeitung eine Zwischendatei PROBE.$$$ angelegt wird, in die der

überarbeitete Text geschrieben wird. Wenn Sie dann die Bearbeitung abschließen, wird die ursprüngliche Datei in PROBE.BAK und die Zwischendatei in PROBE.TXT umbenannt.

Nachdem wir unsere Textdatei bereits auf Diskette mit dem E-Befehl gesichert haben, müssen wir ED erneut starten, um sie zu verändern. Geben Sie deshalb wieder

ED PROBE.TXT

bzw.

ED B:PROBE.TXT

ein, um ED zu laden und zu aktivieren. Daraufhin sucht der Editor das Directory nach dem angegebenen Dateinamen ab. In diesem Fall ist er bereits vorhanden, weshalb nun nicht mehr der Hinweis NEW FILE, sondern nur noch

```
: *
```

erscheint und wir jetzt mit der eigentlichen Bearbeitung beginnen können. Doch zuvor dürfen wir nicht vergessen, die Textdatei in den Puffer zu laden, wozu wir

#A

eingeben. Das Laufwerk beginnt sich noch einmal zu drehen und lädt den Text ein. Wenn wir jetzt

B#T

eingeben, werden unsere fünf Textzeilen wieder aufgelistet:

```
 : *#A                    (Text in Puffer lesen)
1: *B#T                   (Text von Beginn an ausgeben)
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
5: Dies ist Zeile 5.
1:*
```

Statt des Ziffernkreuzes (#) könnten wir auch einen Zahlenwert setzen:

3A

würde nur drei Zeilen in den Textpuffer laden und

B2T

nur die ersten beiden Zeilen im Puffer listen. Übrigens wird mit

B

der interne Pufferzeiger auf den Pufferanfang und mit

-B

wieder auf das Pufferende gesetzt. Somit bedeutet

B#T

"Setze den Zeiger an den Pufferanfang und gib sämtliche Zeilen aus". Dabei wird die Textausgabe mit "T" gesteuert. "B" und "#T" hätten wir allerdings auch getrennt erteilen können, aber ED erlaubt uns, mehrere Befehle in einer Zeile hintereinander und ohne Zwischenraum anzugeben.

Den Pufferzeiger können wir auf jede beliebige Zeile setzen. Wenn wir z.B.

3:

eingeben, steht er am Anfang der dritten Zeile. Geben wir daraufhin

2T

ein, werden von dort aus zwei Zeilen ausgegeben und es erscheint:

```
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
```

Beide Befehle hätten wir auch wieder gemeinsam angeben können und damit dasselbe Resultat erreicht:

3:2T

Erhält "T" einen negativen Zahlenwert, also beispielsweise

-2T

erscheinen die beiden Zeilen über dem Zeiger:

```
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
```

Fassen wir nochmals zusammen:

nA liest n Zeilen von Diskette in den Textpuffer

+/-B setzt den Textpufferzeiger an den Textanfang bzw. das Textende

n: setzt den Pufferzeiger an den Anfang der n-ten Zeile

+/-nT listet n Zeilen ab bzw. vor dem Pufferzeiger

steht anstelle von n und bedeutet "sämtliche Zeilen"

+/-nL bewegt den Pufferzeiger von seiner gegenwärtigen Position n Zeilen auf- bzw. abwärts

Jetzt wollen wir zwischen Zeile 2 und 3 zwei weitere Zeilen einfügen. Dazu müssen wir den Pufferzeiger auf die dritte Zeile setzen, denn die neuen Zeilen erscheinen ja vor dieser Zeile, weshalb sämtliche Zeilen ab der dritten Zeile beim Einfügen nach unten verschoben werden:

```
1: *3:                          (Pufferzeiger auf dritte Zeile)
3: *i                           (Einfügemodus einschalten)
3: Dies ist die neue Zeile 2a.
4: Dies ist die neue Zeile 2b.
5:                              (hier CTRL-Z eingeben)
5: *
```

Nachdem nun der Pufferzeiger am Anfang der dritten Zeile steht, muß mit "i" wieder der Einfügemodus eingeschaltet werden. Wir geben die beiden neuen Zeilen wie gewohnt ein und drücken abschließend CTRL-Z.

Nun kontrollieren wir, ob die neuen Zeilen auch richtig eingefügt sind:

```
1: *B#T                         (Zeiger an Textanfang und alle Zeilen listen)
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist die neue Zeile 2a.
4: Dies ist die neue Zeile 2b.
5: Dies ist Zeile 3.
6: Dies ist Zeile 4.
7: Dies ist Zeile 5.
1: E                            (Datei auf Diskette sichern)
```

Da die neuen Zeilen an der richtigen Stelle stehen, haben wir auch gleich den Text mit "E" auf Diskette gesichert und den Editor verlassen. Wir können jetzt, wenn es uns interessiert, die geänderte Datei mit TYPE ansehen, wie wir es nach ihrer Erstellung bereits getan haben.

Nachdem uns nun das Einfügen der beiden Zeilen gelungen ist, wollen wir sie zu Übungszwecken gleich wieder löschen. Dazu laden wir den Editor erneut unter Angabe des Dateinamens PROBE.TXT. Diesen Vorgang könnten wir uns aber ersparen, wenn wir den Editor vorher nicht mit "E" verlassen hätten. Dann wäre auch der Text noch im Puffer, den wir in unserem Fall mit "#A" erst wieder neu einladen müssen.

Wir setzen den Pufferzeiger auf die dritte Zeile, die nun die erste Zeile darstellt, die es zu löschen gilt. Mit

2K

werden die beiden Zeilen gelöscht, wobei "K" (englisch: Kill) für Löschen und die Ziffer 2 für die beiden Zeilen steht. Abschließend listen wir den Text nochmals auf. Hier nun alles, was sich auf dem Bildschirm abspielt:

```
 : *#A                  (Text in Puffer lesen)
1: *3:                  (Pufferzeiger auf 3. Zeile)
3: *2K                  (von da ab 2 Zeilen löschen)
3: B#T                  (Text auflisten)
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
5: Dies ist Zeile 5.
1: *
```

4.2.3 Text korrigieren

Im letzten Abschnitt haben wir gelernt, Textzeilen einzufügen und zu löschen. Für viele Korrekturen wird es die einzige Möglichkeit bleiben, die gesamte fehlerhafte Zeile zu löschen und im Einfügemodus wieder neu zu schreiben und in den Text einzufügen. Wesentlich komfortabler wäre es allerdings, eine oder mehrere Zeilen aufzulisten, mit dem Cursor an die fehlerhafte Stelle vorzurücken und diese dann auszubessern.

Leider müssen wir bei ED auf diese Möglichkeit verzichten, weshalb sie nur Textverarbeitungsprogrammen wie WordStar vorbehalten bleibt. Den einzigen Luxus, den ED bietet, ist das Suchen und Austauschen von Wörtern im Text.

Haben wir nun in einer Textzeile einen Fehler festgestellt, müssen wir ED das fehlerhafte Wort erst suchen lassen, worauf es gelöscht und durch ein neues ersetzt wird. Dies ist wirklich eine sehr umständliche Methode, Texte zu korrigieren, aber einfacher geht es leider nicht, es sei denn, man ersetzt die gesamte Zeile.

Diese Art der "Korrektur" wollen wir nachfolgend anhand eines Beispiels ausprobieren. Laden Sie dazu erneut den Editor unter Angabe der Datei PROBE.TXT und lesen Sie diese wieder mit "#A" in den Textpuffer. Wenn Sie jetzt

2:T (Zeiger auf Zeile 2 und diese listen)

eingeben, erhalten Sie den Text, den wir ursprünglich eingegeben haben:

```
2: Dies ist Zeile 2.
```

Jetzt wollen wir das Wort "Dies" durch "Hier" ersetzen, was folgendermaßen geschieht:

**sDies^ZHier^Z0LT*

Daraufhin erscheint die korrigierte Zeile:

```
2: Hier ist Zeile 2.
```

Lassen Sie uns die Befehlszeile einmal etwas genauer betrachten. Hinter dem Sternchen, das ja ohnehin zur Befehlseingabe auffordert, steht ein kleines "s". Unter "s" versteht ED "Ersetze" (englisch: substitute). Bei den beiden nachfolgenden Wörtern handelt es sich zuerst um das alte und schließlich um das neue Wort, wobei beide Wörter mit ^Z (CTRL-Z) abschließen müssen. Achten Sie auch hier wieder darauf, daß wir ein kleines "s" angeben. Bei einem Großbuchstaben würde, ähnlich wie bei "I" (Einfügen), nicht "Hier", sondern "HIER" im korrigierten Text erscheinen, selbst wenn wir Kleinbuchstaben eingeben.

Mit 0LT (erstes Zeichen eine Null, kein Buchstabe "O"!) erhält ED schließlich die Anweisung, den Pufferzeiger zurück an den Zeilenanfang zu setzen und die Zeile dann auszugeben. Somit haben wir gleichzeitig die Möglichkeit, festzustellen, ob die Korrektur richtig stattgefunden hat.

ED besitzt noch einige weitere Befehle als diejenigen, die wir bisher kennengelernt haben. Sie sind in der Zusammenfassung aller CP/M-Befehle in Kapitel 10 aufgeführt. Auf ihre ausführliche Behandlung wollen wir verzichten, da wir sonst an dieser Stelle mehr Verwirrung als Klarheit schaffen würden. Mit den hier beschriebenen Befehlen sollten Sie aber in der Lage sein, einfache Texte einzugeben und zu korrigieren, sofern Sie nicht ohnehin schon auf andere Textverarbeitungsprogramme zurückgreifen.

Übrigens kann die Datei PROBE.TXT, die wir mit ED auf Diskette erzeugt haben, auch von BASIC aus ganz gewöhnlich als sequentielle Datei gelesen werden. Wir können sogar so weit gehen, daß wir ein ganzes BASIC-Programm mit ED oder einem anderen Textverarbeitungsprogramm schreiben, es als BAS-Datei auf Diskette ablegen, um es dann von BASIC aus auszuführen. Eine so erzeugte Datei entspricht dann genau einem Programm, das in BASIC mit SAVE "Name",A abgespeichert wurde. Umgekehrt kann natürlich auch ein BASIC-Programmlisting in eine Textdatei übernommen werden, wenn es sich im ASCII-Code auf der Diskette befindet.

5 PIP - Ein universelles Kopierprogramm

5.1 Allgemeines

PIP, englisch "Peripherial Interchange Processor", ist ein weiteres Standardprogramm für jedes CP/M-System. Um ihm eine allgemeine Kurzbezeichnung zu geben, haben wir es in der Überschrift als universelles Kopierprogramm bezeichnet. Dieser Begriff ist zwar durchaus zutreffend, beschreibt aber noch nicht sämtliche Möglichkeiten, die uns PIP bietet.

Mit PIP können Sie Dateien aller Art kopieren. Dabei spielt es keine Rolle, ob es sich um reine CP/M-COM-Dateien, Textdateien oder BASIC-Programme handelt. Sie benutzen nämlich alle das gleiche Disketten-Aufzeichnungsverfahren, was für den Kopiervorgang an sich keine Rolle spielt. Lediglich beim Aneinanderhängen von Dateien sind einige Regeln zu beachten, die wir aber im Verlauf dieses Kapitels noch kennenlernen werden.

Zwei Dinge kann PIP jedoch nicht, nämlich Disketten formatieren und bei Verwendung von nur einem Laufwerk eine Datei auf eine andere Diskette kopieren. Zum Formatieren gibt es ja bekanntlich das Programm DISCKIT3 (JOYCE DISCKIT), das außerdem ganze Disketten kopieren und verifizieren kann. Was das Kopieren mit einem Laufwerk betrifft, so ist dies strenggenommen zwar richtig und war für CP/M 2.2 durchaus auch zutreffend. Statt dessen verwendete man hierfür spezielle Kopierprogramme, wie z.B. FILECOPY, das auf der CP/M-Systemdiskette für den CPC 464 und 664 enthalten ist.

Glücklicherweise unterstützt CP/M Plus aber ein zweites Laufwerk B, selbst wenn nur ein Laufwerk angeschlossen ist. Wird nun PIP, oder ein anderes CP/M-Programm, ausgeführt, das auf dieses Laufwerk zugreift, wird der Benutzer aufgefordert, die Disketten zu wechseln, wenn jeweils das andere Laufwerk angesprochen wird. Diese Methode ist zwar nicht sehr komfortabel, aber immer noch besser, als wenn man solche Programme überhaupt nicht einsetzen könnte. Dabei ist jedoch zu beachten, daß beim Wechsel immer die richtige Diskette einzulegen ist, was ansonsten leicht zu Fehlern führen kann.

Der JOYCE bietet darüber hinaus noch eine weitere Möglichkeit, denn er enthält ein virtuelles Laufwerk M, das nur aus RAM-Speicher besteht, aber wie ein echtes Laufwerk zu behandeln ist. Möchte man beispielsweise mehrere Dateien von einer Diskette auf eine andere kopieren, so kann man sie zunächst im Laufwerk M ablegen und von dort aus auf die Zieldiskette

übertragen. Dies erspart einen häufigen Diskettenwechsel, der beim Kopieren mit nur einem Laufwerk von "A" nach "B" nötig wäre.

PIP wurde zwar in erster Linie zum Kopieren und zur übertragen von Textdateien im ASCII-Code entwickelt, es kann aber durchaus auch mit anderen Dateien, die einen beliebigen Binärcode enthalten, arbeiten, was die universellen Einsatzmöglichkeiten steigert. Wir werden in diesem Kapitel auch eine Vielzahl von Optionen kennenlernen, die beim Kopieren berücksichtigt werden und eine Datei während der übertragung zusätzlich verändern. So kann z.B. eine Textdatei in Groß- oder Kleinbuchstaben umgeformt werden oder bestimmte Steuerzeichen für den Ausdruck erhalten.

Aber nicht nur eine Übertragung von Diskettendateien ist mit PIP möglich, sondern auch der Datentransfer zwischen externen Geräten und Diskette bzw. umgekehrt. Auch diese Möglichkeit werden wir hier noch näher kennenlernen.

5.2 Laden und Aktivieren von PIP

PIP ist eine COM-Datei und auf der Systemdiskette abgelegt. Wenn wir nun mit diesem Programm arbeiten wollen, gibt es zwei Möglichkeiten. Entweder wir starten es und geben gleichzeitig Anweisungen, was wir kopieren bzw. übertragen wollen. Allgemein ausgedrückt schreiben wir dafür:

PIP (Befehlsfolge) <RETURN>

PIP führt nun sämtliche Befehle aus und führt anschließend einen Warmstart durch, ähnlich wie es auch bei STAT der Fall ist. Diese Form des Aufrufs hat jedoch den Nachteil, daß PIP während des Kopiervorgangs immer auf der Diskette in Laufwerk A oder B abgelegt sein muß, was in manchen Fällen sicher sehr lästig ist.

Wir können PIP aber auch ohne Befehlsfolge einsetzen. Nach Eingabe von:

PIP <RETURN>

erscheint auf dem Bildschirm ein

*

PIP ist jetzt geladen und aktiviert und befindet sich im Befehlsmodus, was wir an dem ausgegebenen Sternchen erkennen. Wir können nun die Diskette, die PIP enthält, aus dem Laufwerk herausnehmen und statt dessen die-

jenige einlegen, auf die bzw. von der kopiert werden soll. Dabei geben wir dieselbe Befehlsfolge wie im ersten Fall ein.

Nach Ausführung der Befehle kehrt PIP wieder in den Befehlsmodus zurück und ein weiteres Sternchen erscheint auf dem Bildschirm. Wir können nun auf die gleiche Weise weitere Befehle erteilen, die dann jeweils ausgeführt werden. Diesen Vorgang können wir beliebig oft wiederholen, bis wir unsere Arbeit mit PIP beendet haben. Dann drücken wir die RETURN-Taste, worauf PIP verlassen und ein Warmstart durchgeführt wird.

Gerade für die Anwender, die mit einem Laufwerk auskommen müssen, dürfte die letzte Methode, bei der PIP nicht erst selbst kopiert werden muß, wesentlich angenehmer sein.

5.3 Kopieren von Diskette zu Diskette

In diesem Abschnitt wollen wir den einfachsten und naheliegendsten Einsatz von PIP betrachten, nämlich das Kopieren von Dateien auf die gleiche oder eine andere Diskette. Zum Übertrag auf eine andere Diskette ist jedoch ein zweites Laufwerk B erforderlich.

Die allgemeine Kopieranweisung lautet hier:

PIP (Laufwerk:)Zieldatei=(Laufwerk:)Quelldatei

bzw.

*(Laufwerk:)Zieldatei=(Laufwerk:)Quelldatei

Im ersten Fall wird PIP aufgerufen, wobei gleichzeitig der Kopiervorgang in die Wege geleitet wird, während im zweiten Fall PIP bereits geladen und aktiviert ist, so daß wir nur noch die Befehlszeile eingeben müssen.

Das Laufwerk muß nur angegeben werden, wenn mit einem anderen als dem Bezugslaufwerk gearbeitet wird. Haben Sie ohnehin nur ein Laufwerk zur Verfügung, können Sie auf diese Angabe verzichten.

Eine Quelldatei ist immer diejenige Datei, die kopiert werden soll, während die Zieldatei der kopierten Datei entspricht.

Als Anwendungsbeispiel nehmen wir einmal an, Sie möchten von der Datei MUSTER.TXT eine Backup-Kopie mit der Extension BAK auf der gleichen Diskette herstellen, da dies anderweitig noch nicht geschehen ist. Wir gehen davon aus, daß nur ein Laufwerk angeschlossen ist.

Angenommen, PIP befindet sich auf der gleichen Diskette wie MUSTER.TXT, dann können wir die Befehlsfolge gleich mit angeben:

PIP MUSTER.BAK=MUSTER.TXT <RETURN>

PIP wird geladen, worauf sich unmittelbar der Kopiervorgang anschließt.

Enthält dagegen die Diskette nicht das PIP-Programm, legen wir zunächst die Systemdiskette ein und rufen PIP ohne Befehlsfolge auf:

PIP <RETURN>

Nachdem das Sternchen

*

erschienen ist, legen wir die Diskette mit MUSTER.TXT ein und schreiben erst dann dahinter die Befehlszeile:

**MUSTER.BAK=MUSTER.TXT <RETURN>*

Dabei dürfen Sie natürlich nicht das Sternchen mit eintippen, denn es wird ja ohnehin ausgegeben.

Ist der Kopiervorgang beendet, erscheint in der nächsten Zeile ein weiteres Sternchen:

*

Wenn Sie jetzt mit PIP weiter arbeiten wollen, geben Sie den nächsten Befehl ein, wenn nicht, drücken Sie die RETURN-Taste, worauf Sie PIP wieder verlassen.

Bisher haben wir diese ausführliche Darstellungsform gewählt, damit Sie lernen, mit PIP umzugehen. Künftig geben wir nur noch die Befehle als solche an, also ohne PIP und ohne Sternchen, denn Sie wissen ja jetzt, wie PIP zu starten ist.

Das nächste Beispiel arbeitet mit zwei Laufwerken. Wir übertragen jetzt die Datei MUSTER.TXT von Laufwerk A nach Laufwerk B, wo wir sie unter dem gleichen Namen ablegen. Hierzu lautet der Befehl:

B:MUSTER.TXT=A:MUSTER.TXT

Dieses Mal muß allerdings das Laufwerk angegeben werden. Selbstverständlich hätten wir die Datei auch unter einem anderen Namen in Lauf-

werk B ablegen können. Hier haben wir aber den gleichen Namen beibehalten und könnten deshalb die Befehlszeile auch kürzer schreiben:

B:=A:MUSTER.TXT

Ist nämlich für das Ziellaufwerk kein Name angegeben, wird automatisch der ursprüngliche Dateiname übernommen.

Auch PIP läßt mehrdeutige Dateinamen zu, so daß wir ganze Dateigruppen mit einem Befehl kopieren können. Hier gelten die gleichen Regeln, die wir schon in Kapitel 1 im Zusammenhang mit dem DIR-Befehl kennengelernt haben. Auch dazu zwei Beispiele:

A:=B:.COM*

kopiert sämtliche COM-Dateien von Laufwerk B nach Laufwerk A und

B:=A:.**

kopiert sämtliche Dateien, die sich auf Laufwerk A befinden, nach Laufwerk B.

Indem Sie PIP auf diese Weise einsetzen, können Sie, sofern Sie zwei Laufwerke angeschlossen haben, durchaus auf das Programm FILECOPY verzichten. FILECOPY kann nämlich nur mit einem Laufwerk arbeiten, wobei unter Umständen die Diskette mehrfach gewechselt werden muß, was bei PIP entfällt.

5.4 Dateien aneinanderhängen

PIP bietet die Möglichkeit, mehrere Dateien zu einer neuen Datei zu vereinigen. Wir müssen uns dabei allerdings überlegen, in welchen Fällen eine solche Maßnahme auch tatsächlich sinnvoll ist.

Wenn Sie z.B. Maschinenprogramme, wie es die COM-Dateien in der Regel sind, miteinander verknüpfen, ist dies sicher nicht der Fall. Jedes Maschinenprogramm (Objektcode) enthält nämlich im allgemeinen viele absolute Adressen, die sich auf den Speicherbereich beziehen, in dem es abgelegt und abgearbeitet wird. Durch die Verknüpfung verschieben sich diese Adressen mit Sicherheit, so daß das Programm gar nicht mehr ausführbar ist und der Rechner wahrscheinlich abstürzt.

Vielleicht wäre ein Aneinanderreihen von BASIC-Programemen, die im internen BASIC-Code gespeichert sind, bedingt sinnvoll, wenn die einzel-

nen BASIC-Zeilen anschließend neu gelinkt (verbunden) werden. Aber dafür gibt es andere und bessere Möglichkeiten.

Wirklich empfehlenswert und nützlich ist diese Methode nämlich nur für Textdateien im ASCII-Code. Aus Kapitel 4,in dem wir uns mit dem Editor beschäftigt haben, wissen wir, daß alle Informationen, die der Computer aufnimmt, letztlich nur in Form von ASCII-Zeichen darstellbar sind. So können Sie ein BASIC-Programm als ASCII-Datei speichern, die dann wieder als ganz gewöhnlicher Text gelesen werden kann. Auch Maschinenprogramme werden meist zuerst im Assembler-Quellcode geschrieben, der noch unabhängig vom Speicherbereich ist und ebenfalls einen Text aus ASCII-Zeichen darstellt. Ganz zu schweigen von "normalen" Texten, zu denen auch die Zeilen gehören, die Sie gerade lesen. Sie sind zwar der Rechtschreibung der deutschen Sprache, aber ansonsten keiner besonderen Syntax zum Zweck einer weiteren Umwandlung unterlegen. Alle derartigen Texte können Sie beliebig kombinieren und aneinanderhängen.

Reine Textdateien, die nur aus ASCII-Zeichen bestehen, sollten immer mit dem ASCII-Code 1AH bzw. CTRL-Z abschließen, was normalerweise automatisch von den Texteditoren durchgeführt wird. Beim Lesen des Textes wird dann immer dieses Zeichen abgefragt, um das Textende festzustellen, denn der letzte Record oder Block dieser Datei ist meistens nur zum Teil beschrieben und enthält noch freien Speicherplatz.

Wenn PIP nur eine Datei kopiert, fragt es den Code 1AH nicht ab, denn es überträgt immer den ganzen letzten Block, selbst wenn dieser nicht vollständig belegt ist. Deshalb können wir beispielsweise auch COM-Dateien korrekt übertragen. Anders ist es, wenn Dateien verknüpft werden sollen, da in diesem Fall das genaue Ende festgestellt werden muß. Aus diesem Grund verbindet PIP nur Dateien, von denen es annimmt, daß sie ASCII-Code enthalten. Befindet sich dagegen in einem Maschinen- oder BASIC-Programm an irgendeiner Stelle der Code 1AH, was durchaus wahrscheinlich ist, wird die Übertragung an dieser Stelle abgebrochen, da PIP fälschlicherweise annimmt, eine Textdatei sei zu Ende.

Wenn wir nun Textdateien miteinander verbinden, müssen wir eine Zieldatei, aber mehrere, durch Komma getrennte Quelldateien in der Befehlszeile angeben. Vielleicht haben Sie noch die Textdatei PROBE.TXT, die wir mit dem Editor in Kapitel 4 erstellt haben, auf Ihrer Diskette. Diese Datei wollen wir nun dreimal aneinanderhängen und daraus eine neue Datei mit dem Namen PROBENEU.TXT erzeugen. Wir geben dazu folgendes nach dem Sternchen ein:

PROBENEU.TXT=PROBE.TXT,PROBE.TXT,PROBE.TXT

Befinden sich die Zieldatei und die zu verknüpfenden Dateien in verschiedenen Laufwerken oder aber allesamt in Laufwerk B, müssen wir vor jedem Dateinamen noch die Laufwerksangabe setzen, wie z.B.:

A:PROBENEU.TXT=B:PROBE.TXT,B:PROBE.TXT,B:PROBE.TXT

oder wenn verschiedene Quelldateien verwendet werden,

A:NEUDATEI.TXT=A:DATEI.BAK,B:BRIEF.TXT,B:LAGER.DAT

Hier befindet sich die Zieldatei NEUDATEI.TXT und die Quelldatei DATEI.BAK in Laufwerk A, während die anderen beiden Quelldateien BRIEF.TXT und LAGER.DAT in Laufwerk B abgelegt sind.

5.5 Datenaustausch mit anderen Peripheriegeräten

PIP kann Dateien nicht nur von Diskette zu Diskette übertragen, sondern auch von Diskette auf Peripheriegeräte und umgekehrt.

Lassen Sie uns den Begriff Peripherie einmal etwas genauer betrachten. Darunter fällt jedes externe Gerät, das an die Zentraleinheit angeschlossen ist, also neben der Floppy, ohne die ja CP/M gar nicht betrieben werden kann, auch Drucker, Bildschirm und Tastatur (Terminal) sowie andere Zusatzgeräte wie beispielsweise Modem (einschließlich serieller Schnittstelle) oder Harddisc. Für solche Geräte sind jedoch meist besondere Änderungen im BIOS erforderlich, die z.B. für ein Modem das mitgelieferte Terminal-Programm vornimmt.

Auf all diese Geräte kann PIP Dateien ausgeben. In einigen Fällen, z.B. über ein Modem, können Dateien auch eingelesen und auf Diskette gespeichert werden. Dabei handelt es sich zumeist um reine ASCII-Dateien, besonders wenn Text auf dem Drucker oder dem Bildschirm ausgegeben wird.

Obwohl Ihr CPC ohnehin standardmäßig bereits mit Bildschirm, Tastatur und eventuell Drucker ausgestattet ist, betrachtet PIP diese Geräte als externe Peripherie.

Um Peripheriegeräte anzusteuern, benutzt PIP bestimmte Bezeichnungen für die verschiedenen Ein-/Ausgabekanäle, die sich aber von denen des DEVICE-Befehls (s.Kapitel 3) unterscheiden. Im einzelnen handelt es sich dabei um:

CON:	Konsole (Eingabe Tastatur, Ausgabe Bildschirm)
LST:	Listeinheit (meist zur Ansteuerung des Druckers)
AUX:	vom Benutzer definierte Ein-/Ausgabeeinheit (z.B. Modem)

Nun können wir auch PIP-Befehle angeben, die anstelle einer Ein-/Ausgabedatei einen dieser Kanäle angibt. Den wohl häufigsten Anwendungsfall stellt die Ausgabe von Textdateien auf den Drucker oder auf den Bildschirm dar.

Sie werden sich jetzt vielleicht sagen, daß dies auch mit dem TYPE-Befehl erreicht werden kann, was durchaus richtig ist. Mit PIP stehen uns aber noch weitere Variationsmöglichkeiten zur Verfügung, die wir in diesem und im nächsten Abschnitt kennenlernen werden.

Grundsätzlich gilt auch bei der Einbeziehung von externen Geräten die Befehlsfolge

Zielgerät = (Laufwerk:) Quelldatei

bzw.

(Laufwerk:) Zieldatei = Quellgerät

d.h. es ist immer zuerst die Zieleinheit und dann, durch ein Gleichheitszeichen getrennt, die Quelleinheit anzugeben.

Jetzt wollen wir einmal unsere Beispieldatei PROBE.TXT auf dem Bildschirm auflisten, wozu wir nach dem Sternchen folgendes eingeben:

CON:=PROBE.TXT

bzw.

CON:=B:PROBE.TXT

Dabei gilt der letzte Befehl, wenn die Datei von Laufwerk B, das nicht Bezugslaufwerk ist, gelesen werden soll. Hierin ist CON: (=Bildschhirm) das Zielgerät, das anstelle einer Zieldatei aufgeführt ist.

Dasselbe wollen wir jetzt auch mit dem Drucker ausprobieren, wobei die Anweisungen folgendermaßen lauten:

LST:=PROBE.TXT

bzw.

LST:=B:PROBE.TXT

Als weiteres können wir hier auch die Datei PROBE.TXT dreimal hintereinander auf dem Drucker ausgeben:

LST:=PROBE.TXT,PROBE.TXT,PROBE.TXT

Falls wir über den TTY-Kanal eine serielle Schnittstelle angeschlossen haben, können wir mit

AUX:=MODEM.TXT

die Datei MODEM.TXT über die Schnittstelle ausgeben bzw. sie mit

MODEM.TXT=AUX:

über dieselbe einlesen.

Zum Abschluß betrachten wir noch ein interessantes Beispiel, bei dem wir direkt über die Tastatur in eine Datei schreiben. Versuchen Sie einmal folgendes:

TERMINAL.TXT=CON:

Wenn jetzt der Cursor am Anfang der nächsten Zeile erscheint, tippen Sie einen kurzen Text ein, wie z.B.:

Dies ist die Datei TERMINAL.TXT. <CTRL-Z>

Achten Sie unbedingt darauf, daß Sie am Schluß CTRL-Z drücken, wodurch das Dateiende bei Textdateien gekennzeichnet wird. Die Datei wird dann geschlossen und PIP wartet auf den nächsten Befehl bzw. das Bereitschaftszeichen A> erscheint wieder auf dem Bildschirm.

Denken Sie aber nicht, daß diese Methode zur Textverarbeitung geeignet ist und das Programm ED oder einen sonstigen Texteditor überflüssig macht. Sie haben hier nämlich keinerlei Möglichkeit, den eingegebenen Text zu korrigieren, da er direkt in die Datei geschrieben wird. Außerdem müssen Sie an jedem Zeilenende die Zeichen Carriage-Return und Line-Feed von Hand eingeben, was mit der ENTER-Taste bzw. CTRL-M und CTRL-J geschieht.

5.6 Spezielle Befehle

Wir haben nun die wichtigsten PIP-Befehle kennengelernt, die für den Alltagsgebrauch eigentlich ausreichen müßten. Es gibt aber noch eine Reihe von Spezialbefehlen und Optionen, von denen einige nachfolgend vorgestellt werden sollen. In der Zusammenfassung in Kapitel 10 sind dann sämtliche Befehle noch einmal aufgeführt.

5.6.1 Dateiausschnitte

Mit PIP könnnen Sie auch Teile einer Textdatei übertragen. Dies funktioniert ähnlich, als wenn Sie einen mit ED erstellten Text korrigieren möchten, denn Sie müssen ein Suchkriterium vorgeben. Die Datei wird dann von bzw. ab der Stelle, an der der gesuchte Begriff auftritt, übertragen.

Als Beispiel verwenden wir wieder unsere Datei PROBE.TXT, die wir mit ED erstellt haben. In ihrer ungekürzten Form hat sie folgenden Inhalt:

```
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.
```

Jetzt wollen wir diese Datei ab der Stelle auf dem Bildschirm ausgeben, an der der Begriff "Zeile 3." auftritt. Da dieser Begriff Groß- und Kleinbuchstaben enthält, müssen wir zunächst einmal PIP separat starten und dann die Befehlszeile eingeben. Würden wir ihn dagegen vom CCP aus in der Form von PIP (Befehlszeile) suchen, würden sämtliche Zeichen des Begriffs automatisch in Großbuchstaben umgewandelt, wodurch er natürlich nicht aufgefunden werden kann.

Starten Sie nun also PIP, falls Sie es noch nicht getan haben, und geben Sie, wenn das Sternchen erscheint, folgendes ein:

CON:=PROBE.TXT[SZeile 3.^Z]

Nun erscheint:

```
Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.
```

Im nächsten Beispiel geben wir die gleiche Datei aus, allerdings nur bis zu der Stelle, an der "Zeile 3." auftritt:

CON:=PROBE.TXT[QZeile 3.^Z]

Diesmal erhalten wir:

```
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
```

Selbstverständlich könnten wir hier statt des Bildschirms auch eine Diskettendatei als Ziel angeben oder warten, bis ein bestimmter Begriff über ein Modem eingeht. Entscheidend ist jedoch nur der Inhalt der eckigen Klammer, den wir uns einmal genauer ansehen wollen.

"S" bedeutet, daß die Übertragung erst dann beginnt, wenn der gesuchte Begriff auftritt; bei "Q" dagegen findet die Übertragung ab dem Dateianfang statt und wird an dieser Stelle abgebrochen. Der gesuchte Begriff wird allerdings in beiden Fällen mit übertragen, wie wir gesehen haben.

