Betriebssystem

In diesem Fachgebiet wird das Christiani-Betriebssystem beschrieben. Seine Anwendung wird zum größten Teil in den Fachgebieten PROGRAMMIERUNG und TESTEN ausführlich besprochen und vor allem ausprobiert. Das Betriebssystem besteht aus verschiedenen Teilen, von denen zunächst der Texteditor betrachtet werden soll.

Editor

Der Editor erscheint geeignet, ein erstes Kennenlernen des Betriebssystems zu vermitteln. Dabei ist die kurze Bezeichnung "Editor" eine sanfte Untertreibung, denn dieser Editor ist in Wirklichkeit fast schon ein richtiges Textverarbeitungs-Programm.

Auf dem Bildschirm steht noch die Systemmeldung mit dem blinkenden Cursor, der zu weiterer Eingabe auffordert. Geben Sie mit der Tastatur das Wort

Edit

ein und drücken Sie die rechts liegende Funktionstaste RETURN (sie ist mit "CR" oder "Ret" bezeichnet). Auf dem Bildschirm steht jetzt nur noch der blinkende Cursor.

Ehe mit dem Editor gearbeitet wird, kurz noch ein Blick auf das Eingabewerkzeug, die Tastatur. Bild B1.1 zeigt

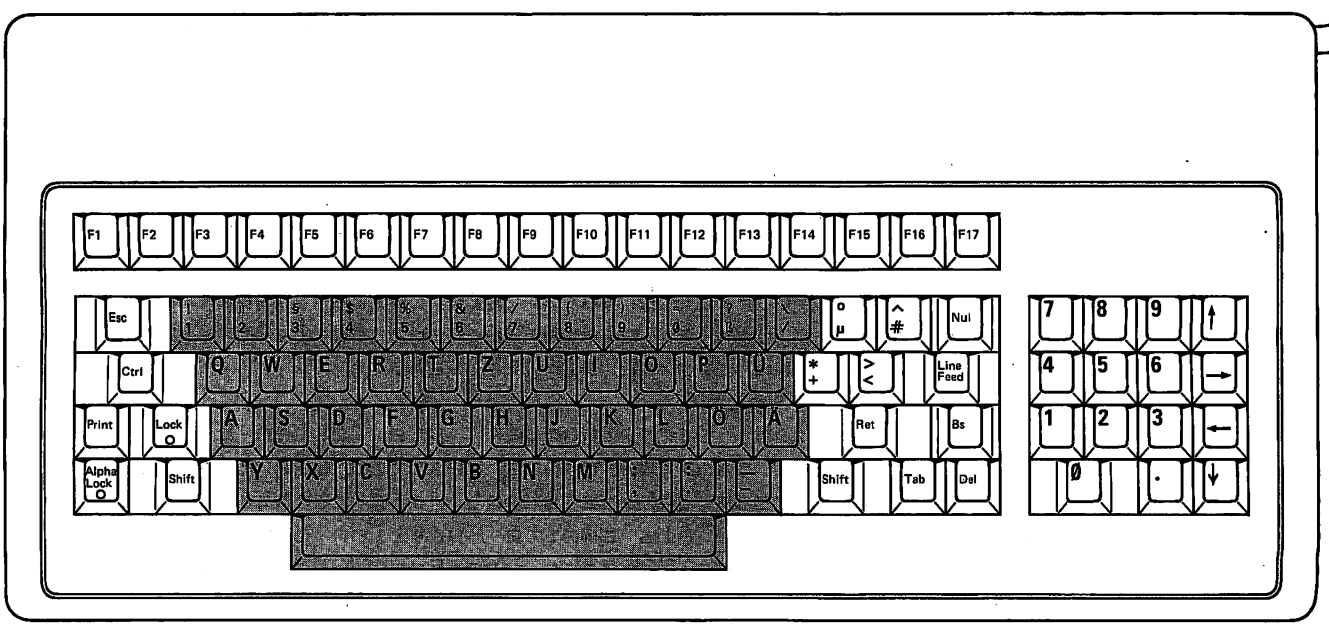


Bild B 1.1
Der einer Schreibmaschinentastatur entsprechende Teil ist grau unterlegt.

ein Beispiel aus der großen Anzahl von Tastaturen. Die Tastatur Ihres Systems muß nicht genau die gleiche sein. Jedenfalls hat sie aber wie das gezeigte Beispiel eine sogenannte deutsche Tastatur (nach DIN 2137, Teil2). Es handelt sich im Grund um eine ganz normale Schreibmaschinentastatur, um die herum eine Reihe von zusätzlichen Tasten angeordnet sind. Außer dem rechtsliegenden Block mit Zahlen, der den Tasten einer Rechenmaschine ähnelt, sind es Funktionstasten, die jeweils dann erklärt werden, wenn sie gebraucht werden.