Hinter "S" bzw. "Q" steht der gesuchte Begriff, der mit CTRL-Z (^Z) abgeschlossen sein muß. Fassen wir diese beiden Optionen noch einmal in allgemeiner Form zusammen:

[S"Begriff"^Z]	Übertragung beginnt bei "Begriff"
[Q"Begriff"^Z]	Übertragung bricht bei "Begriff" ab

5.6.2 Groß- und Kleinschrift

Die folgenden beiden Optionen ermöglichen eine Übertragung von Textdateien, wobei sämtliche Buchstaben entweder in Groß- oder in Kleinschrift umgewandelt werden. Dabei bedeutet

[U]	Umwandlung in Großschrift und
[L]	Umwandlung in Kleinschrift

Wir wollen dies wieder mit unserer Datei PROBE.TXT ausprobieren und geben dazu

CON:=PROBE.TXT[U]

ein. Dabei erscheinen nur große Buchstaben auf dem Bildschirm:

```
DIES IST ZEILE 1.
DIES IST ZEILE 2.
DIES IST ZEILE 3.
DIES IST ZEILE 4.
DIES IST ZEILE 5.
```

Mit

CON:=PROBE.TXT[L]

dagegen werden alle Großbuchstaben in kleine umgeformt:

```
dies ist zeile 1.
dies ist zeile 2.
dies ist zeile 3.
dies ist zeile 4.
dies ist zeile 5.
```

5.6.3 Textausgabe mit Zeilennummern

Wir haben die Möglichkeit, einen Text so zu übertragen, daß am Anfang jeder Zeile eine Zeilennummer steht. Strenggenommen wird die Zeilennummer immer dann eingefügt, wenn im Text ein Carriage-Return- und Linefeed-Zeichen auftritt. Die benötigten Optionen lauten hier:

[N] für fortlaufende Zeilennummern und

[N2] für fortlaufende Zeilennummern (sechsstellig mit führenden Nullen)

Auf unsere Datei PROBE.TXT bezogen erscheint infolge von

CON:=PROBE.TXT[N]

folgende Bildschirmausgabe:

```
1: Dies ist Zeile 1.
2: Dies ist Zeile 2.
3: Dies ist Zeile 3.
4: Dies ist Zeile 4.
5: Dies ist Zeile 5.
```

Geben wir dagegen

CON:=PROBE.TXT[N])

vor, erhalten wir führende Nullen:

```
000001  Dies ist Zeile 1.
000002  Dies ist Zeile 2.
000003  Dies ist Zeile 3.
000004  Dies ist Zeile 4.
000005  Dies ist Zeile 5.
```

Wir können auch mehrere Optionen gleichzeitig in eckiger Klammer angeben, wie z.B.:

CON:=PROBE.TXT[N2U]

Hier wird die Datei PROBE.TXT in Großbuchstaben und mit Zeilennummern einschließlich führender Nullen auf dem Bildschirm ausgegeben.

5.6.4 Verify

Sie können, wenn Sie Dateien von Diskette zu Diskette übertragen, mit der V-Option gleichzeitig ein Verify durchführen. Dabei wird die Zieldatei normal auf Diskette geschrieben und anschließend überprüft, ob die Übertragung richtig stattgefunden hat. Trat hierbei ein Fehler auf, erscheint die Meldung:

```
VERIFY ERROR (Befehlszeile)
```

Nachfolgend ein Beispiel, in dem sämtliche Dateien von Laufwerk A nach Laufwerk B mit Verify übertragen werden:

B:=A:.*[V]*

Eine weitere Prüfung können Sie bei Textdateien mit der Echofunktion (Option "E") durchführen. Dabei wird die Datei während des Kopierens auf dem Bildschirm ausgegeben. Auch hierzu ein Beispiel:

B:=A:PROBE.TXT[E]

5.6.5 Kopieren geschützter Dateien

Wollen Sie in eine Datei kopieren, die schreibgeschützt ist (R/O), erfolgt zunächst eine Sicherheitsabfrage:

```
DESTINATION IS R/O, DELETE (Y/N)?
```

Nur wenn Sie hier mit "Y" (Ja) antworten, können Sie in die Datei schreiben. PIP legt nämlich bei jedem Kopiervorgang auf Diskette zunächst eine Zwischendatei (Extension $$$) an. Erst danach wird eine eventuell vorhandene Datei unter gleichem Namen gelöscht und die Zwischendatei umbenannt.

Systemdateien werden normalerweise nicht von PIP kopiert, es sei denn, man gibt die R-Option an, wie z.B.:

B:=A:GEHEIM.DAT[R]

5.6.6 Zusammenfassen binärer Dateien

Aus Abschnitt 5.4 wissen wir, daß PIP nur solche Dateien zusammenfaßt, die mit CTRL-Z (^Z) enden, was in der Regel bei Textdateien der Fall ist. In speziellen Anwendungsfällen kann es manchmal erforderlich sein, die Abfrage auf CTRL-Z aufzuheben, so daß die gesamte physikalische Dateilänge übertragen wird. Im folgenden Beispiel werden mit der O-Option drei Binärdateien zusammengefaßt:

GESAMT.BIN=DATEI1.BIN,DATEI2.BIN,DATEI3.BIN[O]

Werden Dateien nur einzeln übertragen, also nicht verkettet, kann man auf die O-Option verzichten, da hier keine Abfrage auf CTRL-Z stattfindet. Dies gilt ganz besonders für das Kopieren von COM-Dateien.

6 Stapelverarbeitung

6.1 Allgemeines

Wir haben nun einen Großteil der verfügbaren CP/M-Befehle kennengelernt und können damit schon eine ganze Menge anfangen. Je mehr wir jedoch mit CP/M arbeiten, desto häufiger kommt es vor, daß wir bestimmte Befehlsfolgen immer wieder ausführen müssen, wobei wir jedesmal die Befehle einzeln eingeben und durch Drücken der RETURN-Taste abschliessen.

Vielleicht haben Sie sich bereits die Finger wundgetippt, weil Sie immer und immer wieder die gleichen Befehlsfolgen eingegeben haben. Sicher war das eine gute Übung für Sie, denn Sie wollen ja beim Studium dieses Buches CP/M kennenlernen. Da Sie aber nun schon (fast) ein Profi sind, fragen Sie sich sicherlich zu Recht, ob man bestimmte Befehlsfolgen, die sich ständig wiederholen, auch einfacher handhaben kann.

Genau damit wollen wir uns in diesem Kapitel beschäftigen, und CP/M hilft uns sogar dabei. Wir lernen im folgenden nämlich die Stapelverarbeitung (englisch: Batch Processing) kennen, bei der wir alle Befehle auf einen Stapel (Befehlsdatei) legen, der dann abgearbeitet wird.

Dieser Vorgang ist sehr einfach und leicht zu verstehen. Wir schreiben sämtliche Befehle zeilenweise in eine normale Textdatei, wie wir es bei ED bereits kennengelernt haben. Dann rufen wir nur noch das CP/M-Dienstprogramm SUBMIT auf, das die Befehle der Reihe nach abarbeitet.

6.2 SUBMIT

Nun wollen wir einmal sehen, wie so etwas vor sich geht und sehen uns zur besseren Veranschaulichung wieder ein einfaches Beispiel an.

Angenommen, Sie möchten das Directory der Diskette auflisten, dann die Datei PROBE.TXT mit einem Schreibschutz versehen und schließlich diese Datei auflisten. Die Datei PROBE.TXT hatten wir bereits in Kapitel 4 angelegt und wollen sie hier wieder für unsere Beispiele verwenden.

Diese drei Befehle legen wir zunächst in einer Textdatei ab. Da sie zur Stapelverarbeitung dient, muß ihr Name mit der Extension SUB gekennzeichnet sein. Deshalb geben wir der Datei den Namen LIST.SUB.

In Kapitel 4, in dem wir uns mit ED befaßten, wurden bereits verschiedene Möglichkeiten erörtert, wie eine Texdatei angelegt werden kann. Haben Sie beispielsweise WordStar zur Verfügung, so können Sie damit die Datei schreiben. Da es sich jedoch nur um wenige Textzeilen handelt, wählen wir hier den CP/M-Editor ED und geben folgendes ein:

ED LIST.SUB

```
NEW FILE
   :*I
 1 : DIR
 2 : SET PROBE.TXT [RO]
 3 : TYPE PROBE.TXT
 4 : <CTRL-Z>
 4 :E
```

Wir haben jetzt die Datei LIST.SUB mit folgenden Befehlen angelegt:

```
DIR
SET PROBE.TXT [RO]
TYPE PROBE.TXT
```

Sollte Ihnen die Handhabung des Editors zu kompliziert erscheinen, können Sie die gleiche Datei auch von BASIC aus erzeugen, beispielsweise mit folgendem Programm (für den CPC 6128):

```
10 OPENOUT "LIST.SUB"
20 PRINT #9, "DIR"
30 PRINT #9, "SET PROBE.TXT [RO]"
40 PRINT #9, "TYPE PROBE.TXT"
50 CLOSEOUT
```

Achten Sie darauf, daß sich sämtliche Dateien und Programme, die hier benötigt werden, auf einer Diskette befinden. Dabei handelt es sich, neben der Befehlsdatei LIST.SUB, auch um die Dateien SET.COM, TYPE.COM und SUBMIT.COM, die Sie gegebenenfalls von der Systemdiskette auf Ihre Arbeitsdiskette kopieren müssen. Verwenden Sie dazu am besten PIP. Da auf die Arbeitsdiskette Schreibzugriffe vorgenommen werden, darf sie auf keinen Fall schreibgeschützt sein, weder mechanisch (Riegel am Diskettengehäuse) noch softwaremäßig (mit SET).

Wenn Sie jetzt

SUBMIT LIST

eingeben, werden sämtliche Befehle ausgeführt, die in der Datei LIST.SUB abgelegt sind. Auf dem Bildschirm sieht dies genauso aus, als hätten Sie jeden Befehl einzeln eingegeben:

```
A>SUBMIT LIST

A>DIR
A: PROBE   TXT : PROBE   BAK : SUBMIT   COM : LIST     SUB
A: $$$     SUB
A>SET PROBE.TXT [RO]

A:PROBE.TXT set to Directory (DIR), Read Only (RO)

A>TYPE PROBE.TXT
Dies ist Zeile 1.
Dies ist Zeile 2.
Dies ist Zeile 3.
Dies ist Zeile 4.
Dies ist Zeile 5.

A>
```

Für die Stapelverarbeitung können Sie auch zwei Laufwerke verwenden, jedoch muß sich die SUBMIT.COM-Datei immer in Laufwerk A befinden, das nicht schreibgeschützt sein darf. Alle anderen Dateien können sich aber in Laufwerk B befinden.

Wenn beispielsweise die Befehlsdatei LIST.SUB in Laufwerk B abgelegt ist, muß der SUBMIT-Aufruf folgendermaßen aussehen:

SUBMIT B:LIST

Wenn SUBMIT eine Befehlsfolge abarbeitet, wird zunächst die SUB-Datei, in unserem Fall LIST.SUB, in eine Zwischendatei mit der Bezeichnung $$$.SUB kopiert, die auf derselben Diskette wie SUBMIT angelegt wird, also immer im Laufwerk A. Anschließend wird die Zwischendatei gelesen und jeweils der Befehl aus ihr herausgenommen, der gerade abgearbeitet werden soll. Wenn sämtliche Befehle ausgeführt sind, ist die Zwischendatei leer und wird automatisch wieder gelöscht.

Da unsere Befehlsdatei auch den DIR-Befehl enthält, erscheint selbstverständlich die $$$.SUB-Datei auch im Directory, wie aus der Auflistung ersichtlich ist.

Zusätzlich bietet SUBMIT noch die Möglichkeit, Variablen bzw. Parameter an die Befehlsdatei zu übergeben. Dadurch wird ein noch komfortableres Arbeiten ermöglicht.

Unsere Befehlsdatei enthält zwei Zeilen, in denen der Dateiname PROBE.TXT vorkommt. Wenn wir nun die gleichen Befehle mit einer anderen Datei ausführen wollen, müßten wir die Befehlsdatei jedesmal ändern. Dies können wir dadurch umgehen, daß wir für den Dateinamen eine Variable einsetzen. Die Befehlsdatei LIST.SUB sähe in diesem Fall so aus:

```
DIR
SET $1 [RO]
TYPE $1
```

Anstelle des Dateinamens haben wir hier "$1" angegeben, was dem ersten übergebenden Parameter entspricht. Weitere Parameter wären entsprechend mit "$2", "$3" usw. gekennzeichnet. Wir übergeben hier aber nur einen Parameter, für den wir den Dateinamen PROBE.TXT setzen. Dann lautet der SUBMIT-Befehlsaufruf folgendermaßen:

SUBMIT LIST PROBE.TXT

Soll die Befehlsdatei beispielsweise nicht mit PROBE.TXT, sondern mit LAGER.DAT abgearbeitet werden, müßten wir folgende Befehlszeile eingeben:

SUBMIT LIST LAGER.DAT

Wenn wir nun nicht nur einen, sondern mehrere Parameter an die Befehlsdatei übergeben, die darin mit $1, $2, $3 usw. gekennzeichnet sind, werden die Parameter in der SUBMIT-Befehlszeile der Reihe nach aneinandergefügt:

SUBMIT SUB-Datei Parameter 1 Parameter 2 Parameter 3 ...

Hier ein weiteres Beispiel: Angenommen, wir wollen die Dateien PROBE.TXT, LISTE.TXT und BRIEF.TXT in BAK-Dateien umbenennen, ohne den REN-Befehl dreimal hintereinander aufrufen zu müssen. Dazu legen wir folgende Befehlsdatei unter dem Namen NEUNAME.SUB an:

```
REN $1.BAK=$1.TXT
REN $2.BAK=$2.TXT
REN $3.BAK=$3.TXT
```

Als Variable dient hier nur der Dateiname ohne Extension, da diese bereits in der Befehlsdatei angegeben ist. Die Umbenennung der drei Dateien findet nun mit folgendem SUBMIT-Befehl statt:

SUBMIT NEUNAME PROBE LISTE BRIEF

Wir könnten auch für die jeweilige Extension eine Variable angeben, wobei TXT gleich "$4" und BAK gleich "$5" entspricht. Die Befehlsdatei sähe dann so aus:

```
REN $1.$5=$1.$4
REN $2.$5=$2.$4
REN $3.$5=$3.$4
```

Sie muß mit folgendem SUBMIT-Befehl aufgerufen werden:

SUBMIT NEUNAME PROBE LISTE BRIEF TXT BAK

Was geschieht nun aber, wenn beispielsweise SUBMIT eine Zwischendatei umbenennen soll, deren Dateiname ebenfalls das Dollarzeichen enthält? Die Antwort ist einfach, denn vor jedes Dollarzeichen wird ein weiteres gesetzt. Auch hierzu ein Beispiel, in dem die Befehlsdatei NEUNAME.SUB folgende Zeile enthält:

```
REN $1.TXT=$1.$$$$$$
```

SUBMIT wird jetzt beispielsweise so aufgerufen:

SUBMIT NEUNAME LAGER

In diesem Fall wird die Zwischendatei LAGER.$$$ in LAGER.TXT umbenannt. Die sechs Dollarzeichen, die in der Befehlsdatei stehen, entsprechen in Wirklichkeit nur den drei Zeichen für die Extension $$$.

Grundsätzlich können Sie auf diese Weise jeden Parameter an die Befehlsdatei übergeben. Achten Sie aber darauf, daß die Parameter selbst keine Leerzeichen enthalten, die beim SUBMIT-Aufruf als Trennzeichen interpretiert werden. Umgekehrt kann eine SUBMIT-Zeile nur richtig abgearbeitet werden, wenn die einzelnen Parameter durch Leerzeichen getrennt sind.

6.3 Übergabe von Parametern an Programme

Mit der bisher behandelten Befehlsdatei konnten wir immer nur Parameter an die eigentliche Befehlszeile übergeben, niemals aber an andere Programme, die durch die Befehlsdatei aufgerufen werden.

Angenommen, wir rufen den Editor ED mit einem bestimmten Dateinamen auf, eine Aufgabe, die wir durchaus mit SUBMIT durchführen könnten. Das Arbeiten mit ED ist aber erst dann sinnvoll, wenn wir nach dem Aufruf weitere Anweisungen an den Editor übergeben, die z.B. Zeilen löschen oder einfügen. Mit einem einfachen SUBMIT-Befehl ist dies jedenfalls nicht möglich.

Weitere Fälle dieser Art finden wir bei PIP oder beim Debugger SID vor, die ebenfalls erst nach dem Aufruf Befehle entgegennehmen.

Sie wollen z.B. wiederholt mit PIP verschiedene Kopiervorgänge durchführen, ohne dabei jedesmal die Befehle einzeln aufrufen zu müssen. Die folgende Befehlsdatei KOPIE.SUB kopiert mit Hilfe von PIP eine BAK-Text-

datei im Bezugslaufwerk in eine TXT-Datei um, die anschließend über den Bildschirm und über den Drucker ausgegeben werden soll:

```
PIP
<$1.TXT=$1.BAK
<CON:=$1.TXT
<LST:=$1.TXT
<
```

In der ersten Zeile wird das Programm PIP normal aufgerufen, so daß es sich jetzt im Befehlsmodus befindet. Normalerweise steht nun der Cursor hinter dem Sternchen und wartet auf die Eingabe von Befehlen.

Zur Übergabe der einzelnen PIP-Befehle setzen wir an den Zeilenanfang das Zeichen "<" und geben dann den Befehl normal ein. Da PIP durch einfaches Drücken der RETURN-Taste wieder verlassen wird, steht in der letzten Zeile lediglich "<", was diesem Vorgang entspricht.

Nun wollen wir die Befehlsdatei ausführen und geben für "$1" wieder unsere Textdatei PROBE.TXT an, hier jedoch ohne Extension, da diese bereits in der Befehlsdatei berücksichtigt ist. Der Aufruf mit SUBMIT sieht dann folgendermaßen aus:

SUBMIT KOPIE PROBE

Selbstverständlich hätten wir auch bei dieser Anwendung mehrere Parameter übergeben können. Auf diese Weise kann man z.B. ED aufrufen und sich die Befehlsdatei selbst verändern lassen, indem einzelne Zeilen gelöscht oder neu eingefügt werden. Beim Debugger SID, den wir noch kennenlernen werden, könnten wir sogar so weit gehen, daß wir ein Maschinenprogramm (COM-Datei) oder das CP/M-Betriebssystem während der Ausführung der Befehlsdatei verändern. Sie sehen, daß Ihrer Phantasie dabei kaum Grenzen gesetzt sind.

6.4 Die Datei PROFILE.SUB

Wenn das CP/M Plus-Betriebssystem geladen und initialisiert wird, fragt es automatisch das Directory nach einer PROFILE.SUB-Datei ab. Ist diese vorhanden, wird sie ausgeführt, ohne daß der Benutzer etwas dazu beitragen muß.

Eine PROFILE-SUB-Datei unterliegt den gleichen Regeln wie SUBMIT-Dateien, d.h. sie können eine beliebige Anzahl von Befehlszeilen enthalten. Nur eine Übergabe von Parametern findet hier nicht statt. Dies wäre aber auch nicht sinnvoll, da die Parameter vorher angegeben werden müßten.

Auf der Systemdiskette des CPC 6128 ist bereits eine solche Datei abgelegt, allerdings nicht unter dem Namen PROFILE.SUB, sondern unter PROFILE.ENG. Da sie aber nicht die Extension SUB, sondern ENG enthält, wird sie beim CP/M-Start nicht abgearbeitet.

Wenn wir dennoch die Ausführung dieser Datei wünschen, müssen wir sie zunächst mit REN umbenennen:

REN PROFILE.SUB=PROFILE.ENG

Jetzt schauen wir uns einmal an, was diese Datei an Befehlen enthält. Dazu geben wir

TYPE PROFILE.SUB

ein und erhalten:

```
SETKEYS KEYS.CCP
LANGUAGE 3
```

Den SETKEYS-Befehl hatten wir bereits in Kapitel 3 kennengelernt. Er dient zur Neubelegung der Tastatur, wobei die einzelnen Tastenwerte in einer zusätzlichen Textdatei enthalten sein müssen. Hier ist es die Datei KEYS.CCP, die ein komfortableres Arbeiten mit den Editiertasten in der CP/M-Befehlszeile ermöglicht.

Der zweite angegebene Befehl lautet LANGUAGE 3. Er schaltet auf den britischen Zeichensatz um, der sich nur geringfügig von dem amerikanischen unterscheidet.

Wir haben nun die Möglichkeit, die PROFILE.SUB-Datei in dieser Form zu übernehmen, sie zu erweitern oder neu zu schreiben. Mit einem Texteditor könnten wir uns beispielsweise folgende PROFILE.SUB-Datei erstellen:

```
SETKEYS KEYS.CCP
SETKEYS DEUTSCH.KEY
LANGUAGE 2
PALETTE 8, 60
DATE SET
```

Nach dem Systemstart werden nun automatisch sowohl die Editiertasten als auch die Tasten für die deutschen Sonderzeichen (DEUTSCH.KEY, siehe Kapitel 3) definiert. Anschließend schaltet LANGUAGE 2 auf den deutschen Zeichensatz um, und PALETTE 8, 60 erzeugt gelbe Schrift auf rotem Hintergrund. Schließlich fragt DATE SET noch nach dem Tagesdatum und der Uhrzeit und stellt die interne Uhr ein. DATE SET ist eine

Sonderform des DATE-Befehls und ist besonders gut für Befehlsdateien geeignet.

Auch die Systemdiskette des JOYCE enthält eine PROFILE.SUB-Datei, die hier jedoch unter dem Namen PROFILE.GER abgelegt ist und zur Ausführung entsprechend umbenannt werden muß. Wenn wir diese Datei mit TYPE auflisten, erhalten wir:

```
SETDEF M:,* [ORDER = (SUB,COM) TEMPORARY = M:]
PIP
<M:=BASIC.COM[O]
<M:=DIR.COM[O]
<M:=ERASE.COM[O]
<M:=PAPER.COM[O]
<M:=PIP.COM[O]
<M:=RENAME.COM[O]
<M:=SHOW.COM[O]
<M:=SUBMIT.COM[O]
<M:=TYPE.COM[O]
<
```

Hier wird also eine ganze Reihe von COM-Dateien mittels PIP von Laufwerk A in das virtuelle Laufwerk M kopiert. Sie werden sich jetzt fragen, wozu dies gut sein soll, denn diese Dateien befinden sich ja ohnehin auf der Systemdiskette.

Die Antwort finden wir in der ersten Zeile mit dem SETDEF-Befehl, der einen bestimmten Suchpfad zum Auffinden von Dateien festlegt. In der Befehlszeile sind insgesamt drei Optionen hintereinander angegeben, die auch jeweils einzeln hätten erteilt werden können. Deshalb wollen wir sie jetzt als Einzelanweisungen behandeln, d.h. jeweils unter Angabe von SETDEF:

*SETDEF M:,**

Hiermit sucht CP/M Plus eine angegebene Datei zuerst in Laufwerk M, selbst wenn die Bezeichnung "M" nicht angegeben wurde. Wird die Datei nicht gefunden, wird die Suche im Bezugslaufwerk (normalerweise "A") fortgesetzt. Der nächste Befehl

SETDEF [ORDER = (SUB,COM)]

läßt zuerst nach SUB-Dateien (Befehls-bzw. Stapel-Dateien) suchen. Wird keine SUB-Datei gefunden, wird die Suche nach einer COM-Datei fortgesetzt. Falls Sie z.B.

BILANZ

eingeben, sucht CP/M Plus zuerst nach einer Befehlsdatei BILANZ.SUB, um diese auszuführen. Ist diese Datei jedoch nicht vorhanden, wird BILANZ.COM gesucht und ausgeführt.

Schließlich bestimmt

SETDEF [TEMPORARY = M:]

das virtuelle Laufwerk M als Laufwerk für temporäre Dateien. Dies sind beispielsweise Zwischendateien, die vorübergehend von PIP oder SUBMIT- bzw. PROFILE.SUB-Dateien erzeugt, nach Abarbeitung des jeweiligen Befehls aber wieder gelöscht werden.

Wurde nun die PROFILE.SUB-Datei des JOYCE ausgeführt und rufen Sie eine der kopierten Dateien auf, so sucht CP/M Plus sie zuerst im Laufwerk M. Nehmen wir ein Beispiel: Angenommen, Sie legen in Laufwerk A eine Diskette ein, die nicht das Programm PIP enthält, wenn Sie es aufrufen. Normalerweise erscheint dann die Meldung:

```
PIP?
```

CP/M kann also diesen Aufrufbefehl nicht verstehen. Statt dessen wird zunächst in Laufwerk "M" eine Datei PIP.SUB gesucht, und da diese nicht vorhanden ist, anschließend die Datei PIP.COM. Wenn Sie nun mit PIP Dateien kopieren, werden eventuell angelegte Zwischendateien, deren Namen meist $-Zeichen enthalten, ebenfalls in Laufwerk M abgelegt und später wieder gelöscht.

Der SETDEF-Befehl ist besonders für den JOYCE geeignet, um einen möglichst großen Nutzen aus dem virtuellen Laufwerk M zu ziehen. Selbstverständlich steht dieser Befehl auch für den CPC 6128 zur Verfügung, jedoch müßte man hier mit einem echten Laufwerk arbeiten.

7 Debugging

Die Überschrift dieses Kapitels wird Sie vielleicht überraschen, besonders dann, wenn Sie sich unter Debugging nichts vorstellen können. Debugging ist aber durchaus nichts Ungewöhnliches in der Computerwelt und dient zur Fehlersuche oder -beseitigung.

Debugging kommt von dem englischen Wort "Bug" (Käfer, Wanze, Motte). Sie werden sich jetzt sicher fragen, was eine Fehlersuche mit Insekten zu tun hat. Dafür gibt es eine ganz einfache Erklärung, jedenfalls wenn man der folgenden überlieferten Legende Glauben schenken darf.

In ihrer Anfangszeit waren die Computer, anders als heute, noch riesig groß, so daß sie ganze Räume oder sogar Häuser füllten. Damals enthielten sie nicht etwa die kleinen ICs, wie heute, sondern Relais, die für jede Informationseinheit (Bit) einen Stromkreis öffneten oder schlossen.

Eines Tages trat bei einer solchen Computeranlage eine Störung auf. Als man der Sache auf den Grund ging und die unzähligen Relais auf ihre Funktionstüchtigkeit hin überprüfte, fand man heraus, daß sich in einem Relais eine Motte eingeklemmt hatte, die die ganze Anlage zum Erliegen gebracht hatte.

An diesem Tag wurde der Begriff Debugging geboren, der sich seitdem aber nicht nur auf Relais bezieht. Heute ist ein Debugger meist ein Hilfsprogramm, das man einsetzt, wenn in einem Programm "der Wurm drin sitzt". Mit einem Debugger kann man Speicherbereiche in hexadezimaler Schreibweise auflisten und ändern. Häufig bietet er aber noch weitere Möglichkeiten, wie beispielsweise Speicherbereiche verschieben oder mit einem konstanten Wert füllen.

Auch für CP/M gibt es einen solchen Debugger, der sich unter dem Namen SID.COM auf Ihrer Systemdiskette befindet. Mit ihm werden wir uns in diesem Kapitel noch eingehend befassen.

7.1 DUMP

Die einfachste Möglichkeit, eine Datei in hexadezimaler Schreibweise zu betrachten, bietet das DUMP-Programm. Wir wollen es jetzt anhand unserer in Kapitel 4 angelegten Datei PROBE.TXT einmal ausprobieren.

Wenn Sie nur ein Laufwerk zur Verfügung haben, müssen Sie die DUMP.COM-Datei erst auf die Diskette kopieren, die auch PROBE.TXT enthält. Der Programmaufruf findet dann mit

DUMP PROBE.TXT

statt, d.h. Sie müssen hinter DUMP noch den Namen der Datei angeben, die Sie listen möchten.

Bei zwei Laufwerken legen Sie am besten die Systemdiskette in Laufwerk A und die Diskette, die PROBE.TXT enthält, in Laufwerk B. Sie können auch hier wieder zwischen beiden Laufwerken umschalten, falls Sie nur eines zur Verfügung haben. Jetzt starten Sie DUMP mit

DUMP B:PROBE.TXT

In beiden Fällen erhalten Sie daraufhin die nachfolgende Auflistung:

```
0000: 44 69 65 73 20 69 73 74 20 5A 65 69 6C 65 20 31 Dies ist Zeile 1
0010: 2E 0D 0A 44 69 65 73 20 69 73 74 20 5A 65 69 6C ...Dies ist Zeil
0020: 65 20 32 2E 0D 0A 44 69 65 73 20 69 73 74 20 5A e 2...Dies ist Z
0030: 65 69 6C 65 20 33 2E 0D 0A 44 69 65 73 20 69 73 eile 3...Dies is
0040: 74 20 5A 65 69 6C 65 20 34 2E 0D 0A 44 69 65 73 t Zeile 4...Dies
0050: 20 69 73 74 20 5A 65 69 6C 65 20 35 2E 0D 0A 1A  ist Zeile 5....
0060: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0070: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
```

Wenn Sie sich bisher noch wenig in die Innereien Ihres Computers beschäftigt haben, werden Ihnen diese Zeichen auch nicht allzuviel sagen. Trotzdem wollen wir sie uns einmal näher ansehen.

PROBE.TXT ist eine reine Textdatei, die im ASCII-Code abgelegt ist. Deshalb bestehen die Zeichen, die wir hier sehen, aus reinem ASCII-Code. Auf der rechten Seite erscheint der Text außerdem in lesbarer Form, denn alle HEX-Zeichen, die einem lesbaren ASCII-Code entsprechen, werden automatisch in Klartext umgewandelt, während andere lediglich als Punkt erscheinen.

In jeder Zeile werden insgesamt 16 Bytes bzw. ASCII-Zeichen dargestellt, wobei der vierstellige Wert am Zeilenanfang die relative Position zum Dateianfang angibt. Somit beginnt die erste Zeile mit 0000H (0 dez.), die zweite mit 0010H (16 dez.) usw.

Wir erinnern uns, daß die erste Textzeile in der Datei PROBE.TXT folgendermaßen lautete:

```
Dies ist Zeile 1.
```

Das bestätigt auch die Auflistung der ASCII-Zeichen. Trotzdem wollen wir jetzt einmal die ersten vier Bytes der Datei genauer untersuchen, wozu wir die ASCII-Code-Tabelle im Anhang zu Hilfe nehmen. Bei richtiger Vorgehensweise finden wir dabei folgendes heraus:

	Hex.	Dez.	ASCII-Zeichen
1. Zeichen	44	68	D
2. Zeichen	69	105	i
3. Zeichen	65	101	e
4. Zeichen	73	115	s

Wir erhalten tatsächlich das Wort "Dies". Das Spiel könnten wir beliebig fortsetzen, um den gesamten Text der Datei lesbar zu machen, aber darauf wollen wir hier verzichten. Dennoch interessieren uns noch einige Sonderzeichen, die in jeder Textdatei immer wieder vorkommen.

Wenn wir den aufgelisteten ASCII-Code zeichenweise untersuchen, fällt uns mehrfach die Zeichenfolge 0D 0A auf, erstmals an der Position 11H in der zweiten Zeile. Dabei bedeuten

Hex.	Dez.	
0D	13	Carriage Return (Wagenrücklauf)
0A	10	Line-Feed (Zeilenvorschub)

Mit diesen Zeichen wird eine Textzeile abgeschlossen und eine neue eingeleitet. Wenn der Computer diese Zeichen liest, setzt er den Cursor an den Anfang der nächsten Zeile. Ähnliches geschieht auch beim Drucker, bei dem das Papier um eine Zeile hochgeschoben und der Druckkopf am Zeilenanfang positioniert wird.

Da DUMP nicht die logische, sondern die physikalische Dateilänge liest und anzeigt, werden immer nur ganze Records (128 Byte) ausgegeben. Dies ist auch in unserem Beispiel der Fall. Dabei fällt uns auf, daß die letzten Bytes des Records nicht belegt sind und alle das Dateiende-Zeichen 1AH (26 dez., CTRL-Z) enthalten.

Vom Arbeiten mit dem Editor wissen wir, daß jeder Text mit diesem Zeichen abgeschlossen werden muß, und daß eine Textdatei solange eingelesen wird, bis das Dateiende-Zeichen auftritt.

Theoretisch würde hier ein 1A-Zeichen durchaus genügen; CP/M füllt jedoch die Bytes des letzten Records, die nicht belegt sind, mit diesem Zeichen auf.

Mit DUMP können Sie jedoch nicht nur ASCII-Dateien, sondern auch beliebige andere Dateien im Hexcode auflisten. Versuchen Sie es doch einmal

mit einer COM-Datei, z.B. mit PIP.COM. Solche Dateien bestehen zumeist aus einer Mischung aus Maschinen- und ASCII-Code, denn sämtliche auszugebenden Meldungen sind auch hier im ASCII-Code abgelegt. Die nicht belegten Bytes des letzten Records enthalten dabei meist 00-Zeichen.

Mit DUMP können Sie den Inhalt von Dateien zwar ansehen, aber leider nicht verändern. Das ist nur mit SID möglich, wie wir noch sehen werden.

7.2 SID

SID ist ein universelles Debugging-Programm, mit dem man Speicherinhalte auflisten, ändern oder sogar in mnemonischer Form (Assembler-Quelltext) ausgeben kann. SID ist ein Standard-CP/M-Programm, das sich auf der mitgelieferten Systemdiskette befindet.

7.2.1 SID laden

SID kann, ähnlich wie PIP, auf zweierlei Weise geladen und aktiviert werden. Wir wollen dies wieder anhand unserer Beispieldatei PROBE.TXT ausprobieren. Geben Sie jetzt

SID PROBE.TXT

ein, wenn sich SID auf der gleichen Diskette wie PROBE.TXT befindet oder

SID B:PROBE.TXT

wenn SID in Laufwerk A (z.B. auf der Systemdiskette) und PROBE.TXT in Laufwerk B abgelegt ist.

SID meldet sich jetzt mit:

```
CP/M 3 SID - Version 3.0
NEXT MSZE  PC  END
0180 0180 0100 DAFF
#
```

Die Datei PROBE.TXT wurde zusammen mit SID in den Speicher geladen und ist jetzt zwischen Adresse PC (0100H) und NEXT (0180H) abgelegt. Diese beiden Werte sollten wir uns unbedingt merken (am besten notieren!), wenn wir die geänderte Datei später wieder auf Diskette schreiben wollen. In diesem Fall dürfen wir aber auch nicht vergessen, vorher das SAVE-Programm (siehe unten) auszuführen. Als weiteres geben MSZE die Größe der längsten im Speicher befindlichen Datei (hier identsch mit NEXT) und END die TPA-Obergrenze an.

Als Alternative können wir SID und die zu bearbeitende Datei auch separat laden. Dazu geben wir zunächst

SID

ein, worauf sich SID mit

```
CP/M 3 SID - Version 3.0
#
```

meldet. Das Ziffernkreuz (#) bedeutet, wie auch schon im obigen Fall, daß SID zur Aufnahme von Befehlen bereit ist. Jetzt legen wir die Diskette mit der zu untersuchenden Datei (in unserem Fall wieder PROBE.TXT) ins Laufwerk und geben dann

R PROBE.TXT

ein, worauf die Datei PROBE.TXT geladen wird. Schließlich erscheint, wie bereits oben:

```
NEXT MSZE  PC  END
0180 0180 0100 DAFF
#
```

Normalerweise legt SID die Dateien ab Adresse 100H, dem Beginn des TPA, ab. Dies ist auch die Anfangsadresse, an die sämtliche COM-Dateien automatisch geladen werden. Auch wir wollen für den Normalfall diese Adresse beibehalten, solange wir mit SID arbeiten.