Vorab sei auf diejenigen Funktionstasten hingewiesen, die auch bei einer Schreibmaschine vorhanden sind. Rechts und links der farbig unterlegten alphanumerischen Tastatur liegen in der untersten Reihe zwei mit "Shift" bezeichnete Tasten, mit ihnen geschieht die Umschaltung auf Großbuchstaben. Diese läßt sich feststellen, und zwar mit der links unten liegenden Taste "Alpha Lock" nur für die Buchstabentasten oder mit der Taste "Lock" für alle Tasten. Damit man sieht, daß die Shifttasten dauernd eingeschaltet sind, leuchtet bei den meisten Tastaturen jeweils eine rote LED in diesen Lock-Tasten.

Die mit "CR" oder "Ret" bezeichnete RETURN-Taste entspricht der Einrichtung der Schreibmaschine, mit der das Weiterrücken auf den Anfang einer neuen Zeile bewirkt wird (CR ist die Abkürzung des englischen"carriage return", das bedeutet Wagen-Rücklauf).

Auf dem Bildschirm blinkt immer noch der Cursor, er fordert auf, nun endlich etwas zu schreiben. Probieren wir es in der Praxis.

Von dieser Zeile ab ist der Lehrbrieftext mit dem Texteditor des Betriebssystems geschrieben worden. Um dies auch in der Druckwiedergabe zu dokumentieren, steht der Text hier so, wie er auf dem Bildschirm bzw. auf einem angeschlossenem Nadeldrucker aussieht.

Zuerst fällt gegenüber dem normalen Druck auf, daß die Zeilenlängen nicht gleichmäßig sind. Es ist kein Blocksatz, wie der Fachmann dazu sagt, sondern Flattersatz. Die Zeilen haben aber eine größte Länge, diese Begrenzung läßt sich mit zwei Funktionen einstellen.

Wenn man nacheinander die Tasten Esc und M drückt, dann wird die Position, an welcher der der Cursor gerade steht, als rechter Rand programmiert. Als zweite Funktion bewirkt die anschließende Eingabe von Esc W, daß Worte, die über den programmierten rechten Rand hinaus

```
gehen, automatisch in die nächste Zeile gesetzt
werden. Hier, beim großen B von Beginn, haben
wir das ausprobiert, und schon
kommt das Wort "kommt" in die
nächste Zeile, weil es über den
Rand reicht. Um entstehende große
Lücken zu vermeiden, kann man das
Wort abteilen, muß aber nach dem
Strich noch einen Zwischenraum
eingeben, damit nicht das ganze
Wort samt Abteilstrich in die
nächste Zeile rutscht. Als Bei-
spiel steht das Wort "Bei-spiel"
am Anfang dieses Satzes.
```

Den Umgang mit den beiden Funktionen ESC M und ESC W müssen Sie selber probieren. Auch die längsten Erklärungen ersetzen nicht die praktische Erfahrung. Schreiben Sie etwa eine halbe Seite Text auf den Bildschirm, stellen Sie mit ESC M und ESC W etwa dreiviertel Breite ein. Zur Erklärung der übrigen Funktionen wird dieser Text gebraucht. Wenn Ihnen nichts einfällt, schreiben Sie ein Stück Lehrgangstext ab.

Wenn ESC M und ESC W nicht eingeschaltet wurden, dann läuft der eingegebene Text über die ganze Bildschirmbreite und läuft darüber hinaus, was durch ein Pluszeichen angezeigt wird. Wenn Sie eine Zeile mit RETURN abgeschlossen haben und somit zur nächsten gelangen, zeigt dies eine eckige Klammer (<) an. Probieren Sie auch das in einer Zeile aus. Die beiden besprochenen Funktionen werden durch wiederholte Eingabe ein- und ausgeschaltet, dabei wird jedesmal über dem Text eine Weile angezeigt, ob sie eingeschaltet sind oder nicht, also z.B. "WORD-WRAP ON", wenn ESC W eingeschaltet ist. W= Word-wrap läßt sich mit Wort-Einordnen (wörtlich Einhüllen) übersetzen, M=Margin bedeutet Rand.