Hinter "R" und Dateiname können wir aber noch einen Parameter als Versatz angeben, wenn wir eine andere Ladeadresse als 100H wünschen. So lädt beispielsweise

R PROBE.TXT 1000

die Datei PROBE.TXT mit einem Versatz von 1000H, d.h. ab Adresse 0100H+1000H=1100H. Auf diese Weise können wir verschiedene Dateien gleichzeitig im Speicher ablegen und eventuell mischen.

7.2.2 SAVE

SAVE ist ein Dienstprogramm, das beliebige Speicherinhalte als Datei auf Diskette schreibt. Im Gegensatz zu CP/M 2.2, wo SAVE ein residenter Befehl ist, wird unter CP/M Plus eine Datei SAVE.COM ausgeführt. Dabei unterscheidet sich die Arbeitsweise von SAVE erheblich von der unter CP/M 2.2.

SAVE steht in engem Zusammenhang mit dem Debugger-Programm SID. Wenn Sie nämlich eine Datei mit SID bearbeitet haben, können Sie sie mit SAVE wieder auf Diskette schreiben. Beachten Sie aber dazu den folgenden wichtigen Hinweis:

SAVE ist immer *vor* dem Aufruf von SID auszuführen!

Wenn Sie dies vergessen, können Sie unter Umständen Ihre geänderte Datei nicht wieder auf Diskette schreiben. Führen Sie nämlich SAVE aus, wenn sich eine andere Datei im Speicher befindet, wird ein Teil des TPA überschrieben und die Datei dabei zerstört. Zwar entfällt bei SAVE die umständliche Berechnung der Speicherinhalte in Pages (256 Byte), wie bei der CP/M 2.2-Version, doch führt die Nichtbeachtung der Regel leicht zu nicht unerheblichen Verlusten: wenn mühsam bearbeitete Dateien durch Überschreiben verloren gehen.

Bevor Sie also mit SID arbeiten, geben Sie zunächst

SAVE

ein. Dabei geschieht nach außen gar nichts, denn SAVE wird hierdurch in den oberen Speicherbereich des TPA verschoben. Daraufhin erscheint lediglich wieder das Bereitschaftszeichen (Prompt).

Nun erst geben Sie

SID

ein, entweder mit oder ohne Dateiname, worauf Sie Ihre Datei normal bearbeiten können. Im Anschluß daran geben Sie

G0 (Null, nicht Buchstabe "O")

ein, und SAVE meldet sich folgendermaßen:

```
CP/M 3 SAVE - Version 3.0
Enter file (type RETURN to exit):
Beginning hex address
Ending hex address
```

Nach der Bereitschaftsmeldung werden Sie aufgefordert, den Dateinamen mit Extension anzugeben, unter dem der Speicherinhalt abgelegt werden soll. Anschließend ist die Anfangs- und Endadresse dieses Bereichs in hexadezimaler Form einzugeben.

Damit Sie nicht versehentlich vergessen, SAVE rechtzeitig zu laden, können Sie sich beispielsweise unter dem Namen DEBUG.SUB eine SUBMIT-Datei (siehe Kapitel 6) anlegen, die folgende Befehlszeilen enthält:

```
SAVE
SID
```

Wenn Sie jetzt

SUBMIT DEBUG

ausführen, wird SAVE automatisch geladen, ohne daß Sie sich darum kümmern müssen.

Sie können aber das ganze Risiko mit SAVE umgehen, indem Sie vom SID-Befehlsmodus aus

W Dateiname,Anfangsadresse,Endadresse

(Adressen hexadezimal) eingeben. So legt z.B.

W ABC.DEF,1200,2000

den Speicherinhalt zwischen 1200H und 2000H unter dem Dateinamen ABC.DEF auf Diskette ab.

7.2.3 Speicher ansehen

Ebenso wie mit DUMP können wir auch mit SID den Datei-Inhalt in hexadezimaler Schreibweise ansehen, allerdings in einer etwas anderen Form. SID liest nicht die Datei und listet sie gleichzeitig, sondern kann nur den Speicherinhalt listen. Dabei spielt es keine Rolle, ob im Speicher eine Datei abgelegt ist oder nicht.

Wenn wir nun eine Datei ansehen, müssen wir den Speicherbereich ab 100H listen, nämlich ab der Stelle, an die SID normalerweise Dateien ablegt, wenn beim Laden kein Versatz angegeben ist. Dies wollen wir wieder mit unserer Datei PROBE.TXT ausprobieren.

Zunächst müssen wir SID und PROBE.TXT nach einer der oben gezeigten Methoden laden. Ist das geschehen, erscheint das Ziffernkreuz (#) zur weiteren Befehlseingabe. Wenn wir jetzt

D

eingeben, werden die ersten 12 Zeilen mit je 16 Byte ab Adresse 100H gelistet:

```
0100: 44 69 65 73 20 69 73 74 20 5A 65 69 6C 65 20 31 Dies ist Zeile 1
0110: 2E 0D 0A 44 69 65 73 20 69 73 74 20 5A 65 69 6C ...Dies ist Zeil
0120: 65 20 32 2E 0D 0A 44 69 65 73 20 69 73 74 20 5A e 2...Dies ist Z
0130: 65 69 6C 65 20 33 2E 0D 0A 44 69 65 73 20 69 73 eile 3...Dies is
0140: 74 20 5A 65 69 6C 65 20 34 2E 0D 0A 44 69 65 73 t Zeile 4...Dies
0150: 20 69 73 74 20 5A 65 69 6C 65 20 35 2E 0D 0A 1A  ist Zeile 5....
0160: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0170: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0180: 1A 84 12 13 C3 69 01 D1 2E 00 E9 2A EC 01 22 E7 .....i.....*.."."
0190: 08 23 22 ED 08 3A EB 01 32 D5 0A 32 EA 0F 32 F4 .#"..:..2..2..2.
01A0: 10 C3 3D 01 5F 1E C9 11 00 00 0E 12 CD 05 00 32 ..=._..........2
01B0: 5F 1E C9 21 68 1E 70 2B 71 2A 67 1E EB 0E 13 CD _..!h.p+q*g.....
```

Wir erhalten die gleiche Auflistung wie mit DUMP, nur mit anderen Adressenangaben. Auch hier werden sämtliche Bytes angegeben, soweit möglich. Außerdem sind sie rechts noch als ASCII-Zeichen zu sehen. Deshalb erkennen wir auch sofort, wo Text abgelegt ist und wo nicht. Für alle Zeichen, die keinem listbaren ASCII-Code entsprechen, erscheint ein Punkt.

Da unsere Datei PROBE.TXT nur wenige Zeichen umfaßt, wird auch noch Speicher angezeigt, der nicht mehr zu PROBE.TXT gehört. Aus diesem Grund interessieren uns alle Adressen, die größer als 180H sind, in diesem Fall nicht mehr.

Anders ist es, wenn wir eine Datei ansehen wollen, die mehr Speicherplatz belegt. Reichen nämlich die ersten 12 Zeilen nicht aus, geben wir nochmals "D" ein, worauf die nächsten 12 Zeilen erscheinen, usw.

Wir können auch eine Speicheradresse vorgeben, ab der 12 Zeilen gelistet werden sollen, z.B.:

D2800

Hier erfolgt die Speicherauflistung ab Adresse 2800H. Es besteht auch die Möglichkeit, eine Anfangs- und Endadresse vorzugeben, wie im folgenden Beispiel:

D1800,1980

In diesem Fall wird der Speicher von Adresse 1800H bis 1980H angezeigt.

7.2.4 Speicher ändern

Als nächstes wollen wir unsere Datei PROBE.TXT einmal mit SID ändern. Am Anfang steht der ASCII-Code für "Dies", den wir jetzt durch den Code für "Hier" ersetzen wollen. Da die Begriffe "Dies" und "Hier" jeweils vier Zeichen umfassen, bereitet uns die Änderung keine Schwierigkeit.

Wir suchen uns zunächst die Codes für "Hier" aus der ASCII-Code-Tabelle heraus und geben dann

S100

ein, denn wir wollen den Speicher ja ab Adresse 100H ändern. Daraufhin erscheint die jeweilige Adresse mit dem Inhalt der Speicherzelle. Wollen wir den Inhalt belassen, drücken wir die RETURN-Taste, worauf der Inhalt der nächsten Zelle erscheint. Soll der Inhalt jedoch verändert werden, schreiben wir den neuen Code einfach daneben und drücken erst dann die RETURN-Taste. Durch Eingabe eines Punktes kehren wir wieder in den SID-Befehlsmodus zurück. Wenn Sie nun "Dies" durch "Hier" richtig ausgetauscht haben, muß auf dem Bildschirm folgendes stehen:

```
#S100
0100 44 48
0101 69
0102 65
0103 73 72
0104 20 .
```

Eigentlich müßten wir hier nur das erste und das letzte Zeichen ändern, da beide Begriffe in der Mitte zufällig "ie" enthalten. Deshalb haben wir das zweite und dritte Zeichen übergangen, indem wir die RETURN-Taste drückten. Wir hätten aber ebenso gut die Zeichen neu eingeben können.

Jetzt kontrollieren wir, ob wir die Begriffe richtig ausgetauscht haben, indem wir nochmals mit

D100

den Speicher auflisten und uns rechts die Textzeichen ansehen.

Selbstverständlich kann man mit dem S-Befehl nicht nur ASCII-Code, sondern auch Maschinenprogramme ändern, indem man den Maschinencode neu eingibt. Einen solchen Vorgang bezeichnet man als "Patchen", was in etwa mit "Flicken" zu übersetzen ist.

Falls Sie nun die Datei PROBE.TXT in abgeänderter Form abspeichern möchten, müssen Sie G0 eingeben. Sie können nun den SAVE-Befehl ausführen, sofern er vorher initialisiert wurde (siehe oben). Da es sich bei PROBE.TXT um eine Texdatei handelt, sollten Sie eine Endadresse wählen, die mindestens ein Textende-Zeichen (1AH) mit einschließt.

7.2.5 Assembler/Disassembler

SID besitzt einen einfachen Assembler, mit dem man mnemonische Codes zur Maschinenprogrammierung eingeben kann. Er wird mit

A (Speicheradresse)

aufgerufen. Mit diesem Assembler wollen wir uns aber nicht näher beschäftigen, da er bei weitem nicht so komfortabel ist wie der Assembler ASM, den wir im nächsten Kapitel noch kennenlernen werden.

Zum Assembler gibt es noch einen einfachen Disassembler, der den Speicherinhalt in mnemonischer Schreibweise auslistet und mit dem L-Befehl aufgerufen wird. Beispielsweise disassembliert die Eingabe

L1200,1300

den Speicher zwischen den Adressen 1200H und 1300H. Wird kein gültiger Opcode vorgefunden, erscheint ein Fragezeichen.

7.2.6 Weitere Möglichkeiten

Mit den Befehlen, die wir bisher kennengelernt haben, sind die Möglichkeiten von SID aber noch nicht erschöpft. Deshalb betrachten wir in diesem Abschnitt noch einige weitere Befehle, die in erster Linie für den Maschinenprogrammierer von Interesse sind.

Manchmal kommt es vor, daß man einen ganzen Speicherbereich mit einem konstanten Wert füllen möchte. Zu diesem Zweck gibt es den F-Befehl, mit dem wir folgenden Versuch machen. Geben Sie einmal ein:

F2000,2100,0

und anschließend:

D2000,2100

Sie werden feststellen, daß der Bereich von 2000H bis 2100H mit lauter Nullen gefüllt ist.

Diesen mit Nullen gefüllten Bereich wollen wir jetzt verschieben und zwar *so, daß er ab Adresse 4000H abgelegt wird. Wir können das mit dem M*-Befehl erreichen, den wir folgendermaßen einsetzen:

M2000,2100,4000

Die ersten beiden Werte geben dabei die ursprüngliche Anfangs- und Endadresse und der dritte Wert gibt die Anfangs-Zieladresse an. Nach Ausführung des Befehls muß der Block mit Nullen also auch ab 4000H erscheinen, was Sie ebenfalls mit dem D-Befehl nachprüfen können.

Wenn Sie mit einem Assembler ein Maschinenprogramm geschrieben haben, möchten Sie es sicher auch austesten, was Sie bei SID mit dem G-Befehl tun können. Auch hierzu ein Beispiel:

G135A,1764

läßt ein Maschinenprogramm ab der Adresse 135A ablaufen. Bei 1764H wird ein Haltepunkt gesetzt, so daß es dort abbricht. Es ist keinesfalls erforderlich, daß sich an dieser Stelle ein Abbruchbefehl (z.B. RET) befindet, denn SID setzt ihn hier selbst ein.

Soll die Abarbeitung bei der gegenwärtigen Programmzähler-Adresse beginnen, können Sie den ersten Wert auch weglassen, dürfen aber das Komma nicht vergessen. Bei

G,23AA

beginnt die Abarbeitung bei der gegenwärtigen Programmzähler-Adresse und endet bei Adresse 23AAH.

Eine Abwandlung von G ist der T-Befehl, in dem man eine bestimmte Anzahl von Programmschritten vorgibt, bei deren Abarbeitung jeweils die Prozessorregister ausgegeben werden. Darüber hinaus gibt es noch den U-Befehl, der ebenso wie "T" arbeitet, jedoch keine Registerinhalte ausgibt. "T" und "U" beginnen die Programmausführung bei der jeweiligen Programmzähler-Adresse. So führen die Anweisungen

T23

und

U23

jeweils 23H (35 dez.) Programmschritte aus, wobei im ersten Fall auch die Registerinhalte erscheinen.

Schließlich gibt es noch den X-Befehl, mit dem die Prozessorregister angezeigt und geändert werden können.

X

ohne weiteren Zusatz zeigt die Register an, während beispielsweise

XA

den Akkumulatorinhalt aufzeigt, für den man, ähnlich wie beim S-Befehl, einen neuen Wert daneben schreiben kann.

8 Assemblerprogrammierung

Dieses Kapitel ist für diejenigen bestimmt, die die Möglichkeiten der Assemblerprogrammierung unter CP/M nutzen möchten. Doch bevor wir näher darauf eingehen, wollen wir zunächst einmal klären, was ein Assembler ist und welche Aufgaben er übernimmt.

8.1 Ein paar Worte vorweg

Grundsätzlich dient ein Assembler zur Generierung eines ausführbaren Maschinencodes. Zwar könnten Sie ein Maschinenprogramm theoretisch auch mit dem S-Befehl von SID schreiben, indem Sie sich sämtliche Hexcodes selbst zusammen suchen, um diese erst im Speicher und dann auf der Diskette abzulegen. SID eignet sich aber nur zum Ausbessern einiger weniger Bytes, was man in der Fachsprache als Patchen bezeichnet. Wenn Sie jedoch ganze Programme damit schreiben, wird dies zu einer sehr mühsamen Angelegenheit, die nicht nur eine genaue Kenntnis des internen Maschinencodes erfordert, sondern auch einige Anforderungen in hexadezimalem Kopfrechnen stellt. Sie müssen nämlich sämtliche relativen und absoluten Adressen korrekt berechnen und den Speicher richtig einteilen können.

Zweifellos erhalten Sie, wenn Sie diese Voraussetzungen erfüllen und sonst keinen Fehler machen, ein ebensolches Maschinenprogramm wie mit einem Assembler. Aber warum sollen wir es uns so kompliziert machen, wenn es auch einfacher geht!

Jeder Assembler benötigt, ganz gleich für welchen Zweck und für welchen Mikroprozessor er eingesetzt wird, einen Quelltext mit mnemonischen Befehlen. Dies sind genormte Schlüsselwörter, die der Assembler erkennt und in einen Maschinencode umwandelt. Wenn Sie in BASIC programmieren, geschieht übrigens etwas ganz ähnliches, nur auf einer weitaus höheren Ebene als bei einem Assembler. Dabei werden die BASIC-Schlüsselwörter in einen internen Code umgewandelt, den der BASIC-Interpreter versteht und mit dem er entsprechende Maschinenroutinen aufruft.

Bekanntlich kann CP/M nur auf solchen Computern betrieben werden, die einen 8080- oder Z80-Mikroprozessor besitzen, was auch für den Schneider CPC zutrifft. Zur Enstehungszeit von CP/M gab es nur den 8080-Prozessor, einen Vorgänger vom Z80, mit einem weitaus geringeren Befehlssatz. Bedauerlicherweise verstehen die Assembler für den 8080 einen anderen mnemonischen Code als die für den Z80, obwohl sie einen Maschinencode generieren, der zum Z80 voll kompatibel ist.

Obwohl es den 8080-Prozessor heutzutage gar nicht mehr gibt und alle modernen Computer dieser Art mit einem Z80-Prozessor ausgerüstet sind, gehört immer noch ein 8080-Assembler zur Standard-Ausrüstung eines jeden CP/M-Systems. Wegen seines geringeren Befehlssatzes muß man deshalb auf alle Befehle vezichten, die der Z80-Prozessor zusätzlich bietet.

Trotz dieses Handicaps werden wir uns in diesem Kapitel mit dem CP/M-8080-Assembler beschäftigen, der auf Ihrer Systemdiskette mitgeliefert wird. Selbst mit weniger Befehlen können wir immer noch eine ganze Menge anfangen. Übrigens beziehen sich die meisten Assemblerlistings für CP/M in der Fachliteratur ebenfalls auf den 8080-Assembler.

Sollten Sie jedoch in der glücklichen Lage sein, einen Z80-Assembler zu besitzen (für den CPC werden mehrere angeboten), können Sie damit ebenfalls CP/M-Programme schreiben und somit den gesamten Befehlssatz des Z80 ausschöpfen. Dies gilt selbst dann, wenn der Assembler nur unter AMSDOS arbeitet. In diesem Fall müssen Sie lediglich darauf achten, daß das Assemblerprogramm bei Adresse 100H startet und die genormten BDOS- und BIOS-Schnittstellen verwendet.

Dieses Kapitel kann und soll einen vollständigen Assemblerlehrgang nicht ersetzen, denn dafür gibt es genügend andere Literatur auf dem Markt. Wir wollen uns aber mit der Bedienung des 8080-Assemblers befassen und in diesem Zusammenhang auch ein kleines CP/M-Programm schreiben und austesten.

8.2 Der Quelltext

Wir wissen bereits, daß ein Assembler einen Quelltext benötigt, den er dann in den eigentlichen Maschinen- oder Objektcode übersetzt. Dieser Quelltext unterliegt ganz bestimmten Regeln und besteht ausschließlich aus listbarem ASCII-Code, der mit jedem Texteditor erstellt werden kann. Darüber hinaus muß der Quelltext in einer Datei abgelegt werden, deren Name die Extension ASM enthält, damit sie der Assembler später lesen kann. Wir schreiben jetzt ein kleines Assemblerprogramm, das Sie nach Ihrem Namen fragt und Ihnen dann herzliche Grüße von CP/M ausrichtet.

Zunächst geben wir den Quelltext für dieses Programm ein und legen ihn dann in der Datei MUSTER.ASM ab. Sie können den Text mit ED oder jedem anderen Texteditor, wie z.B. WordStar, erstellen, der reinen ASCII-Code erzeugt. Wir wollen hier jedoch den Editor ED verwenden, der sich mit auf der Systemdiskette befindet. Den Umgang mit ED hatten wir bereits in Kapitel 4 kennengelernt. Obwohl dieser Editor nicht gerade kom-

fortabel in der Bedienung ist, eignet er sich dennoch zum Schreiben von Quelltextdateien recht gut.

Zunächst rufen wir ED mit dem Dateinamen MUSTER.ASM auf und gehen gleich anschließend in den i-Modus zum Schreiben und Einfügen von Text über:

ED MUSTER.ASM

```
NEW FILE
   : *i
  1:
```

Jetzt geben wir zeilenweise den folgenden Quelltext ein:

```
;Herzliche Gruesse von CP/M
;==========================
;
;
;
ORG 100H ; CP/M-Programmstart
BDOS EQU 5 ;BDOS-Aufruf
CCP EQU 0 ;Warmstart
;
;
MVI C,9 ; 1.String ausgeben
LXI D,P1
CALL BDOS
;
;
MVI C,10 ; Texteingabe
LXI D,EINGABE
CALL BDOS
;
;
MVI C,9 ; 2.String ausgeben
LXI D,P2
CALL BDOS
;
;
MVI D,0 ; $-Zeichen an Ende von Eingabe
LDA EINGABE+1
MOV E,A
LXI H,EINGABE
DAD D
INX H
INX H
MVI A,'$'
MOV M,A
;
;
MVI C,9
LXI D,EINGABE+2
CALL BDOS
;
;
```

```
JMP CCP ; Warmstart
;
;
P1 DB 'Ihren Namen bitte? $'
P2 DB 0DH,0AH,0AH,'CP/M gruesst Sie herzlich - $'
EINGABE DB 25
DS 26
END
```

Nach Abschluß der Eingabe schreiben wir den Text mit dem E-Befehl auf Diskette.

Bevor wir fortfahren noch einige Erläuterungen zum Aufbau des Quelltextes. Beim Schreiben jeder Zeile müssen Sie folgende Reihenfolge einhalten:

<Label> <Befehlswort> <Argumente> ; <Kommentar>

Ein Label ist eine Markierung für eine symbolische Adresse und wird meist als Argument für Sprungbefehle verwendet. Es kann aber auch, wie in unserem Beispiel, die Stelle kennzeichnen, an der ein bestimmter ASCII-Text abgelegt ist oder an der sich ein Puffer befindet. Das Label muß jeweils, falls vorhanden, an erster Stelle in der Zeile aufgeführt sein und darf keinem anderen Begriff mit vorgegebener Bedeutung, wie z.B. den Befehlswörtern, entsprechen. Manchmal werden Labels mit einem Doppelpunkt abgeschlossen, was beim CP/M-Assembler aber nicht unbedingt erforderlich ist.

An nächster Stelle steht das Befehlswort, das durch mindestens ein Leerzeichen vom Label getrennt sein muß. Es entspricht der mnemonischen Schreibweise für den Assembler. Falls das Befehlswort noch Argumente benötigt, sind diese dahinter aufzuführen, ebenfalls im Abstand von mindestens einem Leerzeichen.

Schließlich folgt der Kommentar, der vom Assembler nicht berücksichtigt wird und einen beliebigen Text zur Erläuterung enthalten kann. Er dient lediglich der Übersicht und ist keinesfalls zwingend. Wird er aber verwendet, so ist er durch ein Semikolon von dem übrigen Text zu trennen.

Am Anfang eines jeden Quelltextes muß außerdem die ORG-Anweisung stehen, die angibt, ab welcher Stelle im Speicher das Programm später geladen und ausgeführt werden soll. In unserem Fall gibt sie die Adresse 100H an, an der sämtliche CP/M-Programme starten.

Doch nun zu den eigentlichen Anweisungen des Programms. Hinter ORG sind die Label mit den festen Adressen für den BDOS-Aufruf und den Warmstart aufgeführt, denen wir hier die Namen BDOS und CCP gegeben

haben. Die eigentliche Zuordnung zu diesen Adressen findet durch EQU statt. Da bei einem BDOS-Aufruf immer die Routine in Adresse 5 anzuspringen ist, lautet die Anweisung entsprechend "BDOS EQU 5".

Als erstes gibt das Programm den String "Ihren Namen bitte?" aus, der unter dem Label P1 am Schluß des Programms aufzufinden ist. Dazu verwendet es die BDOS-Routine mit der Nummer 9, die für diesen Zweck vorgesehen ist. Allerdings müssen Strings, wie auch in diesem Fall, mit einem $-Zeichen abschließen, welches BDOS als Stringende interpretiert.

Zum eigentlichen Aufruf der BDOS-Routine ist das C-Register mit deren Nummer zu laden (hier Nr. 9) und das Doppelregister D mit der Anfangsadresse des Strings, die hier durch das Label P1 gekennzeichnet ist. Schließlich ist nur noch BDOS in Adresse 5 aufzurufen und der String erscheint auf dem Bildschirm.

Als nächstes erfolgt eine Texteingabe von der Konsole, die normalerweise Ihren Namen enthalten sollte. Auch hierzu gibt es eine BDOS-Routine, die mit der Nummer 10 aufgerufen wird. Zusätzlich ist noch die Adresse des Puffers im D-Register anzugeben, in den die eingegebenen Zeichen abgelegt werden sollen und der hier mit dem Label "EINGABE" gekennzeichnet ist. Der Puffer kann maximal 25 Textzeichen aufnehmen, benötigt aber am Anfang zwei weitere Bytes, in denen die maximale und die wirkliche Länge des eingegebenen Strings abgelegt wird. Deshalb enthält das erste Byte des Puffers die Zahl 25, worauf sich 26 weitere Bytes anschließen (1 Byte für die wirkliche Stringlänge plus 25 Bytes zur Ablage des Strings).

Nach der Texteingabe - abzuschließen durch RETURN - wird sogleich ein weiterer String mit dem Inhalt "CP/M gruesst Sie herzlich -" ausgegeben, der mit dem Label P2 gekennzeichnet ist. Er beginnt mit einem CR- und zwei LF-Zeichen, damit er zwei Zeilen unter der ersten Zeile erscheint.

Nun soll in der gleichen Zeile auch noch der eingegebene Name ausgegeben werden, den wir direkt aus dem Eingabepuffer herauslesen. Dies ist jedoch nicht so ohne weiteres möglich, denn wir müssen ihn am Schluß noch mit einem $-Zeichen versehen. Erst wenn das geschehen ist, können wir ihn wieder mit der BDOS-Routine 9 ausgeben.

Nachdem sämtliche Routinen ausgeführt sind, wollen wir wieder zu CP/M zurückkehren und springen die symbolische Adresse CCP an. Dies hat die gleiche Wirkung wie die Eingabe von CTRL-C, nämlich die Ausführung eines Warmstartes.

Abschließend noch ein paar Worte zur Speicherreservierung für die Strings und den Eingabepuffer. Wie Sie sehen, werden konstante Werte immer hin-

ter der DB-Anweisung abgelegt. Wenn es sich um ASCII-Code handelt, können die entsprechenden Textzeichen direkt angegeben werden, wobei sie allerdings in einfache Anführungszeichen einzuschließen sind.

Soll nur freier Speicher für bestimmte Zwecke reserviert werden, wie hier für den größten Teil des Eingabepuffers, muß das mit der Anweisung DS (Anzahl) geschehen.

Jedes Assemblerlisting sollte mit END abschließen, was wir auch hier nicht vergessen wollen.

8.3 Quelltext assemblieren

Nachdem wir nun unseren Quelltext (hoffentlich richtig) in der Datei MUSTER.ASM abgelegt haben, können wir ihn assemblieren. Dazu benötigen wir den Assembler MAC.COM, der sich auf der Systemdiskette befindet. Im Gegensatz zu manchen anderen Assemblern erzeugt er jedoch noch nicht den reinen Maschinencode, sondern eine ASCII-Datei im Intel-Hex-Format, die erst später mit HEXCOM in Objektcode umgewandelt wird. Zusätzlich wird in einer anderen Datei ein Assembler-PRN-Listing erzeugt, das zur eigentlichen Dokumentation des Programms dient.

Wenn Sie mit einem Laufwerk arbeiten, kopieren Sie am besten die Dateien MAC.COM und HEXCOM.COM auf die Diskette, die den Quelltext enthält. Bei zwei Laufwerken belassen Sie die Systemdiskette wieder in Laufwerk A und die Diskette mit dem Quelltext in Laufwerk B.

Nun rufen Sie den Assembler mit

MAC MUSTER

bzw.

MAC B:MUSTER

auf, worauf die Assemblierung stattfindet. Sollten Ihnen bei der Erstellung des Quelltextes Fehler unterlaufen sein, werden die fehlerhaften Zeilen auf dem Bildschirm angezeigt. Sie müssen dann den Quelltext nochmals korrigieren. Ansonsten erscheint lediglich

```
CP/M MACRO ASSEM 2.0
0181
000H USE FACTOR
END OF ASSEMBLY
```

auf dem Bildschirm, d.h. der Assembler hat keinen Fehler festgestellt.

Auf Ihrer Diskette befinden sich jetzt drei weitere Dateien mit dem Name MUSTER, aber mit verschiedenen Extensionen, die wir mit TYPE auflisten können. Zunächst einmal die PRN-Datei mit dem Assembler-PRN-Listing:

TYPE MUSTER.PRN

```
                ;Herzliche Gruesse von CP/M
                ;==========================
                ;
                ;
                ;
0100                    ORG 100H ; CP/M-Programmstart
0005 =                  BDOS EQU 5 ;BDOS-Aufruf
0000 =                  CCP EQU 0 ;Warmstart
                ;
                ;
0100 0E09               MVI C,9 ; 1.String ausgeben
0102 113201             LXI D,P1
0105 CD0500             CALL BDOS
                ;
                ;
0108 0E0A               MVI C,10 ; Texteingabe
010A 116601             LXI D,EINGABE
010D CD0500             CALL BDOS
                ;
                ;
0110 0E09               MVI C,9 ; 2.String ausgeben
0112 114601             LXI D,P2
0115 CD0500             CALL BDOS
                ;
                ;
0118 1600               MVI D,0 ; $-Zeichen an Ende von Eingabe
011A 3A6701             LDA EINGABE+1
011D 5F                 MOV E,A
011E 216601             LXI H,EINGABE
0121 19                 DAD D
0122 23                 INX H
0123 23                 INX H
0124 3E24               MVI A,'$'
0126 77                 MOV M,A
                ;
                ;
0127 0E09               MVI C,9
0129 116801             LXI D,EINGABE+2
012C CD0500             CALL BDOS
                ;
                ;
012F C30000             JMP CCP ; Warmstart
                ;
                ;
0132 496872656EP1 DB 'Ihren Namen bitte? $'
0146 0D0A0A4350P2 DB 0DH,0AH,0AH,'CP/M gruesst Sie herzlich - $'
0166 19                 EINGABE DB 25
0167                    DS 26
0181                    END
```

Wir erhalten also nochmals den Quelltext, der jetzt aber mit Adressen und Maschinencode versehen ist. Als nächstes listen wir mit

TYPE MUSTER.SYM

die Symboldatei, die die einzelnen Labels mit ihren Adressen angibt:

```
0005 BDOS       0000 CCP        0166 EINGABE    0132 P1         0146 P2
```

Schließlißlich folgt mit

TYPE MUSTER.HEX

die Datei im Intel-Hex-Format, die folgendermaßen aussieht:

```
:100100000E09113201CD05000E0A116601CD050060
:100110000E09114601CD050016003A67015F216600
:1001200011923233E24770E09116801CD0500C370
:10013000000496872656E204E616D656E206269CF
:10014000747465 3F20240D0A0A43502F4D206772B6
:1001500075657373742053696520686572A6C697C
:0701600063682020241923
:0000000000
```

Mit diesem Code können wir allerdings nicht viel anfangen; wir benötigen ihn aber noch zur Erstellung einer ausführbaren COM-Datei. Dies erreichen wir mit:

HEXCOM MUSTER

bzw.

HEXCOM B:MUSTER

Nachdem nun HEXCOM seine Arbeit verrichtet hat, erscheint:

```
FIRST ADDRESS 0100      (Startadresse)
LAST  ADDRESS 0166      (Endadresse)
BYTES READ    0067      (Gelesene Bytes = Länge der Datei)
RECORDS WRITTEN 01      (Belegte Records)
```

Wenn wir uns jetzt das Directory anschauen, finden wir unsere Datei MUSTER mit sechs verschiedenen Extensionen vor:

```
MUSTER.ASM    (Quelltext)
MUSTER.BAK    (BAK-Datei des Editors)
MUSTER.PRN    (Assembler-PRN-Listing)
MUSTER.HEX    (Intel-Hex-Listing)
MUSTER.SYM    (Symboldatei)
MUSTER.COM    (Ausführbare CP/M-COM-Datei)
```

Die zuletzt aufgeführte COM-Datei ist nun das Endresultat unserer Arbeit, und wir wollen unser Programm auch gleich einmal testen. Dazu starten wir es ganz normal mit

MUSTER

und, wenn alles stimmt, erscheint folgende Ausgabe auf dem Bildschirm:

```
Ihren Namen bitte? Franz Huber                    (Namen eingeben)
CP/M gruesst Sie herzlich - Franz Huber
```

Manchmal ist es nützlich, neben der Symboltabelle noch eine weitere Tabelle zu erhalten, die angibt, an welchen Stellen die einzelnen Label aufgerufen werden. Geben Sie nun einmal

XREF MUSTER

ein. Sie erhalten jetzt eine zusätzliche Datei mit dem Namen MUSTER.XRF mit folgendem Inhalt:

```
 1                  ;HERZLICHE GRUESSE VON CP/M
 2                  ;==========================
 3                  ;
 4                  ;
 5                  ;
 6   0100           ORG 100H ; CP/M-PROGRAMMSTART
 7   0005 =         BDOS EQU 5 ;BDOS-AUFRUF
 8   0000 =         CCP EQU 0 ;WARMSTART
 9                  ;
10                  ;
11   0100 0E09      MVI C,9 ; 1.STRING AUSGEBEN
12   0102 113201    LXI D,P1
13   0105 CD0500    CALL BDOS
14                  ;
15                  ;
16   0108 0E0A      MVI C,10 ; TEXTEINGABE
17   010A 116601    LXI D,EINGABE
18   010D CD0500    CALL BDOS
19                  ;
20                  ;
21   0110 0E09      MVI C,9 ; 2.STRING AUSGEBEN
22   0112 114601    LXI D,P2
23   0115 CD0500    CALL BDOS
24                  ;
25                  ;
26   0118 1600      MVI D,0 ; $-ZEICHEN AN ENDE VON EINGABE
27   011A 3A6701    LDA EINGABE+1
28   011D 5F        MOV E,A
29   011E 216601    LXI H,EINGABE
30   0121 19        DAD D
31   0122 23        INX H
32   0123 23        INX H
33   0124 3E24      MVI A,'$'
34   0126 77        MOV M,A
```

```
35                      ;
36                      ;
37   0127 0E09          MVI C,9
38   0129 116801        LXI D,EINGABE+2
39   012C CD0500        CALL BDOS
40                      ;
41                      ;
42   012F C30000        JMP CCP ; WARMSTART
43                      ;
44                      ;
45   0132 496872656EP1 DB 'Ihren Namen bitte? $'
46   0146 0D0A0A4350P2 DB 0DH,0AH,0AH,'CP/M gruesst Sie herzlich - $'
47   0166 19            EINGABE DB 25
48   0167               DS 26
49   0181               END
BDOS                0005    7#  13  18  23  39
CCP                 0000    8#  42
EINGABE             0166    17  27  29  38  47#
P1                  0132    12  45#
P2                  0146    22  46#
```

Der erste Teil der Auflistung ist MUSTER.PRN sehr ähnlich. Er enthält noch zusätzliche Zeilennummern, auf die sich die Angaben in der Referenztabelle beziehen, die unten angefügt ist. Betrachten wir das Label BDOS als Beispiel. Es ist in Zeile 7 definiert (Ziffernkreuz #) und wird in den Zeilen 13, 18, 23 und 29 aufgerufen. Die XRF-Datei eignet sich besonders für umfangreiche Assembler-Listings, in denen man somit die einzelnen Referenzpunkte leicht auffinden kann.

Neben dem Assembler MAC gibt es noch den Assembler RMAC. Auch RMAC erzeugt aus einer ASM- eine PRN- und eine SYM-Datei, jedoch keine HEX-Datei im Intel-Hex-Code. Statt dessen wird eine REL-Datei angelegt, die ein relatives (verschiebbares) Objektmodul enthält. Rufen Sie nun einmal

RMAC MUSTER

auf, wodurch unter anderem auch eine Datei MUSTER.REL erzeugt wird. Mit

LINK MUSTER

wird diese Datei in eine ausführbare Datei MUSTER.COM umgewandelt, und zwar genauso, wie es auch mit MAC und HEXCOM möglich ist.

Das Arbeiten mit RMAC und LINK bietet gegenüber MAC und HEXCOM noch einige zusätzliche Möglichkeiten, die in Kapitel 10 näher erläutert werden.

9 Hinter den Kulissen

In den vorangehenden Kapiteln haben wir bereits sämtliche Befehle kennengelernt, die wir zum normalen Arbeiten mit CP/M Plus benötigen. Im allgemeinen ist dies völlig ausreichend, sofern Sie nicht in das "Innere" von CP/M einsteigen möchten; aber genau das wollen wir in diesem Kapitel tun. Vielleicht gehören Sie zu den begeisterten Assemblerprogrammierern, die eigene CP/M-Programme schreiben und das Letzte aus dem CP/M-Betriebssystem herausholen möchten. Das setzt allerdings eine genaue Kenntnis der Speicherorganisation, der Aufrufadressen und einiger wichtiger Parameter voraus, mit denen wir uns nachfolgend beschäftigen werden.

An dieser Stelle sei aber nochmals betont, daß das vorliegende Buch keinen Assemblerlehrgang und keine komplette Abhandlung sämtlicher Einzelheiten über die interne Funktionsweise von CP/M Plus enthält. Diese Themen sind so umfangreich, daß sie genügend Stoff für mehrere Bücher dieser Art liefern würden. Wir beschränken uns hier deshalb auf die wichtigsten systeminternen Informationen, die für normale Anwendungsfälle eines versierten Assemblerprogrammierers völlig ausreichend sind. Sollte dies in Einzelfällen nicht genügen, wird auf die folgende englischsprachige Dokumentation von Digital Research verwiesen:

CP/M Plus Programmer's Guide
CP/M Plus Operating System Guide

9.1 Interne Speicherorganisation

Wie wir bereits wissen, benötigt CP/M Plus im Regelfall zwei oder mehrere Speicherbänke. Unter einer Speicherbank versteht man 64 KByte, die von der Z80-CPU direkt adressierbar sind. So besitzt der CPC 6128 zwei und der JOYCE sogar vier RAM-Speicherbänke. Bei beiden Computern werden jedoch immer nur zwei Bänke für CP/M Plus genutzt. Die zusätzlichen beiden Bänke des JOYCE, von denen 110K zur Ablage von Dateien zur Verfügung stehen, dienen dem virtuellen Laufwerk M.

Während es für CP/M 2.2 sehr einfach ist, die genaue Speicherbelegung anzugeben, ist dies bei CP/M Plus etwas schwierig, da ständig zwischen beiden Speicherbänken hin- und hergeschaltet wird. Der CPC 6128 adressiert unmittelbar nach dem Einschalten die Speicherbank 0, die normalerweise auch von BASIC genutzt wird. Lediglich das BANKMAN-Programm greift als BASIC-Erweiterung auch auf die zweite Speicherbank 1 zu, um dort Dateien oder Bildschirminhalte abzulegen.

Der Arbeitsspeicher unter CP/M Plus, also der TPA, liegt dagegen immer in Bank 1 und belegt dort 61 KByte - von Adresse 0100H bis F4FFH. Auch wenn Sie mit dem Debugger SID ein Programm untersuchen, befinden Sie sich immer in Bank 1. Eine Umschaltung auf Bank 0 ist jedenfalls nicht ohne weiteres möglich.

Beide Bänke haben jedoch einen Speicherbereich, auf den sie gemeinsam zugreifen, und der mit Common-Bereich bezeichnet wird. Er liegt beim CPC 6128 und JOYCE zwischen C000H und FFFFH, umfaßt also auch einen Teil des TPA. Oberhalb des TPA, also ab Adresse F500H, befindet sich das residente BDOS und BIOS sowie einige hardwarebedingte Puffer. Resident bedeutet hier, daß auf die im Common-Bereich abgelegten BDOS- und BIOS-Teile vom TPA aus ohne Bankumschaltung zugegriffen werden kann, während der Haupteil (gebankte Teil) von BDOS und BIOS in Bank 0 liegt. Die residenten Teile von BDOS und BIOS stellen, da sie im Common-Bereich liegen, somit die Schnittstelle zwischen beiden Bänken dar.

Daß auch ein Teil des TPA im Common-Bereich liegt, ist ohne Bedeutung, denn die Bankumschaltung kann immer nur in Blöcken zu 16 KByte stattfinden, wodurch auch der obere Teil des TPA erfaßt wird.

In Bank 0 befinden sich, neben dem gebankten BDOS und BIOS, noch der CCP, der nach jedem Kalt- und Warmstart in den TPA (Bank 1) geladen wird. Deshalb muß bei einem Warmstart (CTRL-C) nicht mehr auf die Systemdiskette zugegriffen werden, wie es bei der früheren CP/M-Version 2.2 der Fall war. Als weiteres befindet sich in der Bank 0 das Bildschirm-RAM, allerdings in einem anderen Bereich als unter BASIC, da es ansonsten den gesamten Common-Bereich beanspruchen würde.

Der restliche Speicher von Bank 0 ist zur Ablage von Disketten-Sektoren reserviert. Vielleicht haben Sie schon einmal bemerkt, daß CP/M Plus nicht jedesmal auf die Diskette zugreift, wenn Sie aus einer Datei lesen oder in sie schreiben. Indem nämlich ein großer Teil von Dateien im RAM abgelegt ist, sind weniger Schreib-/Lesezugriffe auf die Diskette erforderlich, was einerseits einen erheblichen Geschwindigkeitsgewinn und andererseits eine schonende Behandlung von Laufwerken und Disketten zur Folge hat.

Wie nun die genaue Speicheraufteilung in Bank 0 aussieht, ist schwer zu sagen, da die Bankumschaltung und der Zugriff von außen sehr schwierig sind. In der Regel können wir aber auf eine genaue Kenntnis verzichten, denn CP/M Plus enthält einige BDOS-Aufrufe, die ein Lesen und Verändern aller wichtigen Parameter ermöglicht, die in Bank 0 abgelegt sind. Doch mehr darüber später.

9.2 Directory und File-Control-Block (FCB)

Auf Diskette abgelegte Dateien müssen nach einem ganz bestimmten Schema verwaltet werden, wofür das Directory zuständig ist. Es enthält nämlich noch weit mehr Informationen, als mit dem DIR-Befehl aufgelistet werden können.

Das Directory befindet sich bei CP/M immer in der Spur, die den Systemspuren unmittelbar folgt, beim CPC also in Spur 2, wenn das Betriebssystem die Spuren 0 und 1 belegt. In dieser Spur sind die ersten vier Sektoren zu je 512 Byte, die insgesamt zwei Blöcke aufnehmen können, für das Directory reserviert. Es kann somit maximal 64 Einträge zu je 32 Byte aufnehmen, einschließlich derjenigen, die ein zusätzliches Extent für eine Datei darstellen.

Wir wollen uns nun den Aufbau eines Directory-Eintrags anhand der folgenden Tabelle einmal genauer ansehen:

Byte	Beschreibung
0	Benutzernummer (0 bis 31)
1 bis 8	Dateiname im ASCII-Code (Großbuchstaben)
9 bis 11	Extension im ASCII-Code (Großbuchstaben)
12	Nummer des Eintrags (Extent)
13 und 14	Für interne Zwecke reserviert
15	Anzahl der abgelegten Records im Extent (0 bis 127)
16 bis 31	Nummern der für die Datei belegten Blöcke, wobei maximal 16 Blöcke pro Extent verwaltet werden können.

Insgesamt kann ein Directory-Eintrag also 16 KByte verwalten, sofern ein Block 1 KByte umfaßt. Dies ist beim CPC und den meisten anderen CP/M-Systemen der Fall. Enthält nun eine Datei mehr als 16 KByte, so ist für jede weiteren angefangenen 16 KByte jeweils ein Eintrag in das Directory vorzunehmen. In diesem Fall spricht man von Zusatzeinträgen oder Extents.

Der File-Control-Block (FCB) befindet sich nicht auf der Diskette, sondern im Arbeitsspeicher, wo er einen Directory-Eintrag zur Bearbeitung aufnimmt. Der FCB hat folgenden Aufbau:

Byte	Beschreibung
0	Laufwerksnummer (0 für Bezugslaufwerk, 1 bis 16 für Laufwerk A bis P)
1 bis 8	Dateiname im ASCII-Code (Großbuchstaben)
9 bis 11	Extension im ASCII-Code (Großbuchstaben)
12	Nummer des Eintrags (Extent)
13 und 14	Für interne Zwecke reserviert
15	Anzahl der abgelegten Records im Extent (0 bis 128)
16 bis 31	Nummern der für die Datei belegten Blöcke, wobei maximal 16 Blöcke pro Extent verwaltet werden können.
32	Nummer des nächsten Records
33 bis 34	Nummer des absoluten Records bei Direktzugriff
35	Flag für Überlauf bei BDOS-Funktion 35

Sie sehen also, daß das Directory und der FCB nahezu identisch aufgebaut sind, wobei der FCB jedoch noch einige Zusatzinformationen enthält. So ist hier statt der Benutzernummer das aktuelle Laufwerk angegeben. Darüber hinaus enthalten die Bytes 32 bis 35 Informationen, die insbesondere zum Lesen und Schreiben von relativen Dateien benötigt werden.

Die Bytes 1 bis 11, die für den Dateinamen samt Extension reserviert sind, enthalten normalerweise den 7 Bit Standard-ASCII-Code, jedoch hat auch das achte Bit bei einigen Zeichen eine bestimmte Bedeutung, sofern es gesetzt ist. Nachfolgend die fest reservierten Bits, während die restlichen für Sonderzwecke des Benutzers zur Verfügung stehen:

Bit 7 in

Byte 9	Read Only-Attribut
Byte 10	System-Attribut
Byte 11	Archieve-Attribut (siehe PIP)

An dieser Stelle wollen wir nicht vergessen, auch einige besondere Directory-Einträge zu betrachten, die wir in Kapitel 3 schon kennengelernt haben. So belegen das Directory-Label und das Datei-Password jeweils einen Directory-Eintrag. Auch Timestamps belegen Einträge, wie unten beschrieben.

Eintrag für Directory-Label:

Byte		Beschreibung
0		Laufwerksnummer (0 für Bezugslaufwerk, 1 bis 16 für Laufwerk A bis P)
1 bis 8		Label im ASCII-Code (Großbuchstaben)
9 bis 11		Typ im ASCII-Code (Großbuchstaben)
12		Angaben zu den Dateien auf der Diskette:
	Bit 7	Password-Schutz aktiviert
	Bit 6	Access-Timestamps aktiviert
	Bit 5	Update-Timestamps aktiviert
	Bit 4	Create-Timestamps aktiviert
	Bit 0	Label existiert
13 bis 15		Nicht belegt
16 bis 23		Password für Label (verschlüsselt)
24 bis 27		Create oder Access-Timestamps (Label)
28 bis 31		Update-Timestamps (Label)

Eintrag für Datei-Password (XFCB-Format):

Byte		Beschreibung
0		Laufwerksnummer (0 für Bezugslaufwerk, 1 bis 16 für Laufwerk A bis P)
1 bis 8		Password im ASCII-Code (Großbuchstaben)
9 bis 11		Password-Typ im ASCII-Code (Großbuchstaben)
12		Angaben zum Password-Schutz:
	Bit 7	Read-Modus (vollständiger Schutz)
	Bit 6	Write-Modus (Schutz nur gegen Schreiben)
	Bit 5	Delete-Modus (Schutz nur gegen Löschen)
	Bit 4	Create-Timestamps aktiviert
13 bis 15		Reserviert für interne Zwecke
16 bis 23		Password (verschlüsselt)
24 bis 31		Reserviert

Einträge für Timestamps (Dateien):

Achtung! Nur nach Ausführung von INITDIR und nach gesonderter Aktivierung (siehe Kapitel 3) zu verwenden. Dabei wird für jeweils drei Einträge ein vierter reserviert, der zwei Timestamps für die drei Einträge aufnimmt. Die Timestamps werden darin im BCD-Format abgelegt.

Byte	Beschreibung
0	21H (zur Kennzeichnung)
1 bis 10	2 Timestamps für 1. Eintrag
11 bis 20	2 Timestamps für 2. Eintrag
21 bis 30	2 Timestamps für 3. Eintrag
31	Reserviert

9.3 BDOS-Aufrufe

Das BDOS enthält insgesamt 69 Funktionen, die in eigenen CP/M-Programmen verwendet werden können. Im Kapitel 8 hatten wir bereits ein kleines Assemblerprogramm entwickelt, das einen String von der Tastatur einliest und auf dem Bildschirm ausgibt. Dies war jedoch nur ein Vorgeschmack auf die vielen Aufgabenstellungen, die mit Hilfe der BDOS-Aufrufe gelöst werden können.

Für jeden BDOS-Aufruf ist die angegebene Nummer im C-Register abzulegen und dann Adresse 5 aufzurufen. Nachfolgend sind sämtliche Aufrufe aufgeführt, einschließlich der Register zur Übergabe bzw. Übernahme der Parameter. 8-Bit-Werte werden im E-Register und 16-Bit-Werte im Doppelregister DE an die BDOS-Routinen übergeben. Dagegen erfolgt die Übernahme von den Routinen mit dem A-Register für 8-Bit-Werte und mit dem Doppelregister HL für 16-Bit-Werte.

Die BDOS-Aufrufe 0 bis 40 sind weitestgehend mit denen von CP/M 2.2 identisch. Deshalb sind CP/M 2.2-Programme, die ausschließlich diese Aufrufe verwenden (nicht BIOS-Aufrufe!) in der Regel auch unter CP/M Plus ablauffähig.

0 SYSTEM RESET

führt einen Warmstart aus. Parameter sind nicht notwendig.

1 CONSOLE INPUT

liest ein Zeichen von der Konsole in Register A ein und gibt es auf dem Bildschirm (Echofunktion) aus.

2 CONSOLE OUTPUT

gibt ein Zeichen in Register E auf der Konsole (Bildschirm) aus. Dabei werden Vorgaben wie Tabulator, CTRL-S oder CTRL-P (zusätzliche Drukkerausgabe) berücksichtigt.

3 AUXILIARY INPUT

liest ein Zeichen in Register A aus dem AUXIN:-Kanal bzw. von einem zugeordneten Gerät (z.B. Modem) ein.

4 AUXILIARY OUTPUT

gibt ein Zeichen in Register E auf den AUXOUT:-Kanal bzw. ein zugeordnetes Gerät (z.B. Modem) aus.

5 LIST OUTPUT

gibt ein Zeichen in Register E auf dem Drucker aus.

6 DIRECT CONSOLE I/O

Unmittelbare Konsoleneingabe eines Zeichens in Register A oder Ausgabe in Register E. Sollte normalerweise nicht verwendet werden.

7 AUXILIARY INPUT STATUS

ergibt in Register A den Wert FFH, wenn ein Zeichen im Hilfskanal AUXIN: zur Eingabe anliegt, anderenfalls den Wert 00H.

8 SET IOBYTE

ergibt in Register A den Wert FFH, wenn im Hilfskanal AUXOUT: ein Zeichen zur Ausgabe anliegt, anderenfalls den Wert 00H.

9 PRINT STRING

gibt einen String (Adresse in Registerpaar DE) auf dem Bildschirm aus, der mit einem $-Zeichen abgeschlossen sein muß.

10 READ CONSOLE BUFFER

Übernimmt einen String von der Konsole und legt ihn in einem Puffer (Adresse in DE) ab. Der String ist mit CR oder LF abzuschließen.

11 GET CONSOLE STATUS

prüft, ob Zeichen von Konsole eingegeben werden (A=0 nein, A=1 ja).

12 RETURN VERSION NUMBER

liefert die Nummer (HL) der verwendeten CP/M-Version. Register H liefert 0 für CP/M 80 und Register L die Nummer 31H, die Versionsnummer des BDOS-File-Systems.

13 RESET DISK SYSTEM

setzt das Diskettensystem zurück (keine Parameter). Dabei wird immer Laufwerk A angewählt und die Datenpufferadresse (DMA) auf 80H gesetzt.

14 SELECT DISK

bestimmt ein Laufwerk (E) zum Bezugslaufwerk. Für Laufwerk A ist der Wert 0, für Laufwerk B der Wert 1 usw. zu übergeben.

15 OPEN FILE

Öffnet Datei (DE auf FCB-Adresse).

16 CLOSE FILE

schließt Datei (DE auf FCB-Adresse).

17 SEARCH FOR FIRST

sucht den ersten Eintrag einer Datei im Directory (DE auf FCB-Adresse). Dabei sind auch mehrdeutige Dateinamen zulässig.

18 SEARCH FOR NEXT

sucht nach Ausführung von Funktion 17 den nächsten Eintrag (keine Parameter erforderlich).

19 DELETE FILE

löscht Datei (DE auf FCB-Adresse), sofern nicht schreibgeschützt. Bei mehrdeutigen Dateinamen können auch mehrere Dateien gelöscht werden.

20 READ SEQUENTIAL

liest den nächsten fortlaufenden Record einer Datei (DE auf FCB-Adresse) und legt ihn im Datenpuffer (DMA) ab. Der Recordzähler wird dabei um eins erhöht.

21 WRITE SEQUENTIAL

schreibt den nächsten fortlaufenden Record einer Datei (DE auf FCB-Adresse) aus dem Datenpuffer (DMA) in eine Datei. Der Recordzähler wird dabei um eins erhöht.

22 MAKE FILE

legt eine Datei neu an (DE auf FCB-Adresse), ähnlich wie bei OPEN. In das Directory erfolgt aber zunächst ein Leereintrag, da die Datei noch keine Daten enthält.

23 RENAME FILE

nennt Dateien um. Dazu ist ein spezieller FCB erforderlich (Adresse in DE), in dem die ersten 16 Bytes den alten und die restlichen 16 Bytes den neuen Dateinamen enthalten.

24 RETURN LOGIN VECTOR

fragt ab, welche Laufwerke angeschlossen sind und gibt in HL einen 16-Bit-Wert zurück. Das niederwertigste Bit steht dabei für Laufwerk A und das höchstwertigste Bit für Laufwerk P. Das jeweilige Bit ist gesetzt, wenn das betreffende Laufwerk in Betrieb ist, anderenfalls ist es gelöscht.

25 RETURN CURRENT DISK

gibt das Bezugslaufwerk an. Dabei wird der Wert 0 für Laufwerk A, 1 für Laufwerk B usw. in Register A übermittelt.

26 SET DMA ADDRESS

setzt Datenpufferadresse (in DE) für Schreib- und Lesevorgänge.

27 GET ADDR (ALLOC)

gibt die Adresse des Blockbelegungs-Verzeichnisses in Registerpaar HL an. Diese Funktion wird z.B. von SHOW verwendet, um den freien Diskettenspeicher zu ermitteln.

28 WRITE PROTECT DISK

versieht das selektierte Laufwerk mit einem temporären Schreibschutz (keine Parameter).

29 GET READ-ONLY VECTOR

ermittelt schreibgeschützte Laufwerke, indem in HL ein 16-Bit-Wert zurückgegeben wird (ähnlich wie bei Funktion f4).

30 SET FILE ATTRIBUTES

setzt Dateiattribute für System- oder schreibgeschützte Dateien bzw. setzt sie zurück, indem der entsprechende Eintrag vom FCB (Adresse in DE) in das Directory übertragen wird.

31 GET DPB ADRRESS

ermittelt die Adresse des Disk-Parameter-Blocks (in HL).

32 SET/GET USER CODE

dient zur Ermittlung und zum Setzen der aktiven Benutzernummer. Soll die Nummer ermittelt werden, enthält Register E den Wert 255, die dann in Register A zurückgegeben wird. Beim Setzen enthält "E" die gewünschte Benutzernummer.

33 READ RANDOM

dient zum Lesen eines Records für Direktzugriffe. Die Recordnummer wird hierbei aber nicht wie beim sequentiellen Lesen (Funktion 20), sondern durch Erweiterungseinträge im FCB (Adresse in DE) übergeben.

34 WRITE RANDOM

dient zum Schreiben eines Records, wobei die Recordnummer durch Erweiterungseinträge im FCB (Adresse in DE) übergeben wird.

35 COMPUTE FILE SIZE

Ermittlung der Dateigröße in Records. Durch Vorgabe der FCB-Adresse in DE wird die Größe berechnet und in den Erweiterungsbytes für relativen Dateizugriff des FCB abgelegt.

36 SET RANDOM RECORD

Ermittlung der augenblicklichen Recordnummer einer Datei und Ablage in den Erweiterungsbytes für relativen Dateizugriff des FCB (Anfangsadresse in DE).

37 RESET DRIVE

Zurücksetzen einzelner Laufwerke durch Übergabe eines 16-Bit-Parameters in DE, wobei das niederwertigste Bit Laufwerk A entspricht.

38 ACCESS DRIVE

Nur für MP/M; für den CPC 6128 und den JOYCE ohne Bedeutung.

39 FREE DRIVE

Nur für MP/M; für den CPC 6128 und den JOYCE ohne Bedeutung.

40 WRITE RANDOM WITH ZERO

Schreiben eines Records wie bei Funktion 34, wobei er jedoch zuvor mit Nullen gefüllt wird.

41 TEST AND WRITE RECORD

Nur für MP/M; beim CPC 6128 und den JOYCE ohne Bedeutung.

42 LOCK RECORD

Nur für MP/M; beim CPC 6128 und den JOYCE ohne Bedeutung.

43 UNLOCK RECORD

Nur für MP/M; beim CPC 6128 und den JOYCE ohne Bedeutung.

44 SET MULTI-SECTOR COUNT

Während fortlaufender BDOS-Schreib- und Lesevorgänge können bis zu 128 Records (Zahl in Register E) auf einmal erfaßt und bearbeitet werden.

45 SET BDOS ERROR MODE

dient zur Behandlung physikalischer und erweiterter Fehler. Enthält Register E den Wert 255, wird der "Return Error Mode" aktiviert, beim Wert 254 ist es der "Return and Display Mode".

46 GET DISC FREE SPACE

ermittelt den noch freien Speicherplatz auf Disketten mit dem RW-Attribut. Laufwerk A ist dabei mit 0, Laufwerk B mit 1 usw. in Register E vorzugeben. Im augenblicklichen DMA-Puffer werden dadurch die ersten drei Bytes mit einer Binärzahl (low, middle, high) belegt, die den freien Speicherplatz in Records angibt.

47 CHAIN TO PROGRAM

ermöglicht den Übergang von einem Programm zum nächsten, ohne daß der Bediener eine spezielle Anweisung erteilen muß. Dazu ist die Befehlszeile im gegenwärtigen DMA-Puffer abzulegen und mit 00H abzuschließen. Zusätzlich ist Register E mit FFH zu belegen. Das neue Programm wird dann vom Bezugslaufwerk eingelesen und anschließend ausgeführt.

48 FLUSH BUFFERS

veranlaßt, daß sämtliche Records in den Blocking-/Deblocking-Puffern auf Diskette geschrieben werden. Ist zusätzliche das E-Register mit FFH vorbelegt, werden sämtliche aktiven Datenpuffer gelöscht.

49 GET/SET SYSTEM CONTROL BLOCK

dient zum Lesen/Schreiben des System-Control-Blocks (SCB).

50 DIRECT BIOS CALLS

dient zum Aufrufen von BIOS-Routinen, die in Bank 0 abgelegt sind und auf die ansonsten nur unter schwierigsten Bedingungen zugegriffen werden kann. Nähere Einzelheiten sind in Kapitel 9.4 beschrieben.

59 LOAD OVERLAY

dient zum Laden von RSX-Modulen (siehe Kapitel 9.6) und betrifft nur Programme mit einem RSX-Vorspann (FCB-Adresse in DE). Verschiebbare Module müssen mit der Extension PRL gekennzeichnet sein. Die Module werden nur geladen, aber nicht ausgeführt.

60 CALL RESIDENT SYSTEM EXTENSION

ruft eine RSX-Systemerweiterung auf.

98 FREE BLOCKS

löscht sämtliche Blöcke in allen angeschlossenen Laufwerken, die während den BDOS-Schreiboperationen nur vorübergehend angelegt wurden und nicht im Directory erscheinen sollen. Alle anderen Dateien müssen mit CLOSE geschlossen sein.

99 TRUNCATE FILE

ordnet dem letzten Record einer Datei eine Random-Record-Nummer zu, die im FCB (Adresse in DE) abgelegt ist.

100 SET DIRECTORY-LABEL

zum Einrichten oder Ändern des Directory-Labels (FCB-Adresse in DE).

101 RETURN DIRECTORY LABEL DATA

ermittelt das Byte im Directory-Label (Laufwerk in Register E), das Angaben über Password-Schutz und Timestamps für Dateien enthält (siehe Kapitel 9.2). Das Byte wird in Register A übergeben.

102 READ FILE DATE STAMPS AND PASSWORD MODE

ermittelt Timestamps und Password-Modus für Dateien. Der FCB (Adresse in DE) ist zunächst mit dem Dateinamen zu belegen. Im gleichen FCB befinden sich nach Ausführung der Funktion die betreffenden Angaben (siehe auch Kapitel 9.2).

103 WRITE FILE XFCB

schreibt oder aktualisiert den XFCB (Adresse in DE) für Dateien. Dabei handelt es sich um einen speziellen Directory-Eintrag, der das Password für eine Datei enthält (siehe auch Kapitel 9.2).

104 SET DATE AND TIME

stellt die interne Uhr. Registerpaar DE enthält die Adresse des Feldes, in dem im BCD-Format Datum und Uhrzeit abgelegt sind.

105 GET DATE AND TIME

fragt die interne Uhr ab. Registerpaar DE enthält die Adresse des Feldes, in dem im BCD-Format Datum und Uhrzeit abgelegt sind.

106 SET DEFAULT PASSWORD

ermöglicht die programmäßige Vorgabe eines Passwords (Adresse in DE), bevor auf eine password-geschützte Datei zugegriffen wird.

107 RETURN SERIAL NUMBER

ermittelt die CP/M Plus-Seriennummer. Sie wird in einem 6-Byte-Feld (Adresse in DE) abgelegt.

108 GET/SET PROGRAM RETURN CODE

setzt bzw. ermittelt einen Return-Code für Programme. Diese Funktion wird benötigt, wenn mehrere Programme hintereinander abgearbeitet werden (z.B. Stapelverarbeitung oder BDOS-Funktion 47). Der zu setzende Code ist in Registerpaar DE vorzugeben. Soll der Code ermittelt werden, muß DE den Wert FFFFH enthalten.

109 GET/SET CONSOLE MODE

setzt bzw. ermittelt den Konsolen-Modus. Dieser besteht aus einem 16-Bit-Wert (in DE), der Angaben über die Aktivierung von CTRL-C (Programmabbruch, Warmstart), CTRL-S, CTRL-Q (Scrollen) usw. enthält. Zur Abfrage des Konsolen-Modus ist DE mit dem Wert FFFFH zu belegen. Die Informationen befinden sich dann in Register HL.

110 GET/SET OUTPUT DELIMITER

diese Funktion ermittelt bzw. setzt den Ausgabe-Delimiter für Strings. Standardmäßig ist er mit dem $-Zeichen vorbelegt (siehe Funktion 9). Zur Abfrage ist DE auf FFFFH zu setzen, worauf das Ergebnis in Register A abgelegt wird. Zum Setzen ist das E-Register mit dem ASCII-Code des Delimiters zu belegen.

111 PRINT BLOCK

sendet einen im Console-Control-Block (CCB) (Adresse in DE) definierten String an den Konsolen-Kanal CONOUT:. Byte 0 und 1 des CCB enthält die Adresse, Byte 2 und 3 die Länge des Strings.

112 LIST BLOCK

Wie Aufruf 111, der String wird jedoch über den LST:-Kanal ausgegeben.

152 PARSE FILENAME

formt einen Dateinamen, wie er beispielsweise von der Konsole eingegeben wird, in das interne FCB-Format um. Das Registerpaar DE enthält die Adresse eines Puffers, der mit folgenden Angaben zu belegen ist:

Byte	Beschreibung
0 und 1	Adresse des vorgegebenen Strings
2 und 3	Adresse des FCB

9.4 Mehr über das BIOS

Das BIOS ist der geräteabhängige Teil des CP/M-Systems und führt Funktionen auf einer weit niedrigeren Ebene als das BDOS aus. Die einzelnen BIOS-Routinen können über eine genormte Sprungtabelle aufgerufen werden, die nachfolgend aufgeführt ist. Der Versatz bezieht sich dabei auf den Anfang der Sprungtabelle.

Nr.	Versatz	Funktion
0	00H	Kaltstart
1	03H	Warmstart
2	06H	Konsolenstatusabfrage (A=0, wenn keine Taste gedrückt, sonst A=255)
3	09H	Eingabe eines Zeichens von der Konsole in A
4	0CH	Ausgabe eines Zeichens in C an die Konsole
5	0FH	Ausgabe eines Zeichens in C an den Drucker
6	12H	Ausgabe eines Zeichens in C über den Hilfskanal
7	15H	Eingabe eines Zeichens in A über den Hilfskanal
8	18H	Schreib-/Lesekopf der Floppy auf Spur 0 stellen
9	1BH	Laufwerk anwählen (Nummer 0 bis 15 in C für Laufwerk A bis P). Die DPH-Adresse erscheint in HL.
10	1EH	Spur anwählen (Nummer in BC)
11	21H	Sektor anwählen (Nummer in BC)
12	24H	Adresse (in BC) für Sektorpuffer (DMA) setzen
13	27H	Sektor lesen, muß vorher angewählt sein
14	2AH	Sektor schreiben, muß vorher angewählt sein
15	2DH	Abfrage des Druckerstatus
16	30H	Sektornummer (in BC) anhand von Skewing-Tabelle (Beginn in DE) übersetzen (Ergebnis in HL)
17	33H	Prüft Konsolenstatus
18	36H	Prüft Eingabe-Status für Hilfskanal
19	39H	Prüft Ausgabe-Status für Hilfskanal
20	3CH	Ermittelt Adresse für I/O-Gerätenamen-Tabelle
21	3FH	Initialisiert physikalisches Gerät in Tabelle
22	42H	Ermittelt DPH-Adresse für Laufwerk (in HL)
23	45H	Zählt fortlaufende Sektoren für Read und Write
24	48H	Löscht physikalischen Sektor-Puffer
25	4BH	Blockverschiebung von (DE) zu (HL) Anzahl (BC) im Speicher
26	4EH	Zeit setzen/abfragen
27	51H	*Speicherbank (A) anwählen*
28	54H	Speicherbank (A) für DMA-Operation anwählen
29	57H	Setzt Zielbank (B) und Quellbank (C) für Funktion 25, falls Blockverschiebung zwischen verschiedenen Bänken

Normalerweise kommt man ohne die BIOS-Aufrufe aus, denn viele Funktionen können bereits über die BDOS-Aufrufe ausgeführt werden, die un-

verändert an das BIOS weitergegeben werden. Darüber hinaus ist der Aufruf dieser Funktionen nicht einfach, denn die Sprungtabelle liegt in Bank 0. Mit dem BDOS-Aufruf 50 kann man jedoch direkt auf die BIOS-Sprungtabelle zugreifen, ohne sich um die Bankumschaltung kümmern zu müssen. Dazu ist eine Tabelle im Speicher anzulegen, deren Adresse beim Aufruf der BDOS-Funktion 50 im Register DE abzulegen ist. Hier der Aufbau der Tabelle mit den Registerinhalten, die an das BIOS übergeben werden sollen:

Byte	Beschreibung
0	BIOS-Funktionsnummer (0 bis 29)
1	Inhalt A-Register
2 und 3	Inhalt BC-Register
4 und 5	Inhalt DE-Register
6 und 7	Inhalt HL-Register.

Gibt das BIOS Werte zurück, so befinden sie sich nach der Ausführung der jeweiligen Funktion in den entsprechenden Registern.