An dieser Stelle kann gleich noch eine weitere Funktion mit ESC ausprobiert werden: drücken Sie ESC U, und schon sind in Ihrem Text alle Umlaute Ö,Ä und Ü verschwunden, an ihrer Stelle stehen eckige Klammern und andere Zeichen. Mit ESC U wird die Tastatur auf den amerikanischen Zeichensatz umgeschaltet. Mit ESC D schalten Sie wieder auf den deutschen Zeichensatz um, der zunächst gebraucht wird.

Cursorbewegungen

Vielleicht haben Sie sich vorher beim Schreiben des Textes vertippt und wollten noch einmal zurück in der Zeile

oder eine ganze Zeile neu schreiben. Dabei haben Sie festgestellt, daß die eingegebenen Zeichen immer nur an der Stelle des Cursors erscheinen. Auch zum Einstellen des rechten Randes mit ESC M muß der Cursor bewegt werden. Dafür sind die sogenannten Cursor-Tasten da, kenntlich an den aufgebrachten Pfeilen, welche die Bewegungsrichtung des Cursors beim Betätigen der jeweiligen Taste angeben. Bei der abgebildeten Tastatur stehen sie am äußersten rechten Rand.Probieren Sie die Cursortasten einmal aus. Dabei werden Sie merken, daß diese Tasten (wie die meisten anderen auch) Repeatfunktion haben. Das heißt, wenn man eine Taste länger als ca.2 Sekunden gedrückt hält, wiederholt sich das gewählte Zeichen oder die Funktion fortlaufend. Im vorliegenden Fall läuft der Cursor in der gewählten Richtung weiter, bis kein Text mehr da ist. Bei manchen Tastaturen haben die Tasten mit den waagerechten Pfeilen andere Funktionen, nämlich TAB und DELETE, siehe Seite B5 und B6.

Mit den vier Cursortasten läßt sich der Cursor also an jede Stelle des geschriebenen Textes bringen, um dort Korrekturen usw. auszuführen. Außerdem haben diese Tasten bei gleichzeitig gedrückter Shifttaste noch folgende Funktionen: Pfeil nach rechts oder nach links ergibt wortweises Weiterwandern des Cursors, Pfeil nach oben oder unten ergibt seitenweises Weiterwandern des Cursors (Umblättern, Scrolling).

Der Cursor läßt sich aber nicht nur mit den beschriebenen Funktionstasten bewegen. Diese sind bei Auflistungen oder tabellenartigem Text, wie z.B. Programmen nützlich. Die nachfolgend vorgestellten Funktionen wären da zu umständlich, sind hingegen bei reiner Textverarbeitung angebracht, weil sie inmitten der alphanumerischen Tastatur beqemer erreichbar liegen.

Es wird dabei die Taste mit der Controllfunktion benutzt, sie liegt links oben unterhalb der ESC-Taste und ist meist mit Ctrl bezeichnet. Wichtig ist, daß die CTRL-Taste immer gleichzeitig mit der dazugehörigen Zeichentaste betätigt wird, im Gegensatz zu den ESC-Funktionen, bei denen die Taste ESC und die jeweilige Zeichentaste nacheinander gedrückt werden.Die Controllfunktionen zur Bewegung des Cursors sehen so aus:

CTRL-S	Zeichen links	CTRL-D	Zeichen rechts
CTRL-A	Wort links	CTRL-F	Wort rechts
CTRL-E	Zeile hoch	CTRL-X	Zeile runter

Auch hier wieder: Probieren Sie's aus! Drücken Sie gleichzeitig Ctrl und D : der Cursor geht um ein Zeichen nach rechts, wie beim Drücken der Funktionstaste mit dem

entsprechenden Pfeil. Auch bei diesen Tasten gibt es wieder die Repeatfunktion. Wenn Sie die Anordnung der Controllfunktionen innerhalb der Schreibtastatur betrachten, erkennen Sie ein leicht merkbares System, das der Richtung der Cursorbewegung entspricht:

```
    E
A   S   D   F
    X
```