Außer dieser Sprungtabelle enthält das BIOS für jedes Laufwerk noch einen Disk-Parameter Block (DPB) und einen Disk-Parameter-Header (DBH). Diese beiden Tabellen liegen in Bank 0, so daß auf sie nicht unmittelbar zugegriffen werden kann. Dennoch wollen wir ihren Aufbau kurz betrachten. Beginnen wir mit der DPH-Tabelle, die folgendermaßen aufgebaut ist:

Byte	Beschreibung
0 und 1	Adresse der Skewing-Übersetzungstabelle (0, da beim CPC nicht verwendet)
2 bis 10	Zwischenspeicher für BDOS, enthält aktuelle Spur-, Sektor- und Recordnummer sowie andere Parameter
11	Flag für Betriebsbereitschaft von Laufwerken
12 und 13	Adresse des Disk-Parameter-Blocks (DPB)
14 und 15	Prüfsummentabelle für Directory
16 und 17	Adresse der Blockbelegungstabelle
18 und 19	Adresse des Buffer-Control-Blocks (BCB) für das Directory
20 und 21	Adresse des BCB für Dateien
22 und 23	Adresse der Directory-Hash-Tabelle
24	Bank-Nummer der Hash-Tabelle

Die Skewing-Übersetzunstabelle enthält die Reihenfolge der Sektoren, in welcher der Zugriff in jeder Spur auf der Diskette stattfindet. Manchmal ist es aus Zeitgründen günstiger, die Aufzeichnung nicht mit dem nächst folgenden Sektor fortzusetzen, sondern zunächst erst zwei oder drei Sektoren zu überspringen.

Hier ein Beispiel: Angenommen, eine Diskette mit 20 Sektoren benötigt für eine Umdrehung 20 ms, d.h. 1 ms, um von einem zum nächsten Sektor zu gelangen. Nachdem nun der Inhalt eines Sektors gelesen wurde, muß er erst in den Speicher übertragen werden, was z.B. 2.5 ms dauert. Ist nun eine Datei in fortlaufenden Sektoren abgelegt, müßte in diesem Fall die Diskette eine ganze Umdrehung durchmachen, um auf den nächsten Sektor zugreifen zu können, denn nach 1 ms ist der Datentransfer noch nicht abgeschlossen. Indem nun jeweils zwei Sektoren ausgelassen werden, kann nicht erst nach 20, sondern bereits nach 3 ms mit dem Zugriff auf den nächsten Sektor begonnen werden, was insgesamt eine wesentlich schnellere Datenübertragung ermöglicht. Die Skewing-Übersetzungstabelle könnte in diesem Fall folgendermaßen aussehen:

```
1,   4,   7,   10,  13,  16,  19,
2,   5,   8,   11,  14,  17,  20,
3,   6,   9,   12,  15,  18
```

Obwohl jeweils zwei Sektoren ausgelassen werden, wird dennoch die gesamte Spur genutzt, denn sie kann jetzt anstelle von 20 Umdrehungen bereits während drei Umdrehungen vollständig gelesen bzw. beschrieben werden.

Die CPC-Floppy enthält allerdings keine Skewing-Tabelle, wie der DPH-Tabelle zu entnehmen ist. Es besteht aber noch die Möglichkeit, die Sektornummern gleich in der Form der Skewing-Tabelle zu formatieren, was dieselbe Wirkung hätte. Ob die Floppy allerdings davon Gebrauch macht, ist nicht bekannt, jedenfalls arbeitet sie ohnehin schon recht schnell.

Als weiteres taucht der Begriff Blockbelegungstabelle auf. Dabei handelt es sich um ein Verzeichnis, das für jeden Block auf der Diskette ein Bit enthält. Ist nun ein Block belegt, ist auch das entsprechende Bit in der Tabelle gesetzt, anderenfalls ist es gelöscht.

Der Buffer-Control-Block (BCB) ist eine weitere Tabelle, die das BDOS zum Blocking/Deblocking von Records benötigt. Darunter versteht man das Herausnehmen bzw. Einfügen von logischen Records (128 Bytes) aus/in physikalische Diskettensektoren, die meist größer als ein Record sind (beim CPC 512 Bytes). Unter CP/M Plus geschieht das zumeist durch das BDOS, während unter CP/M 2.2 diese Aufgabe das BIOS übernimmt.

Die Directory-Hash-Tabellen beschleunigen Suchvorgänge im Directory, und zwar auf eine ähnliche Art, wie es bei einer relativen Dateiverwaltung mit Direktzugriff der Fall ist. Zum Auffinden eines Eintrags ist somit das Directory nicht mehr von Anfang an sequentiell zu lesen und zu durchsuchen.

Das Directory enthält noch eine Prüfsummentabelle, mit deren Hilfe CP/M feststellt, ob ein Diskettenwechsel vorgenommen wurde, um dann entsprechende Maßnahmen einzuleiten.

Kommen wir jetzt zum Disk-Parameter-Block (DPB), der verschiedene Angaben über die physikalische Struktur der Diskette enthält und für jedes einzelne Laufwerk benötigt wird. Die hierin enthaltenen Angaben werden z.B. auch von dem Befehl SHOW [DRIVE] (siehe Kapitel 3) abgefragt. Hier nun der Inhalt des DPB:

Byte	Beschreibung
0 und 1	Records pro Spur
2	Blockverschiebefaktor
3	Blockmaske
4	Extent-Maske
5 und 6	Max. Blockanzahl -1
7 und 8	Max. Anzahl der Directory-Einträge
9 und 10	Allocation 0 und 1
11 und 12	Zu überprüfende Blöcke bei Diskettenwechsel
12 und 14	Anzahl der Systemspuren
15	Physikalischer Record-Verschiebe-Faktor
16	Physikalische Record-Verschiebe-Maske

Hier nun noch eine kurze Erläuterung zu den einzelnen Parametern. Der Blockverschiebefaktor gibt die Zweierpotenz der Anzahl der Records pro Block wieder. Sein Wert ist hier gleich 3, da

2^3 * 128 (Recordgröße) = 1024 (Blockgröße)

ergibt. Aus diesem Wert errechnet sich auch die Blockmaske, in der die drei (Verschiebefaktor) niederwertigsten Bits gesetzt sind:

0000 0111 binär = 7

Allocation 0 und 1 gibt die Anzahl der Directory-Blöcke an, wobei für jeden Block ein Bit gesetzt ist. In diesem Fall steht aber das niederwertigste Bit links und das höchstwertigste rechts. Da der CPC vier solcher Blöcke enthält, erhält man:

1111 0000 0000 0000 binär = C000H

Die beiden letzten Werte des DPB beziehen sich auf die physikalischen Records und sind genauso wie der Blockverschiebefaktor bzw. die Blockmaske zu ermitteln.

9.5 Die Nullseite

Wie wissen bereits, daß der TPA bei Adresse 100H beginnt. Darunter befindet sich noch die Nullseite (Zeropage) von 0000H bis 00FFH, die wichtige Informationen enthält und die wir uns jetzt einmal näher anschauen wollen:

Adresse	Beschreibung
00H - 02H	Sprungvektor für BIOS-Warmstart
03H - 04H	Reserviert
05H - 07H	Sprungvektor für BDOS-Aufrufe
08H - 3AH	Reserviert für RST 1 bis RST 7
3BH - 4FH	Unbenutzt
50H	Gibt Laufwerk an, von dem Programm geladen wurde (0=Laufwerk A, 1=Laufwerk B, usw.)
51H - 52H	Adresse des Password-Feldes 1
53H	Länge des Password-Feldes 1
54H - 55H	Adresse des Password-Feldes 2
56H	Länge des Password-Feldes 2
57H - 5BH	Unbenutzt
5CH - 7BH	Standard-File-Control-Block (FCB) 1
6CH - 7BH	Standard-File-Control-Block (FCB) 2
7CH	Augenblickliche Record-Position in FCB 1 für Direktzugriff
7DH - 7FH	Erweiterung zu 7CH
80H - FFH	Standard-128 Byte-Disketten Ein-/Ausgabepuffer

Die in der Tabelle verwendeten Bezeichnungen 1 und 2 beziehen sich auf bestimmte Befehle, in denen zwei Dateinamen (evtuell mit Password) auftreten. Dazu ein Beispiel:

TYPE ABC.TXT

Hier bezieht sich der Befehl TYPE auf die Nummer 1 und die Datei ABC.TXT auf die Nummer 2.

9.6 Residente System-Erweiterungen (RSX)

Mit einer residenten Systemerweiterung (RSX) können Sie das CP/M Plus-Betriebssystem nach Ihren eigenen Wünschen erweitern oder ändern. Dies geschieht jedoch nur zeitweise während der Ausführung eines Programms (COM-Datei), die ein oder mehrere RSX-Module enthält.

Führt nun ein Programm einen BDOS-Aufruf aus, so fragt jedes RSX dessen Nummer ab, um zu bestimmen, ob er unverändert ausgeführt werden soll oder nicht. Wenn nicht, werden die im RSX-Modul enthaltenen Modifikationen ausgeführt.

RSX-Module können mit GENCOM (siehe Kapitel 10) erstellt und in COM-Dateien eingebunden werden. Zusätzlich erhalten diese Dateien noch einen Vorspann (Header) von 256 Bytes, an dem der CCP erkennt, daß RSX-Module geladen werden sollen. Sie werden dann im oberen TPA-Bereich abgelegt und ausgeführt.

Ein Beispiel für residente Systemerweiterungen ist der Befehl GET (siehe Kapitel 10), der Befehle statt von der Tastatur aus einer Datei entgegennimmt und ausführt. Das hier aktivierte RSX lenkt also die normale Konsoleneingabe auf die betreffende Datei um.

Eine weitere Behandlung der RSXs würde an dieser Stelle zu weit führen, da sie umfangreiche Kenntnisse über den internen Aufbau von CP/M Plus und in der Assemblerprogrammierung erfordert. In diesem Zusammenhang wird auf die am Anfang des Kapitels aufgeführte Dokumentation von Digital Research verwiesen.

9.7 System-Control-Block (SCB)

Beim System-Control-Block handelt es sich um eine Tabelle von 100 Bytes, die im BDOS abgelegt ist. Sie enthält diverse Flags und Parameter für den CCP, das BDOS und andere Systemteile. So sind in den SCB beispielsweise Datum und Uhrzeit, Bildschirm-Parameter und die Aktivierung der verschiedenen Ein-/Ausgabekanäle eingetragen. Einige CP/M Plus-Dienstprogramme und BDOS-Aufrufe greifen ebenfalls auf den SCB zu (z.B. DEVICE), um die darin enthaltenen Parameter zu ändern. Hier nun der Inhalt des SCB im einzelnen:

Offset	Beschreibung
00H - 04H	Reserviert
05H	BDOS-Versions-Nummer
06H - 09H	User Flags
0AH - 0FH	Reserviert
10H - 11H	Program Error Return Code
12H - 19H	Reserviert
1AH	Anzahl der Bildschirmspalten
1BH	Augenblickliche Bildschirmzeile
1CH	Seitenlänge Bildschirm
1DH - 21H	Reserviert
22H - 23H	CONIN-Flag
24H - 25H	CONOUT-Flag
26H - 27H	AUXIN-Flag
28H - 29H	AUXOUT-Flag
2AH - 2BH	LSTOUT-Flag
2CH	Page-Modus
2DH	Reserviert
2EH	Flag für CTRL-H = DEL
2FH	Flag für DEL = CTRL-H
30H - 32H	Reserviert
33H - 34H	Konsolen-Modus (siehe BDOS-Aufruf 109)
35H - 36H	Reserviert
37H	Ausgabe-Delimiter (siehe BDOS-Aufruf 110)
38H	Flag für parallele Druckerausgabe (z.B. mit CTRL-P)
39H - 3BH	Reserviert
3CH - 3DH	Derzeitige DMA-Adresse
3EH	Derzeitiges Bezugslaufwerk
3FH - 43H	Reserviert
44H	Derzeitige Benutzernummer
45H - 49H	Reserviert
4AH	BDOS Multi-Sector Count (s. BDOS-Aufruf 44)
4BH	BDOS Error Mode (s. BDOS-Aufruf 45)
4CH - 4FH	Suchpfad für max. 4 Laufwerke (siehe SETDEF)
50H	Laufwerk für temporäre Dateien
51H	Laufwerk, an dem zuletzt Fehler auftrat
52H - 56H	Reserviert
57H	Flag für Ausgabeart der BDOS-Fehlermeldungen
58H - 59H	Tage seit dem 1. Januar 1978
5AH	Stunden im BCD-Format
5BH	Minuten im BCD-Format
5CH	Sekunden im BCD-Format
5DH - 5EH	Basisadresse des Common-Speicherbereichs
5FH - 63H	Reserviert

Obwohl der SCB von vielen Dienstprogrammen und BDOS-Aufrufen verändert wird, kann dies auch unter Anwendung der BDOS-Funktion 49 geschehen. Dabei ist jedoch zu beachten, daß eine unsachgemäße Änderung einiger spezieller Parameter leicht zum Systemabsturz führen kann.

Neben der Änderung einzelner Parameter dient die BDOS-Funktion 49 auch zu deren Abfrage. Beim Aufruf ist das DE-Register mit der Anfangsadresse der folgenden Tabelle zu belegen, die an einer beliebigen Stelle im TPA abgelegt werden kann:

Byte	Beschreibung
0	BCB-Offset
1	FFH, wenn ein Byte gesetzt werden soll FEH, wenn ein Wort (2-Byte-Parameter oder -Adresse in der Reihenfolge low, high) gesetzt werden soll 00H, wenn ein Wert aus dem SCB abgefragt werden soll
2 und 3	Ergebnis bei Abfrage (1-Byte-Werte in Byte 2, 2-Byte-Werte in Byte 2 und 3)

9.8 Anpassen von CP/M-Software

Abschließend wollen wir noch einen kurzen Blick auf die Anpassung von Software richten. Dabei interessiert uns jetzt nicht die Software, die bereits fertig für den CPC geliefert wird und nur noch einer geringen Anpassung bedarf, sondern die Programme, die Sie von einem anderen CP/M-System auf den CPC oder JOYCE übertragen.

Ist eine direkte Übertragung wegen unterschiedlicher Disketten-Aufzeichnungsformate nicht möglich, bietet sich in erster Linie eine Verbindung über die RS232- oder V24-Schnittstelle an. In viele andere Computer ist sie bereits eingebaut, aber in den CPC noch nicht. Es gibt jedoch verschiedene Firmen, die eine solche Schnittstelle zusammen mit einem passenden Terminalprogramm anbieten, wobei die Datenübertragung meist über den AUXIN:- und AUXOUT:-Kanal stattfindet. Da es aber eine Vielzahl von Terminalprogrammen gibt, wollen wir hier auf eine genaue Beschreibung verzichten, die dem Programm bzw. der Schnittstelle ja ohnehin beiliegt. Sollte ein Terminalprogramm nur unter AMSDOS arbeiten, so ist dies nicht hinderlich, denn wir können auch unter AMSDOS CP/M-Programme auf Diskette speichern.

Ist nun ein CP/M-Programm als COM-Datei auf Diskette abgelegt, so ist es häufig schon lauffähig, ohne daß eine weitere Anpassung nötig wäre. Dabei darf das betreffende Programm aber nicht mehr Arbeitsspeicher benötigen, als im TPA vorhanden ist.

Anpassungen sind zumeist bei den Programmen erforderlich, die spezielle Bildschirm- oder Druckersteuerzeichen verwenden. So verwendet WordStar beispielsweise Steuerzeichen zur Cursorsteuerung und zum Löschen des Bildschirms, ohne die das Programm nicht richtig funktioniert. Im Handbuch des CPC bzw. Ihres Druckers können Sie aber sämtliche benötigten Steuerzeichen nachschlagen. Viele CP/M-Programme werden mit einem speziellen Installationsprogramm geliefert, mit dessen Hilfe man diese Steuerzeichen eingeben kann. Zumindest liegt aber eine genaue Installa-

tionsanweisung mit Patch-Adressen bei, in die man mit Hilfe von SID die entsprechenden Werte einschreiben kann.

Echte Schwierigkeiten dürften aber nur bei solchen Programmen auftreten, die andere Adressen als die Standard-BDOS-Sprungadressen verwenden. In diesem Fall handelt es sich aber nicht mehr um reine CP/M-Software, sondern um ein gerätespezifisches Programm, das nur für einen speziellen Computer geschrieben wurde.

10 CP/M-Befehlsübersicht

In diesem Kapitel finden Sie eine Zusammenfassung sämtlicher CP/M-Plus-Befehle und -Dienstprogramme in alphabetischer Reihenfolge geordnet. Zusätzlich aufgenommen wurden außerdem alle Geräte- spezifischen Programme, die sich auf den mitgelieferten Systemdisketten befinden, sofern sie für das Arbeiten mit CP/M Plus auf dem CPC 6128 bzw. JOYCE von Bedeutung sind.

Im Gegensatz zur Beschreibung in den vorangehenden Kapiteln sind hier sämtliche Unterbefehle (z.B. für ED und PIP) mit aufgeführt und großenteils mit Beispielen versehen. Diese Aufteilung verleiht dem Kapitel den Charakter eines ausführlichen Nachschlagewerkes, aus dem sich besonders der fortgeschrittene CP/M-Anwender die benötigten Informationen schnell beschaffen kann.

Einige Befehle benötigen für bestimmte Optionen die Eingabe von eckigen Klammern. Da die deutsche Ausführung des JOYCE nicht über solche Zeichen verfügt, muß statt dessen "Ä" für "[" (eckige Klammer auf) bzw. "Ü" für "]" (eckige Klammer zu) gesetzt werden.

AMSDOS

Gerätespezifischer CPC-6128-Befehl (Datei AMSDOS.COM)

Beschreibung: AMSDOS schaltet den CPC-6128-Computer vom CP/M-Modus in den BASIC/AMSDOS-Modus zurück, wobei ein Reset durchgeführt wird. Sämtliche Daten im Speicher gehen dabei verloren.

Aufruf: **AMSDOS**

BASIC

Datei BASIC.COM (nur JOYCE)

Beschreibung: Der JOYCE ist nicht, wie der CPC 6128, mit dem CPC-BASIC zu betreiben. Statt dessen ist das Mallard-BASIC im CP/M-Modus separat von Diskette zu laden, das in der Datei BASIC.COM abgelegt ist. Das Mallard-BASIC ist dem M-BASIC sehr ähnlich, nicht aber mit dem CPC-BASIC identisch.

Aufruf: **BASIC**

Zur Rückkehr in den CP/M-Befehlsmodus ist der BASIC-Befehl

SYSTEM

auszuführen.

C10CPM3

Datei C10CPM3.EMS (nur CPC 6128)

Beschreibung: Diese Datei enthält das CP/M Plus-Betriebssystem und muß sich auf jeder Diskette befinden, die CP/M Plus bootet (lädt und initialisiert). Eine solche Diskette muß im Systemformat formatiert sein (siehe DISCKIT3), d.h. einen BOOT-Sektor enthalten, der die C10CPM3.EMS-Datei lädt.

Aufruf: **ICPM**

von BASIC aus, worauf zunächst der BOOT-Sektor ausgeführt und infolgedessen CP/M Plus geladen und initialisiert wird.

COPYSYS

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei COPYSYS.COM)

Beschreibung: COPYSYS überträgt das CP/M-Plus-Betriebssystem auf eine neue Diskette. Dieses Programm steht für den CPC 6128 und JOYCE nicht zur Verfügung, da diese Aufgabe hier von DISCKIT3 bzw. DISCKIT übernommen wird. Auf jeden Fall ist zusätzlich noch die Datei C10CPM3.EMS (CPC 6128) bzw. J12DCPM3.EMS (JOYCE) mit PIP auf die Diskette zu kopieren.

DATE

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei DATE.COM)

Beschreibung: DATE dient zum Stellen und Abfragen der internen Uhr mit Datumsanzeige. Ist DATE einmal ausgeführt, können Dateien mit sogenannten Timestamps (Zeitmarken) versehen werden, sofern das Disketten-Directory mittels INITDIR (siehe unten) dafür eingerichtet wurde.

Aufruf: **DATE**

(ohne Zusatz) fragt das augenblickliche Datum mit Uhrzeit ab.

DATE mm/tt/jj hh:mm:ss:

stellt Datum und Uhrzeit ein.

Beispiel: *DATE 12/05/85 13:34:45*

stellt die interne Uhr auf den 5.12.85, 13 Uhr, 34 Minuten, 45 Sekunden. Dieser Befehl wird aber erst nach anschließendem Drücken einer beliebigen Taste ausgeführt, was ein genaues Stellen der Uhr ermöglicht.

Aufruf: **DATE SET**

ist eine Abwandlung des letzten Befehls und eignet sich besonders zum Einbau in SUBMIT- (Stapel-) oder PROFILE.SUB-Dateien. Somit wird beim Aufruf eines Programms oder beim Laden von CP/M Plus automatisch das aktuelle Datum samt Uhrzeit vom Benutzer angefordert, um die interne Uhr zu stellen.

DATE CONTINOUS

bzw.

DATE C

fragt Datum und Uhrzeit fortlaufend ab, so daß der Eindruck einer Digitaluhr entsteht.

DEVICE

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei DEVICE.COM)

Beschreibung: Weist den logischen Ein-/Ausgabekanälen des CP/M-Plus-Betriebssystems physikalische Geräte zu bzw. fragt die Zuordnung ab.

Aufruf: **DEVICE**

liefert beispielsweise folgende Angaben:

```
Physical Devices:
I=Input,O=Output,S=Serial,X=Xon-Xoff
CRT     NONE  IO      LPT     NONE  O

Current Assignments:
CONIN:  = CRT
CONOUT: = CRT
AUXIN:  = Null Device
AUXOUT: = Null Device
LST:    = LPT

Enter new assignment or hit RETURN
```

Hierin bedeuten:

a) Ein-/Ausgabekanäle:

CONIN: = Konsoleneingabe
CONOUT: = Konsolenausgabe
AUXIN: = Hilfskanal Eingabe
AUXOUT: = Hilfskanal Ausgabe
LST: = Listkanal

b) Physikalische Geräte:

Im Anschluß an die Auflistung erfolgt die Abfrage, ob eine neue Zuordnung gewünscht:

Kanal: = physik. Gerät

ordnet dem angegebenen Ein-/Ausgabekanal ein neues physikalisches Gerät zu.

Beispiel: *LST: = CRT*

lenkt die Druckerausgabe auf den Bildschirm um. Wird keine neue Zuordnung gewünscht, ist das Programm durch Drücken der RETURN-Taste zu verlassen.

Aufruf: DEVICE kann auch auf folgende Arten aufgerufen werden:

DEVICE Kanal: [Option] = Phys. Gerät [Option]

DEVICE Kanal: = NULL

DEVICE Phys. Gerät [Option]

Optionen: **[XON]**

Bezugnahme auf das XON/XOFF-Kommunikations-Protokoll

[NOXON]

Übertragung ohne Anlage eines Protokolls

[Baudrate]

Angabe der Baudrate zwischen 50 und 19200 Baud

Beispiele: *DEVICE CONIN:=CRT*
DEVICE AUXOUT:=MODEM[XON,1200]

DIR / DIRSYS

DIR und DIRSYS ohne Optionen sind residente Standard-CP/M-Plus-Befehle. DIR mit Optionen ist ein nichtresidenter Standard-Befehl (Datei DIR.COM).

Beschreibung: Mit dem DIR-Befehl (ohne Option) wird das gesamte Directory (Inhaltsverzeichnis) einer Diskette, oder ein Teil davon, aufgelistet. Die Dateinamen erscheinen dabei in der Reihenfolge ihrer Ablage, d.h. nicht alphabetisch sortiert. Ausgenommen hiervon sind Systemdateien.

Enthält der DIR-Befehl zusätzlich Optionen, erscheint das Directory in einer erweiterten Form, beispielsweise mit Speicherplatzbelegung der einzelnen Dateien, mit Dateiattributen oder mit Timestamps.

Der DIRSYS-Befehl listet ausschließlich Systemdateien. Für ihn sind keine Optionen zulässig.

Hinweis: Für Dateien, die mit DIR (ohne Option) aufgelistet werden können, ist das Directory-Attribut und für Systemdateien, die mit DIRSYS anzuzeigen sind, das Systemattribut gesetzt. Siehe auch SET.

Aufruf: **DIR (Laufwerk:)[Option]**

bzw.

DIRSYS (Laufwerk:)

oder

DIR (Laufwerk:)Dateiname

bzw.

DIRSYS (Laufwerk:)Dateiname

Dabei kann der Dateiname ein- oder mehrdeutig sein.

Beispiele: *DIR*

listet das Directory ohne Systemdateien des Bezugslaufwerkes und

DIR B:

das Directory ohne Systemdateien von Laufwerk B.

DIR PROBE.TXT

sucht die Datei PROBE.TXT. Ist sie auf der Diskette vorhanden, wird der Name PROBE.TXT ausgegeben, wenn nicht, erscheint der Hinweis

```
NO FILE
```

Der Befehl

*DIR *.COM*

listet sämtliche COM-Dateien und

DIR B.**

diejenigen Dateinamen, die mit dem Buchstaben "B" beginnen.

DIRSYS B:.TXT*

listet sämtliche Systemdateien in Laufwerk B, deren Extension TXT lautet.

Optionen: Optionen können nur für das gesamte Directory, nicht aber für einzelne Dateien erteilt werden. In der nachfolgenden Beschreibung enthält ein einfaches Directory-Listing lediglich die Dateinamen - wie nach DIR ohne Option. Ein erweitertes Listing hat dagegen folgende Angaben:

Dateiname mit Extension
Belegter Speicherplatz in KByte und Records
Dateiattribute (Directory- bzw. Systemattribut, Schreibschutz)
Passwordschutz (falls eingerichtet)
Timestamps (falls eingerichtet)

Aufruf: **DIR (Laufwerk:) [Option]**

Hier die Optionen im einzelnen:

Optionen: **[DATE]**

Listet das erweiterte Directory-Listing, sofern Timestamps zugelassen sind.

[DIR]

Zeigt erweitertes Directory-Listing aller Dateien mit Directory-Attribut.

[DRIVE=ALL]

Zeigt erweitertes Directory-Listing aller angeschlossenen Laufwerke an. Funktioniert nur für Laufwerke, auf die seit dem CP/M-Start bereits mindestens einmal zugegriffen wurde.

[DRIVE=d]

Zeigt erweitertes Directory-Listing für Laufwerk d an.

Beispiele: *DIR [DRIVE=B]*

Zeigt das Directory von Laufwerk B. Auf dem CPC 6128 und JOYCE ist dieser Befehl gleichbedeutend mit

DIR B:[FULL]

Option: **[EXCLUDE] Dateiname 1, Dateiname 2, ...**

zeigt das erweiterte Directory-Listing mit allen Dateien an, außer den angegebenen.

Beispiel: *DIR [EXCLUDE] PIP.COM, ED.COM*

zeigt sämtliche Dateien an, mit Ausnahme von PIP.COM und ED.COM.

Optionen: **[Gn]**

zeigt das erweiterte Directory-Listing für alle Dateien des Benutzerbereichs n an.

[FULL]

zeigt das erweiterte Directory-Listing sämtlicher Dateien an.

[NOPAGE]

zeigt das erweiterte Directory-Listing ohne Unterbrechung durch seitenweise Bildschirm-Ausgabe an.

[NOSORT]

zeigt das erweiterte Directory-Listing an, mit allen Dateinamen in der Reihenfolge ihrer Ablage, also nicht alphabetisch sortiert, wie es bei den übrigen Optionen der Fall ist.

[RO]

zeigt das erweiterte Directory-Listing an, jedoch nur mit Dateien, die schreibgeschützt sind (RO).

[RW]

zeigt das erweiterte Directory-Listing an, jedoch nur mit Dateien, die nicht schreibgeschützt sind (RW).

[SIZE]

zeigt das alphabetisch sortierte einfache Directory-Listing in drei Spalten an, wobei für jede Datei der belegte Speicherplatz auf der Diskette in KByte mit angegeben wird.

[USER=ALL]

zeigt das erweiterte Directory-Listing mit Dateien aller Benutzerbereiche an.

[USER=n]

zeigt das erweiterte Directory-Listing für Benutzerbereich n an.

Beispiel: Es können auch mehrere Optionen kombiniert werden:

DIR [DRIVE=B,NOSORT,NOPAGE]

Hier werden sämtliche Dateien in Laufwerk B angezeigt, allerdings nicht alphabetisch sortiert und ohne Unterbrechung, wenn eine Bildschirmseite vollgeschrieben ist.

DISCKIT / DISCKIT3 (JOYCE / CPC 6128)

gerätespezifischer CP/M-Plus-Befehl (Datei DISCKIT.COM bzw. DISCKIT3.COM)

Beschreibung: DISCKIT bzw. DISCKIT3 ist ein komfortables Dienstprogramm, das Disketten mit einem oder zwei Laufwerken kopiert, formatiert oder verifiziert.

Die Bedienung ist menügesteuert, wobei die jeweiligen Menüpunkte mit den Funktionstasten des Zifferntastenblocks auszuwählen sind. Weitere Informationen zu DISCKIT bzw. DISCKIT3 befinden sich in Kapitel 2.3.

Aufruf: **DISCKIT** (JOYCE)

bzw.

DISCKIT3 (CPC 6128)

DISCKIT bzw. DISCKIT3 stellt automatisch fest, ob ein oder zwei Laufwerke angeschlossen sind. Bei einem Laufwerk ist zum Kopieren und Verifizieren jedoch ein mehrfacher Diskettenwechsel nötig.

DUMP

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei DUMP.COM)

Beschreibung: DUMP liest Dateien von Diskette und listet sie in hexadezimaler Form auf dem Bildschirm, wobei darstellbare ASCII-Zeichen angegeben werden.

Aufruf: **DUMP (Laufwerk:) Dateiname**

Beispiel: *DUMP MUSTER.COM*

listet die Datei MUSTER.COM im Hexcode mit darstellbaren ASCII-Zeichen.

ED

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei ED.COM)

Beschreibung: ED ist ein Texteditorprogramm, das sich zum Erstellen und Bearbeiten von Textdateien jeglicher Art eignet. Dabei werden die Texte in einen Textpuffer geschrieben oder müssen nach dem Aufruf von ED dorthin von Diskette übertragen werden.

Bei ED handelt es sich um einen einfachen Editor mit geringem Bedienungskomfort, der weit hinter demjenigen von professionellen Texverarbeitungsprogrammen, wie z.B. WordStar, zurücksteht.

Anwendung: Texte schreiben, Textzeilen löschen, Textzeilen einfügen, Begriffe im Text suchen und austauschen

Aufruf: **ED (Laufwerk:)Dateiname**

Dabei ist der Name der Datei anzugeben, die neu angelegt oder bearbeitet werden soll. Nach dem Aufruf erscheint

```
NEW FILE
: *
```

wenn die Datei neu angelegt wird oder nur

```
: *
```

wenn die Datei bereits vorhanden ist und weiter bearbeitet werden soll. Hinter dem Doppelpunkt und dem Sternchen sind dann die ED-Befehle einzugeben.

Beispiele: *ED PROBE.TXT*

ruft die Datei PROBE.TXT auf oder legt sie neu an. Mit

ED B:BRIEF.TXT

geschieht dasselbe mit der Datei BRIEF.TXT, jedoch in Laufwerk B.

ED-Befehle:

Bei den nachfolgend aufgeführten Befehlen tritt häufig der Wert "n" auf. Für "n" steht dabei die jeweilige Anzahl von Zeichen, Zeilen usw. Für "n"

kann man aber auch ein Ziffernkreuz (#) angegeben, das für die größtmögliche Anzahl von "n" (max. 65535) steht. ED berücksichigt dabei immer die Gesamtzahl von Zeichen oder Zeilen in einer Datei oder im Puffer.

Ist für einen Befehl +/-n angegeben, so kann bei positiven Werten das Pluszeichen auch entfallen.

Befehl: **n:**

setzt den Textzeiger an den Anfang der n-ten Zeile.

Beispiel: *25:*

Der Zeiger steht jetzt am Anfang der 25. Zeile im Textpuffer.

Befehl: **nA**

Dieser Befehl liest n Zeilen aus der Textdatei in den Textpuffer zur Bearbeitung ein.

Beispiele: *30A*

liest 30 Zeilen in den Puffer und

#A

sämtliche Zeilen der Textdatei. Der letzte Befehl sollte immer bei nicht zu umfangreichen Dateien Anwendung finden, wenn sie einschließlich möglicher Erweiterungen in den Puffer hineinpassen.

Befehl: **+/-B**

setzt den Textzeiger an den Pufferanfang (B oder +B) oder an das Pufferende (-B).

Befehl: **+/-nC**

Setzt den Textzeiger von seiner augenblicklichen Position +n Zeichen vor oder -n Zeichen zurück.

Beispiele: *-20C* (Textzeiger um 20 Zeichen zurücksetzen)

154C (Textzeiger um 154 Zeichen vorrücken)

Befehl: **+/-nD**

löscht von der augenblicklichen Position des Textzeigers die +n folgenden oder die -n davorliegenden Zeichen.

Beispiele: Angenommen, der Pufferzeiger steht auf dem 1000-sten Zeichen. Dann werden mit

20D

die Zeichen 1000 bis 1019 und mit

-12D

die Zeichen 988 bis 999 gelöscht.

Befehl: **E**

Beendigung der Textverarbeitung und Rückkehr zu CP/M. Der Text wird aus dem Puffer in eine Zwischendatei (Extension $$$) geschrieben, diese dann in den beim Aufruf von ED angegebenen Dateinamen umbenannt und eine eventuell vorher vorhandene Datei in eine BAK-Datei umbenannt.

Beispiel: Die Bearbeitung der Textdatei BRIEF.TXT, die sich schon vor dem ED-Aufruf auf Diskette befand, soll abgeschlossen werden. Spätestens bei der Ausführung des E-Befehls wird eine Zwischendatei BRIEF.$$$ angelegt, in die der Pufferinhalt abgelegt wird. Im Anschluß daran wird die ursprüngliche Datei BRIEF.TXT in BRIEF.BAK und die Zwischendatei BRIEF.BAK in BRIEF.TXT umbenannt.

Befehl: **nF"Begriff"**

Dieser ED-Befehl sucht das n-te Auftreten des vorgegebenen "Begriffs" ab der augenblicklichen Textzeigerposition. Wird für "n" keine Zahl angegeben, wird n=1 angesetzt. Der hinter "n" stehende Buchstabe "f" sollte normalerweise klein geschrieben werden, da bei einem großen "F" der "Begriff" in Großschrift umgewandelt und in dieser Form nach ihm gesucht wird.

Sollen dem Suchbefehl noch weitere Befehle folgen, muß dieser zusätzlich mit einem CTRL-Z abgeschlossen werden.