Zuvor war bereits die Rede von den Scrollfunktionen, also dem seitenweisen vor- oder zurückblättern im Text (im Textspeicher). Auch dafür gibt es Controllfunktionen innerhalb der Schreibtastatur:

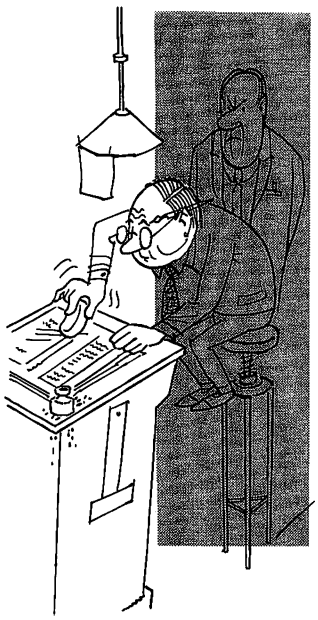
CTRL-C	Seite rückwärts	CTRL-R	Seite vorwärts
CTRL-W	Zeile rückwärts	CTRL-Y	Zeile vorwärts

Die Anordnung dieser Funktionstasten fügt sich sinnfällig zu den anderen:

```
  W   E   R
A   S   D   F
  Y   X   C
```

Der Unterschied zwischen CTRL-E und CTRL-W besteht darin, daß im ersten Fall der Cursor um eine Zeile nach oben bewegt wird, im zweiten Fall wird der Text (gewissermaßen unter dem Cursor hindurch) um eine Zeile nach unten geschoben. Entsprechend verhält es sich bei CTRL-X und CTRL-Y.Eine weitere Möglichkeit, den Cursor z bewegen, ergibt sich mit der Funktion CTRL-I. Der Cursor rückt um 8 Zeichen nach rechts, es ensteht ein Zwischenraum. Der gleiche Effekt wird bei manchen Tastaturen (z.B. der abgebildeten) mit einer Taste "Tab" erreicht, bei anderen ist es die Taste mit dem nach rechts weisenden Pfeil. Diese Tabulator-Funktion ist sehr hilfreich beim Schreiben von Programm-Listings, wir kommen noch darauf zurück. An dieser Stelle der wiederholte Rat: Probieren Sie alle diese Funktionen aus. Gerade das Wissen um die Möglichkeiten des recht komfortablen Texteditors hilft sehr, wenn er später als Werkzeug bei der Assemblerprogrammierung dient. Abgesehen davon, wenn ein Drucker angeschlossen wird, dann können Texte, Briefe usw. geschrieben werden. Und schließlich: wenn Sie einmal mit Diskettenlaufwerken arbeiten und sich ein großes Textverarbeitungssystem als Softwarepaket gekauft haben, dann werden Sie dort viele gleiche und ähnliche Funktionen wiedererkennen.

Löschfunktionen



Ohne Radiergummi geht es beim Schreiben nicht. Die erste Möglichkeit dieser Art werden Sie sicher schon benutzt haben. Wenn nämlich der Cursor durch die Bewegungsfunktionen auf ein vorhandenes Zeichen gesetzt und danach eine andere Zeichentaste gedrückt wird, dann ist das ursprüngliche Zeichen gelöscht, überschrieben durch das Neue.

Der eigentliche Radiergummi ist aber die Taste DELETE (Löschen), die mit Del bezeichnet ist. Beim Betätigen von Del wird das links neben dem Cursor stehende Zeichen ausgelöscht. Wenn der Cursor ganz links am Zeilenanfang steht, wandert er durch Del an das Ende der vorangegangenen Zeile, es wurde das in der Schreibfolge links vom linken Rand stehende Zeichen für Wagenrücklauf und neue Zeile (Taste Ret) gelöscht. Bei manchen Tastaturen ist übrigens auch die Taste mit dem nach links zeigenden Pfeil mit der Delete-Funktion belegt.

Eine weitere Löschfunktion heißt CTRL-G. Damit wird das Zeichen gelöscht, auf dem der Cursor steht. Mit CTRL-T wird ein ganzes Wort rechts vom Cursor gelöscht und mit CTRL-Z schließlich eine ganze Zeile. Bei der anschließenden Vorstellung der Blockoperationen werden Sie erfahren, daß auch ganze Textabschnitte gelöscht werden können.

Einfügen

Sozusagen als Gegenteil vom Löschen ist die Möglichkeit, etwas in den geschriebenen Text einzufügen, eine große Hilfe bei der Textverarbeitung. Mit der Funktion ESC I werden neu eingegebene Zeichen in den vorhandenen Text eingefügt. Der Text wird dabei nach rechts weitergeschoben. Ein Pluszeichen (+) am rechten Rand zeigt an, wenn dabei dieser Rand überschritten wird. Um den "verschwundenen" Text wieder sichtbar zu machen, muß an der Stelle vor dem Pluszeichen (oder bei programmiertem rechten Rand an dessen Stelle) ein Wagenrücklauf (Ret), ebenfalls mit ESC I, eingefügt werden.

Die Einfügungsfunktion wird mit wiederholter Eingabe von ESC I wieder ausgeschaltet. Es erscheint über dem Text für einige Sekunden die Anzeige "INSERT ON" oder "INSERT OFF". Denken Sie daran: bei Funktonen mit ESC die Taste Esc und das zugehörige Zeichen nacheinander betätigen, im Gegensatz zu den Funktionen mit CTRL, wo die Taste Ctrl und das zugehörige Zeichen gleichzeitig zu betätigen sind !

Auch hier der gleiche Hinweis wie bei den Löschfunktionen: bei den anschließend besprochenen Blockoperationen werden Sie sehen, wie ganze Textabschnitte in den vorhandenen Text eingefügt werden können.

Abspeichern

Um bei dem Vergleich mit dem Bleistift-Schreiben zu bleiben: den Radiergummi und die Einfügeklammer kennen Sie jetzt. Aber getrost nach Hause tragen, was man schwarz auf weiß besitzt - haben Sie bei der Besprechung des Texteditors mal eine Pause gemacht und den Computer ausgeschaltet? Dann durften Sie zähneknirschend einen neuen Text zum Ausprobieren der Funktionen eintippen, der alte war weg. Selbstverständlich läßt sich der Inhalt des Textspeichers auf einen nichtflüchtigen externen Speicher übertragen und von dort bei Bedarf wieder einlesen.

Für die Abspeicherung auf dem angeschlossenen Audio-Cassettenrecorder und das Wieder-Einlesen von dort in den Textspeicher hat unser Betriebssystem entsprechende Funktionen.

Drücken Sie nacheinander die Tasten Esc und S, es erscheint die Anzeige:

*SAVE?

Das System möchte einen Namen wissen für die Datei, die auf der Cassette mit ESC S abgespeichert werden soll. Geben Sie eine vierstellige Zahl ein, nennen Sie die Datei mit ihrem Text z.B. 0001.

Stellen Sie den Recorder auf Aufnahme und starten Sie den Abspeichervorgang mit Ret.Die technischen Gegebenheiten sind in den Unterlagen zur Hardware ausführlich beschrieben.

Mit der Funktion ESC A können die abgespeicherten Daten wieder in den Textspeicher des Computers eingelesen werden.Drücken Sie nacheinander die Tasten Esc und A,es erscheint die Anzeige:

*APPEND?

Auch hier wird wieder der Name der Datei einzugesetzt und danach die Taste CR bzw. Ret gedrückt.

Auf diese Weise haben Sie Ihren Text und später die mehr oder weniger langen Programme sichergestellt und brauchen sie nicht mehr jedesmal neu einzutippen.

B
8

Blockoperationen

Unter dem Begriff "Block" wird an dieser Stelle einfach ein Stück Text, also einige Wörter, eine oder mehrere Zeilen, eine ganze Seite usw. verstanden. Solche beliebig ausgesuchten Textabschnitte können mit den Funktionen, die zusammenfassend Blockoperationen genannt werden, markiert und manipuliert werden.

Holen Sie sich Ihren Text mit ESC A in den Textspeicher und auf den Bildschirm, wenn er nicht noch dort steht. Suchen Sie sich ein Textstück aus, das als Block zum Ausprobieren der Blockoperationen dienen soll, vielleicht drei oder vier Zeilen mitten im Text.