Wurde der gesuchte "Begriff" gefunden, steht der Textzeiger auf dem dem "Begriff" folgenden Zeichen.

Beispiele: *3fHurra!*

sucht das dritte Auftreten des Begriffs "Hurra!". Wenn der Befehl

FHurra!

lautet, sucht ED nach dem ersten Begriff in der Form von "HURRA!".

Befehl: **H**

schließt die Textbearbeitung ab. Dieser Befehl hat dieselbe Funktion wie "E", kehrt jedoch nicht zu CP/M zurück, sondern ruft die bearbeitete Textdatei erneut auf, so daß sie weiterbearbeitet werden kann.

Befehl: **I**

Dieser Befehl kennzeichnet den Einfügemodus. Zunächst ist der Textzeiger auf die einzufügende Zeile zu stellen und dann der I-Befehl zu erteilen, worauf neue Zeilen eingefügt werden können.

Ist "I" groß geschrieben, werden alle eingefügten Texte in Großschrift umgewandelt. Bei kleinem "i" ist Groß- und Kleinschrift möglich.

Beispiele: In einen Text sollen zwischen den derzeitigen Zeilen 4 und 5 weitere Zeilen eingefügt werden. Dazu ist zunächst der Textzeiger auf Zeile 5 zu setzen und dann i einzugeben:

5:
i

Die erste eingefügte Zeile ist somit die neue Zeile 5, während die Nummern der alten Zeilen, die 5 oder größer waren, um eins nach oben rücken. Dieser Vorgang wiederholt sich bei jeder zusätzlich eingefügten Zeile.

Da "i" klein angegeben wurde, können die eingefügten Zeilen in Groß- und Kleinschrift geschrieben werden.

-B
i

In diesem Fall wird der Zeiger an das Textende gesetzt, so daß neuer Text angefügt werden kann.

I"Begriff"^Z

Dieser I-Befehl fügt den "Begriff" an der Position des Textzeigers ein. Auch hier gilt, daß bei einem großen "I" sämtliche Zeichen des einzufügenden Begriffs in Großschrift umgewandelt werden, während sonst ein kleines "i" angegeben werden muß.

iTasse^Z

fügt das Wort "Tasse" an der Textzeigerposition ein.

Befehl: **ni"Begriff 1"^Z"Begriff 2"^Z"Begriff 3"^Z**

Zunächst wird "Begriff 1" gesucht, "Begriff 2" an "Begriff 1" angehängt und sämtliche Zeichen bis zum Beginn von "Begriff 3" gelöscht.

Beispiel: *5iRose^ZGeranie^ZTulpe^Z*

Nach dem 5. Auftreten des Wortes "Rose" wird an dieses "Geranie" angefügt und sämtliche Zeichen zwischen "Rose" und "Tulpe" gelöscht.

Befehl: **+/-nK**

Dieser Befehl löscht n Zeilen vor bzw. hinter dem Textzeiger.

Beispiele: Angenommen, der Textzeiger steht am Anfang von Zeile 12. Im Falle von

-3K

werden die Zeilen 9, 10, 11, bzw. im Falle von

4K

die Zeilen 12, 13, 14 und 15 gelöscht.

Befehl: **+/-nL**

verschiebt den Textzeiger um n Zeilen vor bzw. zurück.

Beispiele: *3L*

setzt den Zeiger um 3 Zeilen vor und

-5L

setzt ihn um 5 Zeilen zurück.

Befehl: **nM"Begriff"^Z**

führt den angegebenen ED-"Befehl" n-mal hintereinander aus. Ist n gleich 0 oder 1, wird der "Befehl" so lange wiederholt ausgeführt, bis die Datei zu Ende ist oder ein Fehler auftritt.

Beispiel: *5M1K^Z*

löscht fünfmal hintereinander die nächste Zeile und ersetzt somit den Befehl 5K.

Befehl: **nN"Begriff"^Z**

sucht das n-te Auftreten des "Begriffs". Funktioniert wie der F-Befehl mit der Ausnahme, daß bei Nichtauffinden des "Begriffs" weiterer Text aus der Datei in den Textpuffer nachgeladen und ebenfalls durchsucht wird.

Befehl: **O**

Abbruch der Textbearbeitung bei gleichzeitigem Rücksprung in die Originaldatei, wobei keine Änderungen übernommen werden. Zunächst erfolgt jedoch noch die Sicherheitsabfrage

```
O·(Y/N)?
```

die mit Y (Ja) oder N (Nein) zu beantworten ist.

Befehl: **+/-nP**

Dieser Befehl zeigt "n" Textblöcke zu je 24 Zeilen auf dem Bildschirm an, und zwar vor oder hinter der aktuellen Textzeigerposition. Ist n gleich 0, beginnt die Ausgabe mit der Zeile, auf die der Textzeiger gerade zeigt.

Befehl: **Q**

Abbruch der Textbearbeitung und Rückkehr zu CP/M. Im Gegensatz zum E-Befehl wird die bearbeitete Datei aber nicht gespeichert.

Befehl **R"Dateiname"**

dient zum Einfügen eines Textblockes, der aus einer Datei mit der Extension LIB in den Textspeicher gelesen wird. Der Textzeiger steht anschließend am Ende des eingefügten Textblocks. Statt dieser Datei kann auch die Zwischendatei X$$$$$$$.LIB gelesen werden.

Beispiel: *REINFUEG.LIB*

liest einen Textblock, der in der Datei EINFUEG.LIB abgelegt ist, in den Textpuffer.

Befehl: **nS"Begriff 1"^Z"Begriff 2"^Z**

sucht und ersetzt n-mal den "Begriff 1" durch den "Begriff 2". Ist für "n" kein Wert angegeben, findet dieser Vorgang nur einmal statt.

Ist "S" groß geschrieben, findet die Suche und der Austausch nur in Großbuchstaben statt, bei kleinem "s" dagegen in Groß- und Kleinschrift.

Beispiel: *Apfel^ZBirne^Z*

sucht ab der gegenwärtigen Textzeigerposition das Wort "Apfel" und ersetzt dieses durch "Birne".

Befehl: **+/-nT**

Listet n Zeilen vor bzw. hinter der aktuellen Textzeigerposition.

Beispiele: Angenommen, der Textzeiger zeigt auf Zeile 20. Durch

4T

werden die Zeilen 20 bis 24 und durch

-3T

die Zeilen 18 bis 20 gelistet.

Befehl: **+/-U**

Infolge von "+U" oder "U" wird sämtlicher Text im Puffer in Großschrift umgewandelt. "-U" deaktiviert diesen Vorgang wieder.

Befehl: **+/-V**

"-V" schaltet die Anzeige der Zeilennummern ab und "+V" schaltet sie wieder ein. "0V" zeigt die Größe des noch freien und des gesamten Pufferspeichers in Bytes an.

Beispiel: Beim CPC erscheint infolge von *0V*:

```
24863/25015
```

d.h. der Textpuffer umfaßt insgesamt 25015 Byte, von denen 24863 Byte noch nicht belegt sind.

Befehl: **nW**

Vom Textzeiger an werden "n" Zeilen in die Zwischendatei mit dem Namen der zu bearbeitenden Datei und der Extension $$$ geschrieben. Ohne Zahlenangabe wird n=1 angenommen. Die Zwischendatei kann mit dem R-Befehl wieder gelesen werden.

Befehl: **nX**

Vom Textzeiger an werden "n" Zeilen in die Zwischendatei X$$$$$$$.LIB geschrieben, die mit dem R-Befehl wieder gelesen werden kann. Ist n=0, wird die Zwischendatei gelöscht.

Befehl: **nZ**

Vor der Ausführung eines ED-Befehls wird eine Pause von "n" Zeiteinheiten eingelegt.

Steuerzeichen: **CTRL-C**

Warmstart und Rückkehr zu CP/M

CTRL-E

Fortsetzung der Befehlszeile am Anfang der nächsten Bildschirmzeile

CTRL-H

Löschen des Zeichens links vom Cursor, dabei Cursor um eine Stelle nach links (Backspace)

CTRL-I

Cursor um eine Tabulatorbreite (7 Zeichen) vorrücken

CTRL-J

entspricht Drücken der RETURN-Taste

CTRL-L

dient zum Austauschen von RETURN-Zeichen bei Suchvorgängen

CTRL-M

entspricht Drücken der RETURN-Taste (Carriage Return)

CTRL-R

Erneute Anzeige der Befehlszeile nach Fehlerkorrektur

CTRL-U

Befehlszeile löschen, Cursor in nächste Zeile

CTRL-X

Befehlszeile löschen, Cursor zurück an Zeilenanfang

CTRL-Z

dient für Such- und Austauschoperationen sowie zur Beendigung des Einfügemodus (I)

ERA

Teils residenter, teils nichtresidenter Standard-CP/M-Plus-Befehl (Datei ERASE.COM).

Beschreibung: ERA dient zum Löschen einzelner Dateien oder Dateigruppen. Davon ausgenommen sind schreibgeschützte (RO) Dateien. Bei Angabe eines mehrdeutigen Dateinamens erfolgt vor dem eigentlichen Löschen noch eine Sicherheitsabfrage.

Aufruf: **ERA (Laufwerk:)Dateiname**

Dabei kann der Dateiname ein- oder mehrdeutig sein.

Beispiele: *ERA PROBE.TXT*

löscht die Datei PROBE.TXT im Bezugslaufwerk und

ERA B:PROBE.TXT

löscht die gleiche Datei, wenn sie in Laufwerk B abgelegt ist.

*ERA *.DAT*

löscht alle Dateien mit der Extension DAT und

*ERA *.**

löscht sämtliche Dateien auf der Diskette.

Da in den letzten beiden Beispielen mehrdeutige Dateinamen angegeben sind, erfolgt vor der Ausführung noch die Sicherheitsabfrage

```
ALL (Y/N)?
```

Wird hier "Y" für "Ja" eingegeben, werden die Dateien gelöscht, ansonsten erscheint wieder das Anforderungszeichen.

Wird die angegebene Datei oder Dateigruppe nicht gefunden, erscheint die Meldung:

```
NO FILE
```

Dateien können auch gelöscht werden, indem zunächst die Datei ERASE.COM ausgeführt wird, die dann separat den Dateinamen anfordert.

GENCOM

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei GENCOM.COM)

Beschreibung: GENCOM versieht COM-Dateien mit sogenannten Systemerweiterungen (RSX), die hinten angefügt werden. Solche Systemerweiterungen dienen beispielsweise dazu, die Ein- und Ausgabe von der Konsole über eine Datei abzuwickeln, wie es z.B. bei GET der Fall ist (siehe unten). Am Anfang der neuen COM-Datei richtet GENCOM außerdem einen Vorspann ein, der angibt, daß ein RSX geladen und ausgeführt werden soll.

Für die Erstellung der RSX-Module sind genaue Kenntnisse der CP/M-Plus-BDOS-Aufrufe sowie umfangreiche Assembler-Programmierkenntnisse erforderlich. Weitere Einzelheiten sind der CP/M-Plus-Dokumentation von Digital Research zu entnehmen.

Aufruf: **GENCOM Datein. COM-Dat. Datein. RSX-Mod. [Optionen]**

Optionen: **[LOADER]**

Setzt ein Bit zur Kennzeichnung, damit das Ladeprogramm aktiv bleibt.

[NULL]

Diese Option gibt an, daß nur RSX-Module Verwendung finden, so daß eine Dummy-COM-Datei (Leerdatei, ohne Inhalt) erzeugt wird.

[SCB=(Offset, Wert)]

Bildet SCB-Block.

Beispiele: *GENCOM NEUPROG MODUL1 MODUL2*

Erzeugt die Datei NEUPROG.COM mit angefügten RSX-Modulen MODUL1 und MODUL2.

GENCOM MODUL1 MODUL2 [NULL]

Richtet eine COM-Leerdatei unter dem Namen MODUL1.COM ein, die die RSX-Module MODUL1 und MODUL2 enthält.

GENCOM NEUPROG

Entfernt aus der Datei NEUPROG die RSX-Module und den Vorspann, so daß wieder eine normale COM-Datei entsteht.

GET

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei GET.COM)

Beschreibung: Benötigt ein auszuführendes CP/M-Programm Eingaben von der Tastatur, so können diese statt dessen mit Hilfe des GET-Befehls aus einer Datei gelesen werden. Benötigt das Programm mehr Eingaben, als in der Datei enthalten sind, kehrt das System nach Abarbeitung der Datei wieder zur Tastatureingabe zurück.

Aufruf: **GET CONSOLE INPUT FROM FILE (Dateiname) [Option]**

(Aufruf des auszuführenden Programms)

Beispiel: *GET CONSOLE INPUT FROM FILE EINGABE.DAT*

BILANZ

liest sämtliche Terminal-Eingaben, die für das CP/M-Programm BILANZ.COM erforderlich sind, aus der Datei EINGABE.DAT. Der Aufruf des Programms BILANZ muß separat über die Tastatur erfolgen.

Die Eingabe über die Datei wird abgebrochen, wenn diese entweder abgearbeitet ist oder folgenden Befehl enthält:

```
GET CONSOLE
```

Optionen: **[ECHO]**

Zeigt sämtliche Eingaben über die Datei auf dem Bildschirm an. Diese Option ist standardmäßig voreingestellt.

[NO ECHO]

Zeigt die Eingaben über die Datei nicht an.

[SYSTEM]

Die Datei übergibt nicht nur Eingaben an das auszuführende Programm, sondern ruft dieses Programm zusätzlich selbst auf. Die entsprechende Befehlszeile muß also in der Datei enthalten sein.

HELP

Nichtresidenter Standard-CP/M-Plus-Befehl (Dateien HELP.COM, HELP.HLP)

Beschreibung: HELP enthält Kurzinformationen über die einzelnen CP/M-Plus-Befehle und einige andere Stichwörter. Zu jedem Hauptbegriff können Unterbegriffe, von jedem Unterbegriff weitere Unterbegriffe usw. aufgerufen werden. Insgesamt sind maximal neun Unterbegriffsebenen zulässig.

Aufruf: **HELP**

läßt eine Auflistung der Hauptbegriffe erscheinen, die dann aufzurufen sind. Es folgt die Erläuterung und die Angabe weiterer Unterbegriffe, die allerdings mit vorangestelltem Punkt einzugeben sind. HELP kann auch direkt unter Angabe des Hauptbegriffs und untergeordneter Begriffe aufgerufen werden.

Beispiele: *HELP DIR BUILT-IN EXAMPLES*

ruft HELP mit dem Hauptbegriff DIR, dem Unterbegriff BUILT-IN und dem untergeordneten Unterbegriff EXAMPLES auf, deren Erläuterungen dann angezeigt werden.

Man könnte hier auch schrittweise vorgehen, indem man zunächst

HELP

dann den Hauptbegriff

DIR

anschließend den Unterbegriff

.BUILT-IN (mit Punkt!)

und schließlich den untergeordneten Unterbegriff

.EXAMPLES (mit Punkt!)

aufruft.

HEXCOM

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei HEXCOM.COM)

Beschreibung: HEXCOM liest eine HEX-Datei im Intel-Hex-Format, die durch den Assembler MAC erzeugt wurde, und formt sie in eine ausführbare COM-Datei um, die auf Diskette geschrieben wird.

Aufruf: **HEXCOM (Laufwerk:)Dateiname der HEX-Datei**

Der Dateiname muß dabei ohne Extension eingegeben werden.

Beispiel: *HEXCOM MUSTER*

liest die Intel-Hex-Datei MUSTER.HEX und wandelt sie in die Datei MUSTER.COM um, die von CP/M aus ausführbar ist.

INITDIR

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei INITDIR.COM)

Beschreibung: INITDIR führt eine Neuordnung des Directories durch, damit es Timestamps (Zeitmarken) für einzelne Dateien aufnehmen kann. Dabei wird für jeweils drei Dateien ein Directory-Eintrag reserviert, wodurch die Anzahl der frei verfügbaren Einträge um ein Viertel abnimmt.

Aufruf: **INITDIR (Laufwerk:)**

Wird INITDIR für eine Diskette angewandt, deren Directory bereits Timestamps enthält, so werden sämtliche Timestamps gelöscht.

J12DCPM3

Datei J12DCPM3.EMS (nur JOYCE)

Beschreibung: Diese Datei enthält das CP/M-Plus-Betriebssystem und muß sich auf jeder Diskette befinden, die CP/M-Plus bootet (lädt und initialisiert).

Aufruf: Nach dem Einschalten des JOYCE ist die Diskette mit J12DCPM3.EMS in das Laufwerk zu legen. CP/M Plus wird dann automatisch geladen und initialisiert.

Dasselbe geschieht bei bereits initialisiertem CP/M-System oder Locoscript durch gleichzeitiges Drücken der Tasten SHIFT, EXTRA und EXIT, wodurch ein Reset ausgelöst wird.

LANGUAGE

Gerätespezifischer CP/M-Plus-Befehl (Datei LANGUAGE.COM)

Beschreibung: Mit LANGUAGE kann man acht verschiedene internationale Zeichensätze auf dem CPC 6128 bzw. JOYCE installieren.

Aufruf: **LANGUAGE n**

"n" gibt dabei die Nummer des Zeichensatzes an.

n	Zeichensatz
0	Amerikanisch
1	Französisch
2	Deutsch
3	Englisch
4	Dänisch
5	Schwedisch
6	Italienisch
7	Spanisch

Beispiel: *LANGUAGE 2*

stellt den deutschen Zeichensatz ein. Dabei handelt es sich lediglich um die Bildschirmzeichen, nicht aber um eine Tastaturanpassung.

LIB

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei LIB.COM)

Beschreibung: Der LIB-Befehl dient zum Arbeiten mit Programmbibliotheken, die verschiedene Module enthalten. Neben dem Einrichten einer Bibliothek können einzelne Module eingefügt, gelöscht, ersetzt oder aufgerufen werden. Dabei handelt es sich um verschiebbare Objektmodule im REL-Format, die z.B. mit dem Makroassembler RMAC erstellt wurden. Sie können mit dem LINK-Befehl verbunden und in eine ausführbare COM-Datei umgewandelt werden.

Aufruf: **LIB Dateiname[Option] = Dateiname (Parameter)**

Optionen: **[I]**

Legt eine indizierte Bibliotheksdatei mit der Extension IRL an, die von LINK schneller zu durchsuchen ist als eine nicht-indizierte.

[M]

Anzeige der einzelnen Modulnamen.

[P]

Anzeige der Modulnamen und der globalen Variablen für die neue Bibliotheksdatei.

[D]

Anzeige der Objektmodule im ASCII-Format.

Parameter: **Delete <Modul=>**

Modul löschen

Replace <Modul=Dateiname.REL>

Modul ersetzen/austauschen

Select <Modul>

Modul auswählen

Beispiele: *LIB MUSTER[P]*

Anzeige sämtlicher Module der Bibliothek MUSTER.REL.

LIB BIBLTHEK = MODUL1, MODUL 2

bildet die Bibliothek BIBLTHEK aus den Modulen MODUL 1 und MODUL 2.

LINK

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei LINK.COM)

Beschreibung: LINK verbindet einzeln durch den Assembler RMAC erstellte Module im REL-Format und wandelt sie in eine ausführbare COM-Datei um.

Aufruf: **LINK (Laufwerk:) Dateiname neu [Option] Dateiname 1 [Option], Dateiname 2 [Option] ...**

Ist "Dateiname neu" ausgelassen, erhält die COM-Datei den Dateinamen 1.

Beispiele: *LINK PROBE*

liest die von RMAC erzeugte Modul-Datei PROBE.REL und formt sie in die ausführbare Datei PROBE.COM um.

LINK GESAMT=DATEI1,DATEI2,DATEI3

verbindet die drei REL-Dateien DATEI1, DATEI2 und DATEI3 und bildet die ausführbare Datei GESAMT.COM.

LINK DATEI1,DATEI2,DATEI3

verbindet die REL-Dateien ebenfalls, wobei die ausführbare COM-Datei jedoch die Bezeichnung DATEI1.COM erhält.

Optionen: **[A]**

Anlage temporärer Dateien macht weiteren Speicher frei.

[B]

dient zur Erstellung des CP/M-Plus-BIOS.

[Dhhhh]

setzt Speicheranfang für gemeinsamen Bereich fest.

[Gn]

Setzt Anfangsadresse auf Label "n".

[Lhhhh]

Angabe einer anderen Ladeadresse als 100H.

[NL]

Symboltabelle wird nicht angezeigt.

[NR]

Keine Datei für Symboltabelle.

[OC]

Ausgabe als Standard-COM-Datei.

[Phhhh]

Änderung des Programmspeicheranfangs auf Adresse hhhh.

[Q]

Auflistung der Symbole mit vorangestelltem Fragezeichen.

[S]

Der angegebene Dateiname bezieht sich auf eine Bibliothek, die durchsucht wird.

[$Cd]

Ausgabe der Terminalmeldungen, wobei "d" für folgendes Gerät steht:

X = Terminal
Y = Drucker
Z = keine Ausgabe

[$Id]

Ablage von Zwischendateien in Laufwerk d.

[$Ld]

Bibliotheksdatei in Laufwerk d.

[$0d]

Ablage der Objektdatei in Laufwerk d.

[$Sd]

Ablage der Symboldatei in Laufwerk d.

MAC

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei MAC.COM)

Beschreibung: MAC ist ein 8080-Assembler, der Quelltexte übersetzt, die in einer Datei mit der Extension ASM abgelegt sind. Die Übersetzung findet dabei nicht direkt in den ausführbaren Maschinen- oder Objektcode statt, sondern zunächst in den Intel-Hex-Code, der als ASCII-Text in einer Datei mit der Extension HEX abgelegt wird. Dieser Code wird erst mit HEXCOM in eine ausführbare COM-Datei umgewandelt.

Neben der Datei im Intel-Hex-Format erzeugt MAC noch eine weitere Datei mit der Extension PRN, welche das komplette Assemblerlisting zur Dokumentation enthält, sowie eine Datei mit der Extension SYM, in der die Symboltabelle abgelegt ist.

Fehlerhafte Quelltextzeilen werden nicht mit assembliert und auf dem Bildschirm ausgegeben.

Aufruf: **MAC (Laufwerk:)ASM-Datei $Option**

Die ASM-Datei darf nur mit Einfachnamen, also ohne die Extension ASM angegeben werden.

Beispiel: **MAC MUSTER**

übersetzt die Quelltextdatei MUSTER.ASM und erzeugt die Datei MUSTER.HEX im Intel-Hex-Format, die Datei MUSTER.PRN, die das Assembler-PRN-Listing enthält, sowie die Datei MUSTER.SYM mit der Symboltabelle.

Optionen: **Achtung!** In den nachfolgend beschriebenen Optionen ist häufig eine Laufwerksbezeichnung "d" angegeben. Für "d" können jedoch nicht nur die Laufwerke A, B, ...P angegeben werden, sondern statt dessen auch externe Geräte. So steht "X" für Bildschirm und "P" für Drucker. Bei Angabe von "Z" wird keine Datei erzeugt bzw. ausgegeben.

Ad

Quell-Laufwerk d für ASM-Datei

Hd

Ziel-Laufwerk d für HEX-Datei

Ld

Quell-Laufwerk für Bibliotheks-(LIB)-Dateien, die durch den Befehl MACLIB aufgerufen werden.

Pd

Ziel-Laufwerk d für PRN-Datei

Sd

Zieldatei d für SYM-Datei (Symboltabelle)

+/-L

listet/unterdrückt aus Makrobibliothek gelesene Zeilen

+/-M

listet/unterdrückt Makrozeilen während der Assemblierung

***M**

listet bei Einsetzen des Makros nur Maschinencode

+/-Q

listet/unterdrückt alle LOCAL-Symbole in der Symboltabelle

+/-S

Anfügen/Nicht-Anfügen der SYM- an die PRN-Datei

+/-I

erstellt/unterdrückt beim ersten Durchlauf die Liste zur Makro-Fehlersuche

Beispiel: *MAC MUSTER $AA PB HA SP*

Assembliert die ASM-Datei MUSTER, die in Laufwerk A abgelegt ist. Die PRN-Datei wird dabei in Laufwerk B und die HEX-Datei in Laufwerk A angelegt. Die Symboldatei dagegen wird ausgedruckt.

PALETTE

Gerätespezifischer CP/M-Plus-Befehl (Datei PALETTE.COM)

Beschreibung: PALETTE dient zum Einstellen der Bildschirmfarben unter CP/M.

Aufruf: **PALETTE Hintergrundfarbe, Schriftfarbe**

Beispiel: *PALETTE 46,2*

erzeugt blaue Schrift auf einem rosa Hintergrund.

Hier die Farbtabelle:

Nr.	Farbe
0	Schwarz
2	Blau
3	Hellblau
8	Rot
10	Magenta
11	Hellviolett
12	Hellrot
14	Purpur
15	Helles Magenta
32	Grün
34	Blaugrün
35	Himmelblau
40	Gelb
42	Weiß
43	Pastellblau
44	Orange
46	Rosa
47	Pastellmagenta
48	Hellgrün
50	Seegrün
51	Helles Blaugrün
56	Limonengrün
58	Pastellgrün
59	Pastellblaugrün
60	Hellgelb
62	Pastellgelb
63	Leuchtendweiß

PAPER (nur JOYCE)

Gerätespezifischer CP/M-Plus-Befehl (Datei PAPER.COM)

Beschreibung: Mit PAPER lassen sich verschiedene Drucker-Parameter definieren, wie z.B. Zeilen pro Seite, Anzahl der freien Zeilen am Seitenanfang und -ende, Zeilenabstand und dergleichen. Neben dem mitgelieferten Drucker lassen sich mit PAPER auch die meisten Epson-kompatiblen Drucker einstellen. Weitere Einzelheiten sind im Handbuch beschrieben.

PATCH

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei PATCH.COM)

Beschreibung: PATCH nimmt Änderungen im CP/M-Betriebssystem oder in einer COM-Datei vor. Dabei sind Patch-Nummern zwischen 0 und 31 vorzugeben.

Beispiel: *PATCH SHOW 2*

ändert die Datei SHOW.COM mit Patch Nr. 2.

PIP

Nichtresidenter Standard-CP/M-Plus-Befehl (Datei PIP.COM)

Beschreibung: PIP ist ein universelles Datei-Übertragungsprogramm von Diskette zu Diskette einerseits und von Diskette zu Peripheriegeräten (oder umgekehrt) über die CP/M-Ein-/Ausgabekanäle andererseits.

Grundsätzlich können mit PIP alle Arten von Dateien übertragen werden, also ohne Rücksicht auf ihren Inhalt oder ihre Beschaffenheit. Jedoch sind viele Befehle nur für ASCII-Text-Dateien, die mit einem CTRL-Z (1AH) abschließen, geeignet.

Anwendung: Kopieren einzelner Dateien
Kopieren ganzer Disketteninhalte
Dateien aneinanderreihen
Listen von ASCII-Dateien auf Bildschirm oder Drucker
Verändern von Dateien während des Kopiervorgangs
Auszüge von Textdateien herstellen

Aufruf: **PIP (Befehlsfolge)**

oder

PIP
* **(Befehlsfolge 1)**
* **(Befehlsfolge 2)**
* **(Befehlsfolge 3)**
.
.
.
* **(Befehlsfolge n)**
* **nur RETURN (PIP verlassen mit Warmstart)**

Jede Befehlsfolge ist in sich abgeschlossen und wird einzeln ausgeführt.

Allgemeine Form der Befehlsfolge:

Ziel=Quelle[Option]

wobei Ziel

(Laufwerk:) Dateiname

oder

Ausgabekanal auf Peripheriegerät: LPT: oder AUX:

und Quelle

(Laufwerk:) Dateiname

oder

(Laufwerk:) Dateiname 1, (Laufwerk:) Dateiname 2, ...
... (Laufwerk:) Dateiname n

oder

Eingabekanal von Peripheriegerät: CON: oder AUX:

Standardbelegung der Ein-/Ausgabekanäle:

CON: (Konsole) ==> CRT: (Bildschirm u. Tastatur)
AUX: (Hilfskanal) ==> LPT: (Centronics-Schnittstelle)
LST: (Listeinheit) ==> LPT: (Centronics-Schnittstelle bzw. Drucker)

Beispiele: Befehlsfolgen (ohne Optionen):

DATEINEU.TXT=DATEIALT.TXT

kopiert die Datei DATEIALT.TXT, die im Bezugslaufwerk abgelegt ist, in die Datei DATEINEU:TXT, die ebenfalls im Bezugslaufwerk angelegt wird.

B:DATEI1.COM=A:DATEI1.BAK

kopiert die Datei DATEI1.BAK von Laufwerk A nach Laufwerk B, wo sie unter dem Namen DATEI1.COM abgelegt wird.

A:=B:LAGER.DAT

kopiert die Datei LAGER.DAT von Laufwerk B nach Laufwerk A, wo sie unter dem gleichen Namen abgelegt wird.

B:=A:.COM*

kopiert sämtliche COM-Dateien von Laufwerk A nach Laufwerk B.

B:=A.**

kopiert alle Dateien von Laufwerk A nach Laufwerk B.

B:DAT.TXT=A:DAT1.TXT,A:DAT2.TXT,B:DAT3.TXT

bildet die Datei DAT.TXT in Laufwerk B durch Aneinanderreihen der Dateien DAT1.TXT und DAT2.TXT in Laufwerk A sowie der Datei DAT3.TXT in Laufwerk B.

LST:=PROBE.TXT

gibt die Datei PROBE.TXT im Bezugslaufwerk auf dem Drucker aus.

CON:=B:TEST1.DAT,B:TEST2.DAT

listet die Dateien TEST1.DAT und TEST2.DAT, die sich beide in Laufwerk B befinden, auf dem Bildschirm aus.

Optionen: **[A]**

Es werden nur die Dateien kopiert, die nach dem letzten Kopieren geändert wurden, d.h. deren Archiv-Attribut ausgeschaltet ist (siehe SET). Nach dem Kopiervorgang wird das Archiv-Attribut wieder eingeschaltet.

[C]

Werden mehrere Dateien unter Angabe eines mehrdeutigen Dateinamens kopiert, findet für jede einzelne Datei eine Abfrage statt, an Hand derer der Benutzer entscheiden kann, ob die betreffende Datei kopiert werden soll oder nicht.

[Dn]

Bei Angabe dieser Option werden Textdateien nur bis zu dem Zeichen "n" pro Zeile übertragen. Dies hat beispielsweise den Vorteil, daß nur soviele Zeichen in eine Bildschirm- oder Druckerzeile übertragen werden, wie darin Platz ist. Die Zeilen werden dabei hinter dem n-ten Zeichen abgeschnitten.

Beispiel: *LST:=LISTE.TXT[D35]*

Von der Datei LISTE.TXT werden jeweils nur 35 Zeichen pro Zeile auf dem Drucker ausgegeben, wobei die restlichen Zeichen unterdrückt werden.

Option: **[E]**

Dies ist eine Echofunktion, die Textdateien während ihrer Übertragung auf dem Bildschirm auflistet. Man kann somit überprüfen, ob die Datei richtig übertragen wird.

Beispiel: *B:DATEN2.TXT=A:DATEN1.TXT[E]*

Die Datei DATEN1.TXT in Laufwerk A wird in die Datei DATEN2.TXT in Laufwerk B kopiert und gleichzeitig auf dem Bildschirm ausgegeben.

Option: **[F]**

Diese Option entfernt bei der übertragung sämtliche Formfeed-Zeichen (0CH bzw. CTRL-L) in Textdateien, die auf dem Drucker einen Seitenvorschub veranlassen.

Beispiel: *CON:=PROBE.TXT[F]*

Hierbei werden in der Datei PROBE.TXT, sämtliche Formfeed-Zeichen entfernt, während sie auf dem Bildschirm ausgegeben wird. Für die Bildschirmausgabe sind diese Zeichen nämlich nicht sinnvoll, da sie jedesmal den Bildschirm löschen.

Option: **[Gn]**

Kopiert Dateien aus dem angegebenen Benutzerbereich "n" in den gegenwärtigen Benutzerbereich.

Beispiel: *TEXT.DAT[G3]=BRIEF.BAK*

Die Datei BRIEF.BAK, die im Benutzerbereich 3 abgelegt ist, wird in den aktuellen Bereich kopiert. Falls nichts anderes angegeben ist, handelt es sich hierbei immer um den Standard-Bereich 0.

Optionen: **[H]**

Prüft nach, ob sich die übertragenden Daten im Intel-Hex-Format befinden. Diese Option spielt nur bei der Assembler-Programmierung eine Rolle.

[I]

Übergeht bei der übertragung von Intel-Hex-Dateien sämtliche Nullbytes. Diese Option spielt nur bei der Assembler-Programmierung eine Rolle.

[L]

Wandelt Textdateien während der übertragung in Kleinbuchstaben um.

Beispiel: *CON:=PROBE.TXT[L]*

Die Datei PROBE.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei nur in Kleinschrift.

Option: **[N]**

Jeweils am Zeilenanfang von Textdateien werden Zeilennummern gesetzt.

Beispiel: CON:=BRIEF.TXT[N]

Die Datei BRIEF.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei in folgender Form:

```
1:  (1. Zeile)
2:  (2. Zeile)
3:  (3. Zeile)
usw.
```

Option: **[N2]**

Jeweils am Zeilenanfang von Textdateien werden sechsstellige Zeilennummern mit führenden Nullen gesetzt.

Beispiel: *CON:=BRIEF.TXT[N2]*

Die Datei BRIEF.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei in folgender Form:

```
000001   (1. Zeile)
000002   (2. Zeile)
000003   (3. Zeile)
usw.
```

Option: **[O]**

Diese Option ist nur dann notwendig, wenn andere Dateien als Textdateien miteinander verbunden werden sollen, wobei keine Abfrage auf das Textendezeichen (CTRL-Z) stattfindet.

Beispiel: *DATEINEU.BIN=DATEI1.BIN,DATEI2.BIN,DATEI3.BIN[O]*

Hierin werden die Dateien DATEI1.BIN, DATEI2.BIN und DATEI3.BIN hintereinander in die Datei DATEINEU.BIN geschrieben, wobei jeweils die volle physikalische Länge ohne Abfrage auf CTRL-Z übertragen wird.