Setzen Sie den Cursor auf den ersten Buchstaben des Blocks und geben Sie ESC B ein. Sofort wird der Blockbeginn, der ja mit dem Cursor festgelegt war, mit einem B, zwischen eckigen Klammern stehend, markiert. Ab dieser Markierung beginnt der Block.

Bringen Sie danach den Cursor an die Stelle, an welcher der Block enden soll. Dabei ist zu beachten, daß der Block eine Stelle vor der Cursorposition endet.

Geben Sie dann ESC K ein. Das Blockende wird mit einem zwischen eckigen Klammern stehenden K markiert.

Wenn bei der Blockmarkierung irgendetwas nicht so gelaufen ist, wie Sie wollten, wenn Sie es sich anders überlegt haben -, macht nichts, wiederholen Sie die Prozedur mit ESC B und ESC K, die vorhergehende Markierung verschwindet automatisch.

Der Texteditor hat insgesamt fünf Blockoperationen auf Lager:

ESC B Blockanfang markieren
ESC K Blockende markieren

ESC C Block an Cusorposition kopieren
ESC V Block an Cursorposition verschieben
ESC Z Block löschen

Blockanfang und Blockende sind markiert, setzen Sie den Cursor ein paar Zeilen weiter und geben Sie ESC C ein.

Zweierlei ist wichtig:
der Block steht noch an seinem ursprünglichen Platz und ist außerdem an der gewünschten und durch den Cursor markierten Stelle in den dort stehenden Text eingefügt worden, es wurde nichts überschrieben dabei.

Markieren Sie sich mit ESC B und ESC K einen anderen Block, bringen Sie den Cursor auf eine Position, auf die der Block verschoben werden soll und geben Sie ESC V ein. Auch bei dieser Funktion ist zweierlei zu beachten: der Block wird an der gewünschten Stelle eingefügt, etwa dort stehender Text wird nicht überschrieben. Daneben verschwindet der Block von seinem ursprünglichen Platz. Sicher wird bei reiner Textverarbeitung diese Funktion mehr gebraucht: man nimmt einen Absatz von seinem Platz fort und fügt ihn woanders im Text ein.

Geben Sie schließlich noch ESC Z ein, damit wird der Block gelöscht. Das und die Tatsache, daß Ihr Text vor lauter Versuchen mit den Blockoperationen jetzt ganz schön durcheinander und nicht mehr vollständig ist, macht nichts. Holen Sie sich mit ESC A den unveränderten Text noch einmal in den Textspeicher und auf den Bildschirm.

Fehlermeldung

Um die Fehlermeldung des Texteditors zu erzeugen, wird ein möglichst großer Textblock benötigt. Markieren Sie mit ESC B und ESC K Ihren gesamten Text als Block. Setzen Sie den Cursor mit der Taste Ret unterhalb des Textes und geben Sie ESC C ein. Jetzt steht der Text zweimal hintereinander im Textspeicher. Mit CTRL-C blättern Sie weiter im Textspeicher bis an das Ende des kopierten Blocks. Mit einem weiteren ESC C wird der Block ein zweitesmal kopiert.

Treiben Sie diese "Vervielfältigung" des Textes solange, bis auf dem Bildschirm folgende Anzeige erscheint:

FULL

Das englische Wort Full bedeutet "voll". Was ist hier voll? Mit dieser Frage werden wir uns in der Fortsetzung dieses Fachgebiets befassen, davor aber ist ein Ausflug in das Fachgebiet SYSTEMBESCHREIBUNG notwendig. Wenn Sie die nächsten Seiten gleich anschließend durcharbeiten, dann verlassen Sie den Editor mit der Eingabe von ESC E, es erscheint wieder das Promptzeichen (<) des Betriebs-

systems.Lassen Sie das Gerät eingeschaltet, um den Tatbestand, der zur Fehlermeldung "FULL" führte, im Textspeicher zu erhalten.

Zusammenfassung

Der Texteditor hat weitgehend die Eigenschaften eines Textverarbeitungssystems. Er wird mit dem Kommando EDIT aufgerufen und die eingegebenen Zeichen werden an der Stelle des Cursors auf dem Bildschirm sichtbar bzw. in eventuell schon vorhandenen Text eingefügt. Mit dem Funktionen zur Cursorsteuerung kann der Cursor waagerecht und senkrecht, zeichenweise,wortweise, zeilenweise und seitenweise bewegt werden. Neue Eingaben können vorhandene überschreiben oder eingefügt werden. Falsche Eingaben können zeichenweise, wortweise, zeilenweise oder in markierten Textblöcken gelöscht werden. Textblöcke können innerhalb des Textes beliebig markiert und dann verschoben, kopiert oder gelöscht werden.

Fertige Texte können auf externe Speicher (Cassette, Diskette) abgespeichert und wieder zurück in den Textspeicher geladen werden. Ferner besteht die Möglichkeit, zwischen deutschem und amerikanischem Zeichensatz umzuschalten.

Fragen:

1) Warum erscheint nach mehrmaligem Block-Kopieren die Fehlermeldung "FULL"?

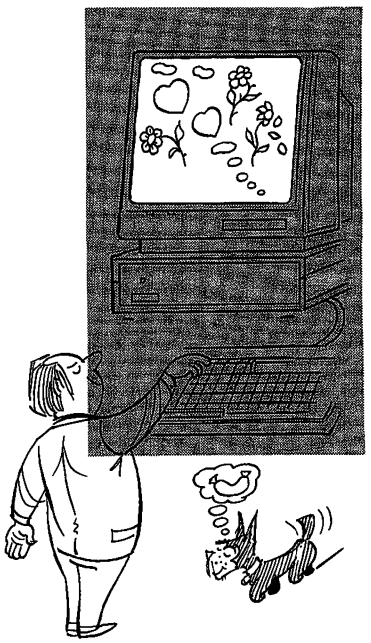
2) Mit der Funktion ESC I fügen Sie in eine Textzeile einige Wörter ein.Dabei läuft der Text am Zeilenende über, ein Pluszeichen zeigt dies an.Wie machen Sie diesen Überlauf, der im Textspeicher ja noch vorhanden ist, wieder auf dem Bildschirm sichtbar ?

3) Welche Taste wird bei der Auflistung von Programmen unentbehrlich werden ?

Die Antworten zu diesen Fragen am Ende der einzelnen Fachgebiete innerhalb eines Heftes finden Sie auf den Seiten mit der Griffmarke G (die Antworten zu den Fragen auf dieser Seite z.B. auf Seite G4).Sie sollten sich aber zuerst einmal Ihre Antwort überlegen und dann erst nachprüfen, ob Sie damit richtig liegen.

Speicherverwaltung

Als ersten Teil des Betriebssystems haben Sie den Texteditor kennengelernt. Sie können damit Briefe schreiben, irgendwelche Texte und natürlich auf recht komfortable Art Programme entwerfen (das kommt noch dran). Irgendwann stösst man dann, wenn das "Geschriebene" immer mehr wird, an eine Grenze: der Textspeicher ist zu klein. Damit Sie keinen Roman verfassen müssen, um soweit zu kommen, haben wir auf Seite B9 mit wiederholtem Block-Kopieren das Betriebssystem veranlaßt, seines Amtes zu walten: nämlich einzugreifen, ehe es womöglich selber an seinem Platz im Speicher mit Text überschrieben wird, die Fehlermeldung "FULL" auszugeben.



Dies ist also eine der wichtigen Aufgaben des Betriebssystems: die Speicherverwaltung. Sie sind nun sicher neugierig, wie groß der Textspeicher ist, und vor allem, wie man ihn vergrössern kann, damit ein sehr langes Programm weitergeschrieben werden kann.

Beide Fragen können beantwortet werden. Zunächst kann die momentane Größe der variablen Speicherbereiche festgestellt werden. Geben Sie das Kommando MEMORY ein und auf dem Bildschirm erscheint folgende Darstellung:

```
>memory
TPA:   FREE=7B00   USED=7B00
SYM:   FREE=0000   USED=0000
TXT:   FREE=000D   USED=04EC
```

Die absoluten Werte (in sedezimaler Schreibweise dargestellt) sind hier nur als Beispiel eingesetzt und können bei Ihrem Computer etwas abweichen. Die Verteilung ist aber die gleiche. Da gibt es eine sehr große TPA, die bisher noch garnicht gebraucht wurde. Das Betriebssystem kann allerdings nicht wissen, welcher Teil der TPA benutzt wird. Es nimmt daher an, daß z.B. ein Anwenderprogramm die gesamte TPA ausfüllt und gibt immer bei FREE die gleiche Zahl wie bei USED aus.