Option: **[Pn]**

Diese Option fügt alle "n" Zeilen ein Formfeed-Zeichen (0CH bzw. CTRL-L) in eine Texdatei ein. Formfeed-Zeichen sorgen bei der Druckerausgabe für einen Seitenvorschub.

Beispiel: *LST:=PROBE.TXT[P64]*

Bei der Ausgabe der Datei PROBE.TXT über den Drucker findet alle 64 Zeichen ein Seitenvorschub statt.

Option: **[Q"Begriff"^Z]**

Mit dieser Option wird eine Textdatei bis zu der Stelle übertragen, an der ein vorgegebener Suchbegriff erscheint. Dabei wird der "Begriff" selbst mitübertragen. Die Option sollte nur nach separatem Starten von PIP erteilt werden, wenn das Sternchen zur Befehlseingabe auffordert.

Beispiel: *CON:=LAGER.DAT[QStiefel^Z]*

Die Datei LAGER.DAT wird auf dem Bildschirm ausgegeben und zwar bis zu der Stelle, an der der Begriff "Stiefel" auftritt.

Option: **[R]**

Dient zum Kopieren von Systemdateien.

Beispiel: *B:=A:GEHEIM.DAT[R]*

Die Systemdatei GEHEIM.DAT wird von Laufwerk A nach Laufwerk B übertragen, wo sie ebenfalls als Systemdatei abgelegt wird.

Option: **[S"Begriff"^Z]**

Mit dieser Option wird eine Textdatei ab der Stelle übertragen, an der ein vorgegebener Suchbegriff erscheint. Dabei wird der "Begriff" selbst mit übertragen. Die Option sollte nur nach separatem Starten von PIP erteilt werden, wenn das Sternchen zur Befehlseingabe auffordert.

Beispiel: *CON:=LAGER.DAT[SHemd^Z]*

Die Datei LAGER.DAT wird auf dem Bildschirm ausgegeben und zwar ab der Stelle, an der der Begriff "Hemd" auftritt.

Optionen: **[Tn]**

Setzt Tabulator für Ausdrucke, wobei "n" der Spaltenbreite in Zeichen entspricht.

[U]

Wandelt Textdateien während der übertragung in Grußbuchstaben um.

Beispiel: *CON:=PROBE.TXT[U]*

Die Datei PROBE.TXT wird auf dem Bildschirm ausgegeben und erscheint dabei nur in Großschrift.

Option: **[V]**

Während der Übertragung von Diskettendateien führt diese Option ein Verify (Vergleich) durch. Nach der normalen Übertragung wird dabei die Ziel- mit der Quelldatei nochmals verglichen. Bei fehlerhafter Übertragung erscheint die Meldung VERIFY ERROR (Befehlsfolge).

Beispiel: *B:=A:*.TXT[V]*

Hier werden sämtliche TXT-Dateien von Laufwerk A nach Laufwerk B mit Verify übertragen.

Option: [W]

Mit dieser Option werden schreibgeschützte Dateien (R/O) ohne Sicherheitsfrage überschrieben.

Beispiel: *4B:=A:PROTECT.COM[W]*

Die Datei PROTECT.COM wird von Laufwerk A nach B übertragen und überschreibt dort eine möglicherweise vorhandene Datei unter gleichem Namen, selbst wenn diese schreibgeschützt ist.

Option: [Z]

Die Option löscht bei der Übertragung von ASCII-Dateien das höchstwertige Bit (Bit 7) und wird nur für spezielle Anwendungsfälle eingesetzt.

Besondere Gerätezuordnung:

Neben den standardmäßigen Ein-/Ausgabekanälen, CON:, AUX: und LST: (siehe oben) kann bei PIP noch folgende Gerätebezeichung verwendet werden:

PRN:

PRN: verwendet den LST:-Kanal, über den in der Regel der Drucker angeschlossen ist, zum Ausdrucken von Texten. Dabei werden jedoch folgende Optionen automatisch gesetzt, sofern nichts anderes vorgegeben ist:

Seitenvorschub alle 60 Zeilen
Tabulator auf 8 Positionen
Numerierung der Textzeilen

Diese Eigenschaften entsprechen den Optionen [NPT8] (siehe oben).

Beispiel: *PRN:=DATEN.TXT*

druckt die Datei DATEN.TXT mit den oben angegebenen Optionen aus.

PROFILE.SUB, PROFILE.ENG, PROFILE.GER

Beschreibung: Eine PROFILE.SUB-Datei ist eine Stapel-Datei (Befehls-Datei), die beim Booten des CP/M Plus-Betriebssystems ausgeführt wird. Sie unterliegt den gleichen Regeln wie eine SUBMIT-Datei (siehe SUBMIT) und besteht aus einer ASCII-Datei mit den entsprechenden Befehlszeilen.

Die Dateien PROFILE.ENG (nur Schneider CPC 6128) bzw. PROFILE.GER (nur JOYCE) können jedoch beim Booten nicht ausgeführt werden, da sie nicht die Extension SUB enthalten, und müssen zu diesem Zweck erst mit REN umbenannt werden:

Beispiele: *REN PROFILE.SUB=PROFILE.ENG* (CPC 6128)

bzw.

REN PROFILE.SUB=PROFILE.GER (JOYCE)

Die Datei PROFILE.ENG des CPC 6128 hat folgenden Inhalt:

```
SETKEYS.CCP
LANGUAGE 3
```

und die Datei PROFILE.ENG des JOYCE ist folgendermaßen aufgebaut:

```
SETDEF M:,* [ORDER = (SUB,COM) TEMPORARY = M:]
PIP
<M:=BASIC.COM(0)
<M:=DIR.COM(0)
<M:=ERASE.COM(0)
<M:=PAPER.COM(0)
<M:=PIP.COM(0)
<M:=RENAME.COM(0)
<M:=SHOW.COM(0)
<M:=SUBMIT.COM(0)
<M:=TYPE.COM(0)
<
```

Die PROFILE-SUB-Dateien können jedoch beliebige Befehlsfolgen enthalten und, entsprechend angepaßt, neu geschrieben oder erweitert werden.

PUT

Nichtresidenter Standard-CP/M Plus-Befehl (Datei PUT.COM)

Beschreibung: PUT leitet die Bildschirm- oder Drucker-Ausgabe in eine Disketten-Datei um. Nach Erteilung des PUT-Befehls wird die Ausgabe des nächsten aufgerufenen Programms in die Datei geschrieben.

Aufruf: **PUT CONSOLE OUTPUT TO Dateiname [Option]**

für die Bildschirmausgabe und

PUT PRINTER OUTPUT TO Dateiname [Option]

für die Druckerausgabe, wobei die Dateinamen ohne Extension anzugeben sind.

Optionen: **[ECHO]**

Die Ausgabe findet, außer in die Datei, auch auf dem Bildschirm statt (Standardeinstellung).

[NOECHO]

Keine zusätzliche Ausgabe auf dem Bildschirm.

[FILTER]

Sämtliche Steuerzeichen (ESC und CTRL) werden in listbare ASCII-Zeichen umgewandelt.

[NOFILTER]

Eine Übersetzung der Steuerzeichen findet nicht statt.

[SYSTEM]

System- und Programmausgabe werden in die angegebene Datei geschrieben und zwar solange, bis ein PUT CONSOLE-Befehl die Ausgabe wieder auf den Bildschirm lenkt.

Beispiele: *PUT CONSOLE OUTPUT TO TEXT.DAT [NOECHO]*

lenkt die Terminalausgabe in die Datei TEXT.DAT um, wobei sie nicht zusätzlich noch auf dem Bildschirm angezeigt wird.

PUT PRINTER OUTPUT TO DRUCK.DAT

lenkt die Druckerausgabe in die Datei DRUCK.DAT um.

PUT CONSOLE OUTPUT TO CONSOLE

lenkt die Konsolenausgabe wieder auf die Konsole.

PUT PRINTER OUTPUT TO PRINTER

lenkt die Druckerausgabe wieder auf den Drucker.

REN

Teils residenter, teils nichtresidenter Standard-CP/M Plus-Befehl (Datei RENAME.COM)

Beschreibung: REN dient zum Umbenennen einzelner Dateien, wobei auch mehrdeutige Dateinamen zugelassen sind. Existiert bereits eine Datei unter dem neuen Namen, findet zunächst eine Sicherheitsabfrage statt.

Aufruf: **REN (Laufwerk:)Neuer Dateiname=Alter Dateiname**

Beispiele: *REN ARTIKEL.TXT=LAGER.DAT*

nennt im Bezugslaufwerk die Datei LAGER.DAT in die Datei ARTIKEL.TXT um. Dasselbe geschieht mit

REN B:ARTIKEL.TXT=LAGER.DAT

wenn sich die Datei in Laufwerk B befindet.

*REN *.TXT=*.BAK*

nennt sämtliche BAK-Dateien in TXT-Dateien um.

Eine weitere Möglichkeit besteht in der Eingabe von

REN

oder

RENAME

worauf der neue und der alte Dateiname gesondert angefordert werden. In diesem Fall, wie auch bei der Angabe mehrdeutiger Dateinamen, ist die Datei RENAME.COM erforderlich.

RMAC

Nichtresidenter Standard-CP/M Plus-Befehl (Datei RMAC.COM)

Beschreibung: RMAC ist ein Makro-Assembler, der ASM-Quelltextdateien in verschiebbare Objektmodule vom Typ REL umsetzt, die anschließend mit LINK verbunden und in ausführbare COM-Dateien umgesetzt werden können. Zusätzlich wird noch eine PRN- und eine SYM-Datei mit Symbolen erzeugt (siehe auch MAC).

Aufruf: **RMAC Dateiname $Option**

Optionen: **$Rd**

Laufwerk d für REL-Datei

$Sd

Laufwerk d für SYM-Datei

$Pd

Laufwerk d für PRN-Datei

Achtung! Anstelle des Laufwerks kann "d" auch für Bildschirm (d=X) oder für Drucker (d=P) stehen. Soll keine Ausgabe stattfinden, ist d=Z zu setzen.

Beispiel: *RMAC MUSTER $RA SB PP*

Hier wird die Datei MUSTER.ASM assembliert, wobei die REL-Datei in Laufwerk A und die SYM-Datei in Laufwerk B abgelegt wird. Die PRN-Datei dagegen wird auf dem Drucker ausgegeben.

SAVE

Nichtresidenter Standard CP/M Plus-Befehl (Datei SAVE.COM)

Beschreibung: SAVE dient im Zusammenhang mit SID zum erneuten Abspeichern von Dateien, die im TPA zur Bearbeitung abgelegt sind. Vor SID muß dabei jedoch SAVE aufgerufen werden.

Aufruf: **SAVE**

und anschließend

SID

(eventuell mit Dateiname). Nachdem die Datei mit SID bearbeitet wurde, ist SID mit

G0

zu verlassen, wodurch SAVE automatisch aufgerufen wird. Daraufhin ist der Dateiname, die Anfangs- und die Endadresse des abzuspeichernden Speicherbereichs anzugeben.

Achtung! SAVE ist unbedingt vor SID aufzurufen, da ansonsten ein Teil des TPA überschrieben wird, wodurch im Speicher abgelegte Dateien zerstört werden.

SET

Nichtresidenter Standard-CP/M Plus-Befehl (Datei SET.COM)

Beschreibung: SET ist ein Befehl, der in einem CP/M Plus-System die verschiedensten Aufgaben übernimmt. Dabei handelt es sich vorwiegend um die Einrichtung bzw. Aufhebung von Schutzvorrichtungen vor unberechtigtem Zugriff auf die Diskette oder auf einzelne Dateien.

Anwendung:
1) Setzen des Directory-Labels
2) Setzen von Passwords
3) Initialisierung der Timestamps
4) Setzen/Aufheben von Datei-Attributen

Der Aufruf unterscheidet sich je nach Anwendungsbereich.

1a) Setzen des Directory-Labels:

Passwords und Timestamps sind nur möglich, wenn die Diskette mit einem Directory-Label (Namen) versehen ist. Dieses besteht aus einer Namensbezeichnung, ähnlich wie die einzelner Dateinamen.

Aufruf: **SET (Laufwerk:) [NAME=Directory-Label]**

Beispiel: *SET B: [NAME=RECHNUNG.DAT]*

Die Diskette in Laufwerk B erhält das Directory-Label RECHNUNG.DAT.

1b) Schützen des Directory-Labels mit einem Password

Das Directory-Label kann durch ein Password (Schutzwort) geschützt werden. Dieser Schutz ist besonders dann zu empfehlen, wenn auch einzelne Dateien Password-geschützt werden sollen, da ansonsten dieser Schutz leicht aufgehoben werden kann.

Aufruf: **SET (Laufwerk:) [PASSWORD=Schutzwort]**

Beispiel: *SET [PASSWORD=GEHEIM]*

Das Directory-Label der Diskette im Bezugslaufwerk erhält hier das Password GEHEIM.

2) Initialisierung des Password-Schutzes für Dateien

Bevor Dateien mit einem Password versehen werden können, ist der Datei-Password-Schutz für eine Diskette zu initialisieren. Geschieht dies nicht, ist der Schutz unwirksam, selbst wenn ein Password vereinbart wird. Um diesen Schutz einzurichten oder aufzuheben, wird jedoch das Password des Directory-Labels abgefragt, sofern es vereinbart wurde. Wenn nicht, ist auch der Password-Schutz für Dateien sinnlos, da er ohne weiteres aufgehoben werden kann.

Aufruf: **SET (Laufwerk:) [PROTECT=ON]** (Initialisierung)

SET (Laufwerk:) [PROTECT=OFF] (Aufhebung)

Beispiele: *SET B: [PROTECT=ON]*

richtet den Password-Schutz für Dateien in Laufwerk B ein, wogegen

SET B: [PROTECT=OFF]

ihn wieder aufhebt. In beiden Fällen wird jedoch zunächst das Password des Directory-Labels abgefragt, falls dieses vereinbart wurde.

Die hier gezeigte Einrichtung/Aufhebung des Password-Schutzes für Dateien bezieht sich auf Lesen, Beschreiben und Löschen der betreffenden Datei. Soll jedoch nur einer dieser Bereiche geschützt werden, so ist dies im Anschluß daran gesondert zu vereinbaren. Dabei wird nicht das Password des Directory-Labels, sondern dasjenige der betreffenden Datei abgefragt.

Aufruf: **SET (Laufwerk:) Dateiname [PROTECT=READ]**

oder

SET (Laufwerk:) Dateiname [PROTECT=WRITE]

oder

SET (Laufwerk:) Dateiname [PROTECT=DELETE]

oder

SET (Laufwerk:) Dateiname [PROTECT=NONE]

Hierin bedeuten

READ

Password-Schutz zum Lesen, Schreiben und Löschen (standardmäßig durch (PROTECT=ON) vorgegeben)

WRITE

Password-Schutz ausschließlich zum Schreiben

DELETE

Password-Schutz ausschließlich zum Löschen

NONE

Aufhebung des Password-Schutzes

Beispiel: *SET B: MUSTER.TXT [PROTECT=DELETE]*

Der Password-Schutz für die Datei MUSTER.TXT wird nachträglich nur auf das Löschen beschränkt. Dabei wird das Password vor Ausführung des Befehls abgefragt.

3) Timestamps

Grundsätzlich ist vor dem Setzen der Timestamps das Programm INITDIR auszuführen, welches das Directory für die Aufnahme von Timestamps einrichtet. Erst wenn dies geschehen ist, können die einzelnen Timestamps separat aktiviert werden. Dies kann jedoch nur für eine ganze Diskette (Laufwerk), nicht jedoch separat für eine einzelne Datei geschehen. Außerdem sind Timestamps nur dann sinnvoll, wenn vorher mit dem DATE-Befehl die interne Uhr mit Datumsanzeige gestellt wurde.

Aufruf: **SET (Laufwerk:) [CREATE=ON]**

aktiviert Timestamps für die Erstellung von Dateien,

SET (Laufwerk:) [ACCESS=ON]

aktiviert Timestamps für den letzten Lese-Zugriff auf Dateien und

SET (Laufwerk:) [UPDATE=ON]

aktiviert Timestamps für den letzen Schreib-Zugriff auf Dateien.

Wird anstelle von ON OFF angegeben, werden die betreffenden Timestamps wieder deaktiviert. Außerdem kann immer nur CREATE oder ACCESS aktiviert sein, und nicht jedoch beides gleichzeitig.

4a) Dateiattribute

Mit SET können verschiedene Dateiattribute gesetzt oder zurückgesetzt werden. Sie stellen zusätzlich zum Password-Schutz eine weitere Schutzeinrichtung dar.

Aufruf: **SET (Laufwerk:) Dateiname [Dateiattribut]**

Hier die einzelnen Dateiattribute:

[ARCHIVE=OFF]

setzt das Archive-Attribut zurück. Eine solche Datei kann dann mit PIP und der Option [A] kopiert werden. Nach dem Kopieren mit PIP wird das Archive-Attribut gesetzt. Diese Einrichtung von PIP ermöglicht es, nur solche Dateien zu kopieren, die seit dem letzten Kopieren geändert wurden.

[ARCHIVE=ON]

schaltet das Archive-Attribut ein (siehe oben) Es erscheint dann im erweiterten Directory-Listing.

[SYS]

erklärt Dateien zu Systemdateien, wobei das System-Attribut gesetzt wird.

[DIR]

erklärt Systemdateien zu Directory-Dateien, wobei das Directory-Attribut gesetzt wird. Dieses Attribut ist bereits standardmäßig eingeschaltet und kann nur durch das System-Attribut aufgehoben werden.

[RO]

versieht eine Datei mit einem Schreibschutz.

[RW]

hebt den Schreibschutz wieder auf. Falls kein Schreibschutz eingerichtet wird, erhalten sämtliche Dateien automatisch das RW-Attribut.

[F1=ON/OFF]
[F2=ON/OFF]
[F3=ON/OFF]
[F4=ON/OFF]

Die benutzerdefinierten Attribute F1 bis F4 werden gesetzt (ON) bzw. zurückgesetzt (OFF).

Beispiele: *SET MUSTER.DAT [RO]*

versieht die Datei MUSTER.DAT im Bezugslaufwerk mit einem Schreibschutz,

SET MUSTER.DAT [RW]

hebt diesen Schreibschutz wieder auf,

SET B:BILANZ.COM [RO,SYS]

versieht die Datei BILANZ.COM in Laufwerk B mit einem Schreibschutz und erklärt sie gleichzeitig zur Systemdatei.

Im letzten Beispiel hätten beide Attribute auch getrennt gesetzt werden können, jedoch ist es einfacher, dies in einer Befehlszeile vorzunehmen.

4b) Schreibschutz für Laufwerk

Neben einzelnen Dateien kann auch ein Laufwerk vor Schreibzugriffen geschützt werden. Dabei ist jedoch zu beachten, daß der Schreibschutz nur solange bestehen bleibt, wie der Computer eingeschaltet ist. Im Gegensatz zum Schreibschutz für Dateien wird hier nämlich kein Vermerk auf die Diskette geschrieben.

Aufruf: **SET (Laufwerk:) [RO]**

richtet den Schreibschutz ein und

SET (Laufwerk:) [RW]

hebt ihn wieder auf.

Beispiel: *SET B: [RO]*

versieht Laufwerk B mit einem temporären Schreibschutz.

SET24X80

Gerätespezifischer CP/M Plus-Befehl (Datei SET24X80.COM)

Beschreibung: Manche Programme benötigen einen Bildschirm mit 24 Zeilen zu je 80 Spalten, der mit diesem Befehl ein- bzw. abgeschaltet werden kann.

Aufruf: **SET24X80 ON**

schaltet den 24x80 Modus ein und

SET24X80 OFF

schaltet ihn wieder ab.

SETDEF

Nichtresidenter CP/M Plus-Befehl (Datei SETDEF.COM)

Beschreibung: Mit SETDEF kann ein sogenannter Suchpfad festgelegt werden, der angibt, in welcher Reihenfolge die einzelnen Laufwerke nach SUB- oder COM-Dateien durchsucht werden sollen. Auch kann ein Laufwerk für temporäre Dateien bestimmt werden.

Aufruf: **SETDEF (Laufwerk 1:, Laufw. 2:, Laufw. 3:, Laufw. 4:)**

Zum Auffinden einer Datei werden die Laufwerke in der angegebenen Reihenfolge durchsucht. Das Bezugslaufwerk wird hierbei mit "*" gekennzeichnet.

SETDEF [TEMPORARY=Laufwerk:]

bestimmt das angegebene Laufwerk zur Ablage von temporären Dateien. (Das sind Zwischendateien, deren Dateinamen meist $-Zeichen enthalten).

SETDEF [ORDER=(Extension 1, Extension 2)]

sucht zuerst nach dem angegebenen Dateinamen mit der Extension 1 und falls dieser nicht gefunden wird, nach demjenigen mit der Extension 2. Als Extensionen sind hier nur SUB und COM zulässig.

SETDEF [DISPLAY]

zeigt zusätzlich an, von welchem Laufwerk und welcher Benutzernummer die betreffende Datei geladen wurde.

SETDEF [NO DISPLAY]

schaltet DISPLAY wieder ab.

SETDEF [PAGE]

läßt die Bildschirmausgabe nach einer vollen Seite anhalten, und zwar solange, bis eine beliebige Taste gedrückt wird. Dieser Vorgang wird mit

SETDEF [NOPAGE]

wieder rückgängig gemacht.

Beispiele: *SETDEF M:,B:,**

durchsucht zuerst Laufwerk M, dann Laufwerk B und schließlich das Bezugslaufwerk (*) nach der gesuchten Datei.

SETDEF [TEMPORARY=B:]

legt alle temporären (Zwischen-) Dateien in Laufwerk B ab, die beispielsweis von PIP, ED oder SUBMIT erzeugt werden.

SETDEF [ORDER=(SUB,COM)]

sucht zunächst eine SUB-Datei unter dem angegebenen Namen. Wird diese nicht gefunden, wird die Suche nach einer COM-Datei fortgesetzt.

Es folgt ein Beispiel, das in der PROFILE.GER-Datei des JOYCE enthalten ist und das mehrere Befehle in einer SETDEF-Anweisung beinhaltet (siehe auch Kapitel 6.4):

SETDEF M:, [ORDER = (SUB,COM) TEMPORARY=M:]*

Hier werden zunächst SUB- und dann COM-Dateien in Laufwerk M gesucht. Wird die Datei dort nicht aufgefunden, wird die Suche im Bezugslaufwerk (meist A) fortgesetzt. Zwischendateien werden in Laufwerk M abgelegt.

Aufruf: **SETDEF**

(ohne Zusatz) ergibt eine Auflistung der einzelnen Zuordnungen, die somit abgefragt werden können.

SETKEYS

Gerätespezifischer CP/M Plus-Befehl (Datei SETKEYS.COM)

Beschreibung: Dieser Befehl dient zum Umstellen der Tastatur, so daß beispielsweise eine deutsche DIN-Tastatur auf dem CPC 6128 erzeugt werden kann. Darüber hinaus können noch bis zu 32 Erweiterungstasten mit Strings oder Befehlszeilen definiert werden.

Aufruf: **SETKEYS ASCII-Datei mit Tastaturbelegung**

Beispiel: *SETKEYS DEUTSCH.KEY*

Die Datei, die die neue Tastaturbelegung enthält, ist eine reine Text- bzw. ASCII-Datei, die mit jedem Texteditor (z.B. ED) erstellt werden kann. Ein ausführliches Beispiel hierfür ist in Kapitel 3.10 beschrieben.

SETLST

Gerätespezifischer CP/M Plus-Befehl (Datei SETLST.COM)

Beschreibung: Dieser Befehl dient zum Initalisieren des Druckers, indem an ihn bestimmte Steuerzeichen geschickt werden. SETLST wird nur für bestimmte Drucker und bestimmte Anwendungsfälle benötigt.

Aufruf: **SETLST Dateiname**

Beispiel: *SETLST PRINT*

Dieser Aufruf sendet die in der Datei PRINT enthaltenen Steuerzeichen an den Drucker. Der Aufbau dieser Datei unterliegt den gleichen Regeln wie derjenige einer Datei, die für SETKEYS benötigt wird. Weitere Einzelheiten sind im Handbuch beschrieben.

SETSIO

Gerätespezifischer CP/M Plus-Befehl (Datei SETSIO.COM)

Beschreibung: Dieser Befehl dient zum Setzen verschiedener Parameter für eine serielle Schnittstelle, sofern diese angeschlossen ist.

Aufruf: **SETSIO Parameter**

Folgende Parameter lassen sich eingestellt werden, wobei auch gleichzeitig mehrere Parameter vorgegeben werden können:

Parameter **TX n**

Baudrate für Empfänger (n = 50 bis 19200)

RX n

Baudrate für Sender (n = 50 bis 19200)

n

Baudrate für Sender und Empfänger (n = 50 bis 19200)

BITS n

n Datenbits (n = 5, 6, 7 oder 8)

STOP n

n Stop-Bits (n = 1, 1.5 oder 2)

PARITY EVEN

Gerade Parität

PARITY ODD

Ungerade Parität

PARITY NONE

Keine Parität

XON ON

XON-Protokoll ein

XON OFF

XON-Protokoll aus

HANDSHAKE ON

Handshake ein

HANDSHAKE OFF Handshake aus

Beispiele: *SETSIO 1200*

setzt die Baudrate für Sender und Empfänger auf 1200.

SETSIO 300, PARITY NONE, HANDSHAKE=ON, STOP 1, BITS 7

Die Übertragung findet hier mit einer Baudrate von 300 in beiden Richtungen statt. Darüber hinaus werden 7 Bits und 1 Stop-Bit ohne Parität mit eingeschaltetem Handshake übertragen.

SHOW

Nichtresidenter Standard-CP/M Plus-Befehl (Datei SHOW.COM)

Beschreibung: Mit SHOW können verschiedene Laufwerks- bzw. Diskettenparameter abgefragt werden, die in den einzelnen Optionen anzugeben sind.

Aufruf: **SHOW (Laufwerk:) [Option]**

Optionen: **[SPACE]**

zeigt den noch freien Speicherplatz des Laufwerks an.

[DRIVE]

Anzeige der physikalischen Laufwerks-Eigenschaften.

[DIR]

Anzeige der noch nicht belegten Directory-Einträge.

[LABEL]

Anzeige des Directory-Labels (falls vorhanden) mit dessen Timestamps und der Angabe, ob Dateien auf der Diskette für Passwords und Timestamps aktiviert sind.

[USER]

Angaben über die belegten Benutzerbereiche.

Keine Option:

Wie SPACE-Option, jedoch für sämtliche angeschlossenen Laufwerke, sofern auf sie bereits mindestens einmal zugegriffen wurde.

Ausführliche Beispiele (einschließlich Auflistungen) zu sämtlichen Optionen befinden sich in Kapitel 3.

SID

Nichtresidenter Standard-CP/M Plus-Befehl (Datei SID.COM)

Beschreibung: SID ist ein universeller Debugger, um Speicherinhalte und Dateien zu untersuchen und gegebenenfalls zu ändern. Falls eine Datei bearbeitet werden soll, wird sie im Regelfall ab der Adresse 100H abgelegt, an der sonst auch CP/M-Programme geladen und abgearbeitet werden. In diesem Fall untersucht man den Speicherbereich, in dem sich die Datei befindet.

SID eignet sich weniger zum Schreiben ganzer Maschinenprogramme als vielmehr zum Ändern einzelner Speicherzellen. Dieser Vorgang, den man auch als Patchen bezeichnet, muß häufig zur Anpassung von CP/M-Software durchgeführt werden, wenn das mitgelieferte Installationsprogramm keine Anpassung an den CPC6128 oder JOYCE vorsieht.

Anwendung: Speicherinhalt ansehen
Speicherinhalt ändern
Ausführen von Maschinenprogrammen, wobei Haltepunkte (Breakpoints) gesetzt werden können
Speicherbereiche verschieben
Speicherbereiche mit konstantem Wert füllen
Assemblieren von 8080-Quelltext (nur bedingt möglich)
Disassemblieren von Maschinencode (nur bedingt möglich)

Aufruf: **SID (Laufwerk:) Dateiname**

oder

SID

Im ersten Fall wird die zu untersuchende Datei gleich mitgeladen, im zweiten Fall nicht. In beiden Fällen erscheint dann ein Ziffernkreuz, das zur Eingabe spezieller SID-Befehle auffordert:

```
#(SID-Befehl)
```

Achtung! Soll die geänderte Datei wieder auf Diskette geschrieben werden, so ist dies entweder mit dem SID-W-Befehl (siehe unten) oder mit SAVE vorzunehmen. Um eine Zerstörung der Datei zu vermeiden, ist vor dem Aufruf von

SID unbedingt der SAVE-Befehl auszuführen. SID ist dann entweder mit

G0

oder

CTRL-C

(Warmstart) zu verlassen und die Datei anschließend mit SAVE abzuspeichern. Nähere Einzelheiten hierzu siehe unter SAVE.

Beispiel: SID-Aufruf mit Dateinamen:

SID COPY.COM

lädt SID und gleichzeitig die Datei COPY.COM in den Speicher ab Adresse 100H. SID befindet sich anschließend im Befehlsmodus, so daß weitere Befehle zur Bearbeitung der Datei erteilt werden können.

SID-Befehle:

Befehl: **A**

Dieser Befehl ruft einen einfachen 8080-Assembler auf und ist in der Form von

Aufruf: **A Anfangsadresse** (hex.)

einzugeben. Daraufhin erscheint am linken Rand die Anfangsadresse und fordert zur Eingabe eines mnemonischen 8080-Befehlswortes auf. Handelt es sich dabei um einen gültigen Befehl, wird er durch Drücken der RETURN-Taste sogleich übersetzt und der Maschinencode an der richtigen Stelle im Speicher abgelegt. Es erscheint die nächstfolgende Adresse, worauf ein neues Befehlswort einzugeben ist, und der Vorgang kann von neuem beginnen. Ist das Befehlswort ungültig, erscheint ein Fragezeichen und fordert zur erneuten Eingabe auf.

Der Assembler ist einfach durch Drücken der RETURN-Taste wieder zu verlassen. Eine Verwendung von symbolischen Adressen und Namen (Label) ist nicht möglich und der Maschinencode ist nicht verschiebbar (relokatibel).

Beispiel: *A3000*

ruft den Assembler auf, worauf mit der Eingabe von Befehlswörtern begonnen werden kann. Der übersetzte Maschinencode wird dabei ab Adresse 3000H abgelegt.

Befehl: **C**

"C" dient zum Aufruf eines Maschinenprogramms, wobei der Inhalt des BC- und DE-Registers übergeben werden kann.

Beispiel: *C1234 013A,FFFF*

ruft das Maschinenprogramm ab Adresse 1234H auf, wobei das BC-Register den Inhalt 013AH und das DE-Register den Inhalt FFFFH enthält.

Befehl: **D**

zeigt einen Speicherinhalt auf dem Bildschirm an. Er erscheint dabei als hexadezimales Listing, wobei darstellbare ASCII-Zeichen rechts im Klartext erscheinen. Der Aufruf von "D" kann auf verschiedene Arten stattfinden:

Aufruf: **D**

(ohne Adressenangabe) listet die nächsten 12 Zeilen, von denen jede 16 Bytes darstellt. Wurde SID gerade aufgerufen (mit oder ohne Dateiname) beginnt die Darstellung automatisch bei 100H. Durch erneute Eingabe von "D" erscheinen dann jeweils die nächsten 12 Zeilen.

D Adresse

arbeitet genauso wie "D", beginnt aber bei vorgegebener Anfangsadresse.

Beispiel: *D3000*

gibt 12 Zeilen ab Adresse 3000H aus.

Darüber hinaus kann "D" auch eine Anfangs- und Endadresse enthalten.

Aufruf: **D Anfangsadresse, Endadresse**

Beispiel: *D1800,2000*

listet den Speicherinhalt zwischen den Adressen 1800H und 2000H aus.

Befehl: **F**

Füllt einen Speicherbereich mit einem vorgegebenen konstanten Wert.

Aufruf: **F Anfangsadresse, Endadresse, Wert**

Beispiel: *F3000,3FFF,AA*

füllt den Speicher zwischen den Adressen 3000H und 3FFFH (je einschließlich) mit dem konstanten Wert AAH.

Befehl: **G**

Führt ein Maschinenprogramm aus, wobei die Möglichkeit besteht, bis zu zwei Haltepunkte (Breakpoints) vorzugeben.

Aufruf: **G Anfangsadresse, Breakpoint 1, Breakpoint 2**

Fehlt die Anfangsadresse, beginnt die Ausführung beim momentanen Programmzählerstand. Auch ist die Angabe eines zweiten Breakpoints nicht unbedingt erforderlich.

Achtung! Beim Setzen der Haltepunkte ist unbedingt darauf zu achten, daß sie immer auf das erste Byte eines Maschinenbefehls zeigen, da sonst das Programm abstürzen kann.

Beispiele: *G124A,1271,1344*

beginnt die Programmausführung bei 124AH, unterbricht sie bei 1271H und beendet sie bei 1344H.

G,327F

führt ein Maschinenprogramm ab der gegenwärtigen Programmzähleradresse bis Adresse 327FH aus. Da keine Anfangsadresse angegeben ist, muß hinter "G" unbedingt ein Komma stehen.

Befehl: **H**

Mit diesem Befehl lassen sich verschiedene Rechenoperationen bzw. Umrechnungen dezimal/hexadezimal durchführen.

Aufruf: **H Zahl/ASCII-Zeichen**

zeigt den eingegebenen Wert in dezimaler und hexadezimaler Schreibweise und zusätzlich als ASCII-Zeichen an, falls es sich um ein listbares ASCII-Zeichen handelt.

Beispiel: Hier haben alle drei Ausgangswerte das gleiche Ergebnis zur Folge:

H41 (hexadezimale Vorgabe)

H#65 (dezimale Vorgabe)

H'A' (Vorgabe als ASCII-Zeichen)

In allen Fällen erscheint als Ergebnis:

```
0041 #65 'A'
```

wobei der erste Wert hexadezimal, der zweite dezimal und der dritte als ASCII-Zeichen erscheint.

Eine weitere Anwendungsmöglichkeit von "H" besteht darin, gleichzeitig Summe und Differenz zweier Hexadezimalwerte zu berechnen:

Aufruf: **H Zahl 1, Zahl 2**

Es erscheint:

Summe Differenz

Beispiele: *H1000,800*
```
1800 0200
```

H100,200
```
0300 FF00
```

Befehl: **L**

"L" ist ein einfacher Disassembler, der 8080-Quellcode erzeugt. Aufgerufen wird er mit:

Aufruf: **L Anfangsadresse, Endadresse**

Fehlt die Anfangsadresse, wird die Disassemblierung bei der gegenwärtigen Programmzähleradresse begonnen, fehlt die Endadresse, werden insgesamt nur elf Zeilen mit Quelltext aufgelistet.

Beispiel: *L34A7,361F*

disassembliert den Maschinencode zwischen den Adressen 34A7H und 361FH.

Befehl: **M**

Verschiebt einen Speicherbereich und wird folgendermaßen aufgerufen:

Aufruf: **M Anfangsadresse (alt), Endadr. (alt), Anfangsadr. (neu)**

Beispiel: *M1000,1FFF,3000*

verschiebt den Speicherbereich zwischen 1000H und 1FFFH (je einschließlich) und legt ihn ab der Adresse 3000H ab.

Befehl: **R**

Liest nach dem Aufruf von SID eine Datei von Diskette in den Speicher.

Aufruf: **R (Laufwerk:) Dateiname**

lädt dann die Datei und speichert sie ab Adresse 100H.

Hinter dem Dateinamen kann auch noch angegeben werden, mit welchem Versatz die Datei zu laden ist:

R (Laufwerk:) Dateiname, Versatz

Eine Datei soll beispielsweise nicht wie gewöhnlich ab Adresse 100H, sondern ab Adresse 6000H abgelegt werden. Der Versatz beträgt demnach

```
  6000H
 -0100H
---------
  5F00H
```

weshalb die Datei mit dem Befehl

R (Laufwerk:) Dateiname,5F00

zu laden ist.

Beispiele: *R PROBE.TXT*

lädt die Datei PROBE.TXT in den Arbeitsspeicher ab 100H, also ohne Versatz und

R B:PROBE.TXT,1000

lädt dieselbe Datei von Laufwerk B, jedoch mit einem Versatz von 1000H, weshalb sie erst ab Adresse 1100H abgelegt wird.

Befehl: **S**

Mit dem S-Befehl kann der Speicherinhalt byteweise aufgelistet und geändert werden.

Aufruf: **S Anfangsadresse**

Auf dem Bildschirm erscheint jetzt die Anfangsadresse und der Inhalt von deren Speicherzelle. Rechts davon ist der Cursor positioniert und wartet auf Eingabe.

Soll der Inhalt der Speicherzelle unverändert bleiben, ist lediglich die RETURN-Taste zu drücken. Ansonsten ist der neue Wert neben den alten zu schreiben und ebenfalls die RETURN-Taste zu drücken.

In beiden Fällen erscheint dann die nächste Adresse, und der gleiche Vorgang beginnt von neuem. Durch die Eingabe eines Punktes bricht der S-Befehl ab und kehrt in den SID-Befehlsmodus zurück.

Beispiel: *S1234*

listet den Speicherinhalt byteweise ab Adresse 1234H und bietet die Möglichkeit zur Änderung.

Befehl: **T**

arbeitet ein Maschinenprogramm in Einzelschritten ab, wobei nach jedem Schritt die Prozessor-Registerinhalte angezeigt werden. "T" wird in Form von

Aufruf: **T Anzahl**

aufgerufen, wobei die Anzahl der abzuarbeitenden Programmschritte vorzugeben ist. Die Abarbeitung beginnt bei der gegenwärtigen Programmzähleradresse.

Fehlt die Anzahl der Schritte, so wird nur ein Programmschritt ausgeführt.

Beispiel: *T25*

arbeitet 25H (37 dez.) Programmschritte ab, beginnend bei der augenblicklichen Programmzähleradresse, die gegebenenfalls mit dem X-Befehl (siehe unten) einzustellen ist.

Befehl: **U**

Schrittweise Abarbeitung von Maschinenprogrammen wie bei "T", jedoch werden die Registerinhalte nur im Anschluß daran angezeigt.

Beispiel: *U10*

führt, beginnend bei der augenblicklichen Programmzähler-Adresse, 10H (16 dez.) Einzelschritte aus und zeigt erst nach dem 16. Schritt die Prozessorregisterinhalte an.

Befehl: **V**

Anzeige der Speicherwerte NEXT, MSZE, PC und END, die normalerweise nach dem Einladen einer Datei unter SID angegeben werden.

Befehl: **W**

Schreibt einen Speicherbereich auf Diskette und stellt somit eine Alternative zum SAVE-Befehl dar. Im Gegensatz zu SAVE besteht hier keine Gefahr, daß ein Teil der Datei zerstört wird.

Aufruf: **W (Laufwerk:)Dateiname,Anfangsadresse,Endadresse**

Beispiel: *W B:BILANZ.COM,100,4600*

Hier wird der Speicherinhalt zwischen 100H und 4600H unter dem Namen BILANZ.COM auf die Diskette in Laufwerk B geschrieben.

Befehl: **X**

Dient zum Anzeigen und Ändern der Prozessor-Registerinhalte, einschließlich Flags, und wird folgendermaßen aufgerufen:

Aufruf: **X Register oder Flag**

Daraufhin erscheint der Inhalt des betreffenden Registers bzw. Flags. Soll er unverändert bleiben, ist lediglich die RETURN-Taste zu drücken, ansonsten ist ein neuer Wert einzugeben und ebenfalls die RETURN-Taste zu drücken.

Folgende Register bzw. Flags können aufgerufen werden:

A	Akkumulator
B	Doppelregister BC
D	Doppelregister DE
H	Doppelregister HL
S	Stackpointer
P	Programmzähler
C	Carry-Flag
Z	Zero-Flag
M	Minus-Flag
E	Paritätsflag
I	Interdigit-Carry-Flag

Beispiele: *XB*

zeigt das Doppelregister BC an, dessen Inhalt dann wahlweise verändert werden kann oder nicht. Das gleiche gilt mit

XC

für das Carry-Flag, für das allerdings nur die Werte 0 (nicht gesetzt) oder 1 (gesetzt) in Frage kommen.

SID-Dienstprogramme

SID benutzt noch die Dienstprogramme HIST.UTL und TRACE.UTL, die sich ebenfalls auf der Systemdiskette befinden. Sie sind zu aktivieren mit:

SID HIST.UTL

bzw.

SID TRACE.UTL

Das HIST.UTL-Programm ermöglicht im Zusammenhang mit dem SID U- oder T-Befehl die Darstellung eines Histogramms, welches die Häufigkeit der Code-Ausführung innerhalb des angegebenen Speicherbereichs angibt. Mit TRACE.UTL lassen sich die Anweisungen zurückverfolgen, die zu einem bestimmten Breakpoint (Haltepunkt) im Testprogramm geführt haben. Auch dieses Programm bezieht sich auf den SID-U- bzw. T-Befehl.

Weitere Einzelheiten zu diesen SID-Dienstprogrammen sind im "CP/M-SID-Symbolic Instruction Debugger User's Guide" von Digital Research beschrieben.

SUBMIT

Nichtresidenter Standard-CP/M Plus-Befehl (Datei SUBMIT.COM)

Beschreibung: SUBMIT arbeitet eine Befehlsdatei (Stapeldatei) ab, die mehrere CP/M-Befehls- oder Programmaufrufe enthalten kann. Die Befehlsdatei muß die Extension SUB tragen und ist eine reine ASCII-Datei, die mit jedem Texteditor erstellt werden kann.

Vor der Ausführung der einzelnen Befehle legt SUBMIT zunächst die Zwischendatei $$$.SUB an, in die die Befehlsdatei kopiert wird. Bei der Abarbeitung eines jeden Befehls wird dieser dann aus der Zwischendatei herausgenommen, bis schließlich die Datei leer ist. Dann nämlich sind sämtliche Befehle abgearbeitet und die Zwischendatei wird wieder gelöscht.

Die einzelnen Befehle erscheinen während ihrer Ausführung auf dem Bildschirm, und zwar in der gleichen Weise, als wären sie von Hand eingegeben.

Aufruf: **SUBMIT Dateiname SUB-Datei Parameter**

Der Dateiname der SUB-Datei muß ohne die Extension SUB angegeben werden.

Beispiel: Mit Hilfe von SUBMIT sollen die CP/M-Befehle

```
DIR
REN B:LAGER.DAT=LAGER.BAK
BESTAND                    (Ausführung der Datei BESTAND.COM)
SHOW
```

ausgeführt werden. Dazu sind sie zeilenweise mit einem Texteditor (z.B. ED) in eine Datei mit der Extension SUB (z.B. INVENTUR.SUB) zu schreiben. Infolge von

SUBMIT INVENTUR

werden schließlich die einzelnen Befehle der Reihe nach ausgeführt.

Nachfolgend das gleiche Beispiel, jedoch unter Verwendung von Variablen. Für LAGER wird die Variable $1 und für BESTAND die Variable $2 eingesetzt. Dann sähe die Datei INVENTUR.SUB so aus:

```
DIR
REN B:$1.DAT=$1.BAK
$2
SHOW
```

Die Befehsldatei ist dann mit

SUBMIT INVENTUR LAGER BESTAND

auszuführen, wobei in diesem Fall auch andere Dateinamen angegeben werden könnten.

SUBMIT kann auch Parameter an ein CP/M-Dienstprogramm wie PIP oder ED übergeben. Die entsprechende Zeile der Befehlsdatei muß dabei mit dem Zeichen "<" beginnen.

Beispiel: Befehlsdatei mit dem Namen TEST.SUB:

```
PIP
<CON:=$1
<B:=A:SHOW.COM
<
DIR
```

Der Aufruf dieser SUB-Datei geschieht, wenn hier für $1 PROBE.TXT stehen soll, folgendermaßen:

SUBMIT TEST PROBE.TXT

Nach dem Aufruf von PIP wird die Datei PROBE.TXT ($1) auf dem Bildschirm ausgegeben und die Datei SHOW.COM von Laufwerk A nach Laufwerk B kopiert. Anschließend wird PIP wieder verlassen, was in der Befehlsdatei lediglich mit "<" ohne Zusatz angegeben ist und dem Drücken der RETURN-Taste entspricht. Schließlich folgt noch die Auflistung des Directories.

Eine PROFILE.SUB-Datei ist eine Sonderform der SUBMIT-Datei und wird nach dem Booten des CP/M-Plus-Betriebssystems automatisch ausgeführt. Näheres hierzu siehe unter dem Stichwort PROFILE.SUB.

TYPE

Teils residenter, teils nichtresidenter Standard-CP/M Plus-Befehl (Datei TYPE.COM)

Beschreibung: TYPE dient zum einfachen Auflisten von ASCII-Dateien auf dem Bildschirm, wobei die Ausgabe nach dem Vollschreiben einer Bildschirmseite jeweils durch Drücken der RETURN-Taste fortzusetzen ist.

Aufruf: **TYPE (Laufwerk:) Dateiname [Option]**

Beispiel: *TYPE PROBE.TXT*

listet die Textdatei PROBE.TXT seitenweise auf dem Bildschirm, während die Ausgabe mit der Option [NOPAGE] in Form von

TYPE PROBE.TXT [NOPAGE]

fortlaufend ohne Unterbrechung stattfindet.

USER

Residenter Standard-CP/M Plus-Befehl

Beschreibung: USER schaltet zwischen verschiedenen Benutzerbereichen um. Dateien können nämlich nur unter der Benutzernummer aufgerufen werden, in deren Bereich sie abgelegt sind. Die Standardbenutzernummer ist 0.

Aufruf: **USER n**

Dabei gibt "n" die Nummer des Benutzerbereichs (0 bis 15) an.

Beispiel: *USER 5*

schaltet in den Benutzerbereich 5 um.

Eine weitere Möglichkeit besteht darin, lediglich

USER

(ohne Nummer) einzugeben, worauf die Benutzernummer anschließend separat angefordert wird.

XREF

Nichtresidenter Standard-CP/M Plus-Befehl (Datei XREF.COM)

Beschreibung: XREF ist nur im Zusammenhang mit den Assemblern MAC oder RMAC anzuwenden. Aus den, durch die Assemblierung erstellten, PRN- und SYM-Dateien erzeugt XREF eine weitere XRF-Datei unter gleichem Namen. Sie enthält das PRN-Listing mit Zeilennummern versehen und zusätzlich eine Referenz-Tabelle mit den Zeilen, in denen die einzelnen Label definiert und aufgerufen werden (siehe auch Kapitel 8).

Aufruf: **XREF (Laufwerk:)Dateiname**

Hier ist der einfache Dateiname der PRN- und SYM-Datei ohne Extension anzugeben.

Beispiel: *XREF MUSTER*

bildet aus den Dateien MUSTER.PRN und MUSTER.SYM die Datei MUSTER.XRF.

Mit der Option $P wird die XRF-Datei auf dem Drucker ausgegeben.

Beispiel: *XREF MUSTER $P*

Anhang

Steuerzeichen für CP/M-Plus-Befehlszeile

Unter CP/M Plus gelten folgende Steuerzeichen, die durch gleichzeitiges Drücken der CTRL- und der angegebenen Buchstabentaste erzeugt werden:

CTRL-A	Cursor um ein Zeichen nach links
CTRL-B	setzt Cursor an den Zeilenanfang. Wenn er sich dort bereits befindet, springt er zum Zeilenende.
CTRL-C	beendet Programmausführung.
CTRL-E	Physikalischer Carriage Return, ohne Übergabe der Kommandozeile an das System
CTRL-F	Cursor um ein Zeichen nach rechts
CTRL-G	löscht das Zeichen unter dem Cursor.
CTRL-H	löscht das Zeichen links vom Cursor und setzt Cursor um eine Stelle nach links (Backspace).
CTRL-I	Cursor springt zum nächsten Tabulatorstop
CTRL-J	setzt Cursor an den Anfang der Kommandozeile und übergibt das Kommando an das System.
CTRL-K	löscht sämtliche Zeichen von Cursorposition bis Zeilenende.
CTRL-M	Carriage Return
CTRL-P	schaltet Drucker an und ab.
CTRL-Q	nimmt Bildschirm-Scrollen nach CTRL-S wieder auf.
CTRL-R	Alle Zeichen links vom Cursor werden in die nächste Zeile kopiert und im Kommandopuffer abgelegt.
CTRL-S	unterbindet Bildschirm-Scrollen.
CTRL-U	kopiert alle Zeichen links vom Cursor in den Kommandopuffer.

CTRL-W	wiederholt letzte Kommandozeile, wenn die neue Kommandozeile kein Zeichen enthält, anderenfalls wird der Cursor ans Zeilenende gesetzt.
CTRL-X	löscht alle Zeichen links vom Cursor.
CTRL-Z	Textende

ASCII-Code-Tabelle

00H	0	NUL
01H	1	SOH
02H	2	STX
03H	3	ETX
04H	4	EOT
05H	5	ENQ
06H	6	ACK
07H	7	BEL
08H	8	BS
09H	9	HT
0AH	10	LF
0BH	11	VT
0CH	12	FF
0DH	13	CR
0EH	14	SO
0FH	15	SI
10H	16	DLE
11H	17	DC1
12H	18	DC2
13H	19	DC3
14H	20	DC4
15H	21	NAK
16H	22	SYN
17H	23	ETB
18H	24	CAN
19H	25	EM
1AH	26	SUB
1BH	27	ESC
1CH	28	FS
1DH	29	GS
1EH	30	RS
1FH	31	US
20H	32	SP
21H	33	!
22H	34	"
23H	35	#
24H	36	$
25H	37	%
26H	38	&
27H	39	'
28H	40	(
29H	41	)
2AH	42	*
2BH	43	+
2CH	44	,
2DH	45	-
2EH	46	.
2FH	47	/
30H	48	0
31H	49	1
32H	50	2
33H	51	3
34H	52	4
35H	53	5
36H	54	6
37H	55	7
38H	56	8
39H	57	9
3AH	58	:
3BH	59	;
3CH	60	<
3DH	61	=
3EH	62	>
3FH	63	?
40H	64	@ §
41H	65	A
42H	66	B
43H	67	C
44H	68	D
45H	69	E
46H	70	F
47H	71	G
48H	72	H
49H	73	I
4AH	74	J
4BH	75	K
4CH	76	L
4DH	77	M
4EH	78	N
4FH	79	O
50H	80	P
51H	81	Q
52H	82	R
53H	83	S
54H	84	T
55H	85	U
56H	86	V
57H	87	W
58H	88	X
59H	89	Y
5AH	90	Z
5BH	91	[Ä
5CH	92	\ Ö
5DH	93	] Ü
5EH	94	^
5FH	95	_
60H	96	`
61H	97	a
62H	98	b
63H	99	c
64H	100	d
65H	101	e
66H	102	f
67H	103	g
68H	104	h
69H	105	i
6AH	106	j
6BH	107	k
6CH	108	l
6DH	109	m
6EH	110	n
6FH	111	o
70H	112	p
71H	113	q
72H	114	r
73H	115	s
74H	116	t
75H	117	u
76H	118	v
77H	119	w
78H	120	x
79H	121	y
7AH	122	z
7BH	123	{ ä
7CH	124	| ö
7DH	125	} ü
7EH	126	~ ß
7FH	127	

CTRL-/ASCII-Code

CTRL-A	01H	1
CTRL-B	02H	2
CTRL-C	03H	3
CTRL-D	04H	4
CTRL-E	05H	5
CTRL-F	06H	6
CTRL-G	07H	7
CTRL-H	08H	8
CTRL-I	09H	9
CTRL-J	0AH	10
CTRL-K	0BH	11
CTRL-L	0CH	12
CTRL-M	0DH	13
CTRL-N	0EH	14
CTRL-O	0FH	15
CTRL-P	10H	16
CTRL-Q	11H	17
CTRL-R	12H	18
CTRL-S	13H	19
CTRL-T	14H	20
CTRL-U	15H	21
CTRL-V	16H	22
CTRL-W	17H	23
CTRL-X	18H	24
CTRL-Y	19H	25
CTRL-Z	1AH	26

Stichwortverzeichnis

Akustikkoppler, 76
ALT-Taste, 30, 31
AMSDOS, 27, 166
Arbeitskopie, 54
ASCII-Code, 83, 120
Assembler, 128, 131, 136

Backup-Kopie, 88
Banking, 28
BASIC, 12, 167
BASIC-Interpreter, 15
BDOS, 25, 142
BDOS-Aufruf, 134, 146
Befehl, mnemonischer, 131
  nichtresidenter, 50
  residenter, 27, 50
Befehlsdatei, 36
Befehlswort, 134
Befehlszeile, 32
Benutzerbereich, 69
Bereich, Common-, 142
Bereitschaftszeichen, 24
Betriebssystem, 15
Bildschirm, 19
Bildschirmfarbe, 80
Binärwert, 14
BIOS, 25, 27, 47, 77, 142, 156
BIOS-Routine, 156
Bit, 13
Block, 47, 75
Blockbelegungstabelle, 158
Byte, 13

C10CPM3, 168
CCP, 25, 27
Common-Bereich, 142
Concurrent CP/M 86, 18

COPYSYS, 169
CP/M 2.2, 28, 55
CP/M 86, 18
CP/M-68K, 18
CP/M-Software
  Anpassen von, 163
CP/M-Start, 23
CPU, 12
CTRL-C, 30
CTRL-E, 34
CTRL-H, 34
CTRL-M, 34
CTRL-P, 34
CTRL-Q, 34
CTRL-S, 34
CTRL-Steuerzeichen, 34
CTRL-Taste, 31, 33
CTRL-X, 33
CTRL-Z, 86

DATE, 56, 170
Datei, 35
Dateiattribut, 67
Dateiausschnitt, 104
Dateien, aneinanderhängen, 99
Dateiname, eindeutiger, 39
  mehrdeutiger, 39
Datenformat, 44
Datum, 55
Debugging, 119
DEL-Taste, 33
DEVICE, 75, 171
DIN-Tastatur, 77
DIR, 63, 68, 70, 173
DIR-Befehl, 35
Directory, 35, 48, 74, 143
Directory-Hash-Tabelle, 158
Directory-Label, 57, 145
DIRSYS, 68, 173
Disassembler, 128
DISCKIT, 177
DISCKIT3, 43, 177

Disk-Parameter-Block, 157, 159
Disk-Parameter-Header 157
Diskettenlaufwerk, 20
Diskettenwechsel, 30, 50
Drucker, 22, 76
  Laser-, 23
  Matrix-, 22
  Tintenstrahl-, 23
  Typenrad-, 23
DUMP, 119, 178

ED, 85, 110, 113, 132, 179
Editor, 15, 83
Ein-/Ausgabekanal, 76, 101
ERA, 52, 188
EXIT-Taste, 46
Extension, 36
Extent, 41, 48, 143

File-Control-Block, 143
formatieren, 43
FORTH, 17
FORTRAN, 17

GENCOM, 161, 189
Gerätezuordnung, 75
GET, 161, 161, 191
Großschrift, 105
Harddisc, 22
Hardware, 25
HELP, 81, 192
Hexcode, 121
HEXCOM, 136, 193
HIST.UTL, 242

INITDIR, 61, 66, 71, 73, 194
Intel-Hex-Format, 136, 138
J12DCPM3, 195

KEYS.CCP, 80, 115
KEYS.WP, 80
Kleinschrift, 105
Konsole, 19
Kopie,Backup-, 88
  Sicherheits-, 48, 88
kopieren, 43
  Datei, 48
Label, 134
  Directory-, 145
LANGUAGE, 77, 196
Laufwerk, virtuelles, 20, 29, 95
LIB, 197
LINK, 140, 199
löschen, Datei, 52

M-BASIC, 16
MAC, 201
MAC.COM, 136
Mallard-BASIC, 16
Maschinencode, 132
Modem, 76
MP/M, 18
MS-DOS, 18
NLQ 401, 22
Nullseite, 160

Objektcode, 132
Objektmodul, 140

PALETTE, 80, 203
PAPER, 204
Password, 57, 60, 65, 71, 145
PATCH, 205
Peripherie, 101
PIP, 48, 95, 114, 206
PRN-Datei, 137
PRN-Listing, Assembler-, 137
PROFILE.ENG, 115, 214
PROFILE.GER, 116, 214

PROFILE.SUB, 114, 214, 245
Programmiersprache, 16
Programmzähler, 129
Prozessor-Register, 129
PUT, 215

Quelltext, 132

RAM, 13
RAM-Disc, 29
RAM-Floppy, 29
Record, 41, 47, 75
Referenztabelle, 140
Register, Prozessor-, 129
REL-Datei, 140
REN, 51, 217
RMAC, 140, 218
ROM, 13
RS232-Schnittstelle, 163
RSX, 189

SAVE, 123, 219
Schnittstelle
  RS232-, 163
  V24-, 163
Schreibschutz, 66, 67
Sektor, 47, 75
  BOOT-, 54
SET, 58, 59, 63, 65, 66, 68, 220
SET24X80, 226
SETDEF, 116, 227
SETKEYS, 79, 115, 229
SETLST, 230
SETSIO, 231
SHOW, 41, 57, 60, 61, 65, 69, 70, 73, 74, 233
  Sicherheits-Kopie, 48, 88
  SID, 122, 234
  Skewing-Übersetzungstabelle, 157
Software, 25
Speicher, ändern, 125
  ansehen, 125
Speicherbank, 28, 141
Speicherorganisation, 141
Spur, 47
Stapelverarbeitung, 109
STAT, 55
SUBMIT, 109, 243
SUBMIT-Datei, 125
Symboldatei, 138
System-Control-Block, 161
System-Erweiterungresidente, 161
Systemattribut, 67
Systemdisketten, 24
Systemformat, 44

Tastatur, 19
Tastaturbelegung, 77
Terminal, 76
Text, 83
Texteditor, 132
Textpuffer, 88
Timestamps, 57, 60, 65, 71, 145
TPA, 28, 142
TRACE.UTL, 242
Track, 47
Turbo-PASCAL, 17
TYPE, 63, 65, 87, 246

Übersetzungstabelle, Skewing-, 157
Uhrzeit, 55
USER, 69, 247

V24-Schnittstelle, 163
Vendorformat, 44
verifizieren, 43
Verify, 46, 107

WordStar, 19

XREF, 139, 248

Z80A, 12
Zeichensatz, 77
Zeilennummer, 106
Zeropage, 160
Zwischendatei, 88
|CPM, 168

Bücher zu Schneider CPCs

J. Hückstädt
CP/M-2.2-Anwenderhandbuch CPC 464/664/6128
Dezember 1985, 212 Seiten

Wenn Sie glücklicher Besitzer eines Schneider-Computers sind und mehr wissen wollen über das leistungsstarke Betriebssystem CP/M 2.2, dann ist dieses Buch genau das richtige für Sie! Es behandelt CP/M 2.2 nicht nur in seiner allgemeinen Form, wie Sie für sämtliche CP/M-Computer gültig ist, sondern bezieht auch die Hardware der CPC-Computer mit ein.

Best.-Nr. MT 859
ISBN 3-89090-204-9
DM 46,-/sFr. 42,30/öS 358,80

C. Straush
Schneider CPC Grafik-Programmierung
Februar 1986, 225 Seiten

Dieses Buch wendet sich an die Schneider-CPC-Besitzer, die alles über die Grafikfähigkeiten ihres Computers wissen wollen. Es bietet einen umfassenden Überblick über die verschiedenen Anwendungsbereiche der Grafikprogrammierung: zwei- und dreidimensionale Diagrammdarstellungen, Definition und Bewegung von Sprites, Entwurf von Titelgrafiken, Einsatz der Grafik bei der Unterstützung anderer Programme.

• Besonders interessant: ein Sprite-Generator, ein Malprogramm für hochauflösende Grafik, ein Programm zur Erstellung von Titelgrafiken sowie ein universelles Darstellungsprogramm.

Best.-Nr. MT 90182
ISBN 3-89090-182-4
DM 46,-/sFr. 42,30/öS 358,80

J. Hückstädt
Der Schneider CPC 6128
September 1985, 273 Seiten

Dieses Buch ist für jeden CPC-6128-Besitzer eine wertvolle Hilfe, die vielfachen Möglichkeiten dieses bisher einmaligen Computers kennenzulernen und anzuwenden. Der Computerneuling wird Schritt für Schritt in den Umgang mit dem Computer und die BASIC-Programmierung eingeführt, bis er alle notwendigen Kenntnisse besitzt, die mancher Profi bereits mitbringt. Aber an dieser Stelle wird das Programmieren mit dem CPC 6128 erst interessant, nämlich dann, wenn es darum geht, eine eigene Dateiverwaltung aufzubauen oder Grafik und Sound zu programmieren. Weiterhin erfahren Sie alles über CP/M Plus auf dem CPC 6128.

Best.-Nr. MT 90192
ISBN 3-89090-192-1
DM 46,-/sFr. 42,30/öS 358,80

Markt & Technik-Fachbücher erhalten Sie bei Ihrem Buchhändler

Markt&Technik

Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Spitzen-Software für Schneider-Computer

WordStar 3.0 mit MailMerge

Der Bestseller unter den Textverarbeitungsprogrammen für PCs bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.

WordStar/MailMerge für den Schneider CPC 464*, CPC 664*
Bestell-Nr. MS 101 (3"-Diskette)
Bestell-Nr. MS 102 (5¼"-Diskette im VORTEX-Format)
WordStar/MailMerge für den Schneider CPC 6128
Bestell-Nr. MS 104 (3"-Diskette)
WordStar/MailMerge für den Schneider Joyce PCW 8256
Best.-Nr. MS 105 (3"-Diskette)

Hardware-Anforderungen: Schneider CPC 464*, CPC 664*, CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle
* Der Standard-Speicherplatz beim CPC 464/664 erlaubt ohne Speichererweiterung Blockverschiebe-Operationen nur bedingt und Simultan-Drucken gar nicht.

Dieses Programm kostet
DM 199,– inkl. MwSt. Unverbindliche Preisempfehlung

Markt&Technik
Schneider CPC-Software

MicroPro
WordStar 3.0
mit MailMerge für den Schneider CPC 464/664

3" Schneider-Format

Und dazu die weiterführende Literatur:

Mit diesem Buch haben Sie eine wertvolle Ergänzung zum WordStar-Handbuch: Anhand vieler Beispiele steigen Sie mühelos in die Praxis der Textverarbeitung mit **Wordstar** ein. Angefangen beim einfachen Brief bis hin zur umfangreichen Manuskripterstellung zeigt Ihnen dieses Buch auch, wie Sie mit Hilfe von MailMerge Serienbriefe an eine beliebige Anzahl von Adressen mit persönlicher Anrede senden können.

Best.-Nr. MT 779
ISBN 3-89090-180-8

DM 49,–

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **WordStar-Programm** für Ihren Schneider-Computer fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler.

Markt&Technik

Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

Spitzen-Software für Schneider-Computer

dBASE II, Version 2.41

dBASE II, das meistverkaufte Programm unter den Datenbanksystemen, eröffnet Ihnen optimale Möglichkeiten der Daten- u. Dateihandhabung. Einfach u. schnell können Datenstrukturen definiert, benutzt und geändert werden. Der Datenzugriff erfolgt sequentiell oder nach frei wählbaren Kriterien, die integrierte Kommandosprache ermöglicht den Aufbau kompletter Anwendungen wie Finanzbuchhaltung, Lagerverwaltung, Betriebsabrechnung usw.

dBASE II für den Schneider CPC 464*, CPC 664*
Bestell-Nr. MS 301 (3"-Diskette)
Bestell-Nr. MS 302 (5¼"-Diskette im VORTEX-Format)
dBASE II für den Schneider CPC 6128
Bestell-Nr. MS 304 (3"-Diskette)
dBASE II für den Schneider Joyce PCW 8256
Best.-Nr. MS 305 (3"-Diskette)

Hardware-Anforderungen: Schneider CPC 464*, CPC 664*, CPC 6128 oder Joyce, beliebiger Drucker mit Centronics-Schnittstelle
* dBASE II für den Schneider CPC 464/664 ist lauffähig mit der VORTEX-Speichererweiterung auf 128 KByte. Diese erhalten Sie direkt bei der Firma VORTEX oder bei Ihrem Computerhändler.

Dieses Programm kostet
DM 199,– inkl. MwSt. Unverbindliche Preisempfehlung

Und dazu die weiterführende Literatur:

Zu einem Weltbestseller unter den Datenbanksystemen gehört auch ein klassisches Einführungs- und Nachschlagewerk! Dieses Buch von dem deutschen Erfolgsautor Dr. Peter Albrecht begleitet Sie mit nützlichen Hinweisen, die nur von einem Profi stammen können, bei Ihrer täglichen Arbeit mit **dBASE II.** Schon nach Beherrschung weniger Befehle ist der Einsteiger in der Lage, Dateien zu erstellen, mit Informationen zu laden und auszuwerten.

Best.-Nr. MT 90377
ISBN 3-89090-377-0
DM 49,–

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **dBASE II-Programm** für Ihren Schneider-Computer fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

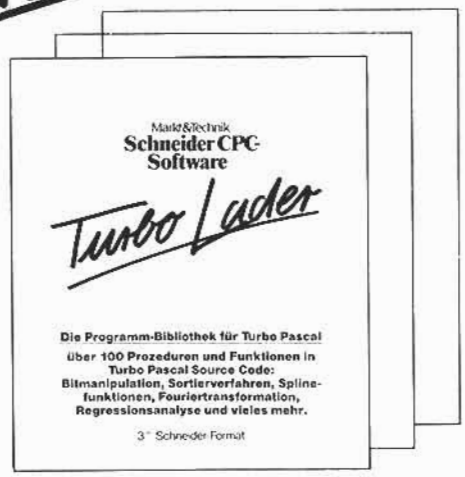
Diese Markt & Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler.

Markt&Technik

Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München

JETZT AUF SCHNEIDER-COMPUTERN:

Turbo Lader



DIE PROGRAMM-BIBLIOTHEK FÜR TURBO PASCAL®

TURBO-Lader-Grundpaket

Das TURBO-Lader-Grundmodul ist eine umfangreiche Programm-Bibliothek für den TURBO-Pascal-Programmierer. Sie umfaßt zahlreiche ausführlich dokumentierte Prozeduren und Funktionen, die der Profi zur schnellen Lösung seiner Programmieraufgaben verwenden kann.

Das TURBO-Lader-Grundpaket erfordert den TURBO-Pascal-Compiler. Es ist lieferbar auf 3"- und 5¼"-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128 und Joyce.

3"-Disk. Best.-Nr. MS 413
5¼"-Disk. Best.-Nr. MS 415

DM 138,-* · inklusive MwSt.

TURBO-Lader Business

TURBO-Lader Business umfaßt einen komfortablen Bildschirm-Maskengenerator und eine professionelle Dateiverwaltung. Der Maskengenerator gibt dem Pascal-Programmierer ein Werkzeug zur einfachen Bearbeitung von Bildschirm-Masken in die Hand.

TURBO-Lader Business erfordert den TURBO-Pascal-Compiler und das TURBO-Lader-Grundpaket. Es ist lieferbar auf 3"- und 5¼"-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128 und Joyce.

3"-Disk. Best.-Nr. MS 423
5¼"-Disk. Best.-Nr. MS 425

DM 148,-* · inklusive MwSt.

TURBO-Lader Science

TURBO-Lader Science ist eine Sammlung technisch/wissenschaftlicher Funktionen und professioneller statistischer Verfahren für die Bereiche Medizin, Betriebs- und Volkswirtschaft, Technik und Naturwissenschaften.

TURBO-Lader Science erfordert den TURBO-Pascal-Compiler und das TURBO-Lader-Grundpaket. Es ist lieferbar auf 3"- und 5¼"-Disketten und lauffähig auf dem Schneider CPC 464, CPC 664, CPC 6128 und Joyce.

3"-Disk. Best.-Nr. MS 433
5¼"-Disk. Best.-Nr. MS 435

DM 189,-* inklusive MwSt.

Übrigens können Sie auch folgende Turbo-Pascal-Produkte für Schneider CPC und Joyce bei Markt&Technik beziehen:
Turbo Pascal 3.0, Turbo Pascal 3.0 mit Grafikunterstützung,
Turbo Tutor (deutsch), Turbo Tutor (englisch), Turbo Graphix Toolbox, Turbo Toolbox.

Markt&Technik

Unternehmensbereich Buchverlag
Hans-Pinsel-Straße 2, 8013 Haar bei München