Hinter SYM stehen Nullen, für die Symboltabelle des Assemblers ist noch kein Speicherplatz reserviert. Klar, Sie haben den Assembler bis jetzt noch nicht aufgerufen.

Im eigentlich interessierenden Textspeicher sind wahrscheinlich auch bei Ihrem Computer noch einige Byte frei. Dabei hatte die Fehlermeldung *FULL* doch einen vollen Textspeicher angezeigt.Das war weise Voraussicht, denn Sie hatten ja versucht, einen ganzen Textblock zu kopie-

ren. Der Editor hat nicht einfach draufloskopiert, sondern vorher erst mal nachgerechnet, ob noch genügend Platz im Speicher vorhanden ist, und das war nicht der Fall.

Die Belegung der einzelnen Speicherbereiche läßt sich auch getrennt darstellen. Geben Sie MEMORY TPA ein, auf dem Bildschirm erscheint:

```
>memory tpa
FREE=7B00  USED=7B00
```

Nach der Eingabe von MEMORY SYM erscheint:

```
>memory sym
FREE=0000  USED=0000
```

Um den Textspeicher allein zu betrachten, genügt die Eingabe von MEMORY TXT:

```
>memory txt
FREE=0000  USED=04EC
```

Was zuvor interessierte, war die Frage, wie es nun weitergeht mit dem Text- oder Programmschreiben nach der *FULL*-Meldung. Mit dem Kommando PACK können Sie noch mehr in den Textspeicher hinein 'PACK'-en, besser gesagt, den Textspeicher vergrößern. Vielleicht um 10000 Zeichen, dann wird eingegeben:

```
>pack 10000
TPA:   FREE=53F0   USED=53F0
SYM:   FREE=0000   USED=0000
TXT:   FREE=2710   USED=04F9
```

Der Textspeicher ist jetzt größer geworden, auf Kosten der TPA. Die ist aber immer noch sehr groß, der Textspeicher kann bei Bedarf noch weiter vergrößert werden.

Eine Formalität am Rand: die Speichergrößen werden auf dem Bildschirm in sedezimaler Darstellung ausgegeben und trotzdem haben wir eben die gewünschte Erweiterung des Textspeichers mit dem Kommando PACK und einer dezimal dargestellten Zeichenmenge angegeben.Das geht also, Sie können aber auch bei PACK die sedezimale Schreibweise

wählen. In diesem Fall muß hinter der Zahl ein H stehen.

Beispiel: 10000 dezimal entspricht 2710 sedezimal (oder einfach Hex, wie der Praktiker sagt).Geben Sie es ein:

```
>pack 2710h
TPA:   FREE=53F0   USED=53F0
SYM:   FREE=0000   USED=0000
TXT:   FREE=2710   USED=04F9
```


Wenn Sie lustig sind, können Sie sogar die binäre Schreibweise benutzen, in diesem Fall muß ein b zur Kennzeichnung nachgestellt sein.

Mit dem PACK-Kommando kann der Textspeicher auch verkleinert werden, es wird einfach eine kleinere Zahl als die mit MEMORY angezeigte angegeben:

```
>pack 100
TPA:   FREE=7A9C   USED=7A9C
SYM:   FREE=0000   USED=0000
TXT:   FREE=0064   USED=04F9
```

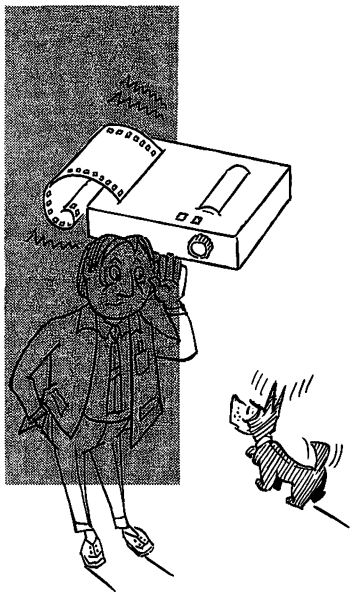
Sie können den Textspeicher auch auf die gerade benötigte Größe reduzieren, indem Sie einfach PACK 0 oder nur PACK eingeben:

```
>pack
TPA:   FREE=7B00   USED=7B00
SYM:   FREE=0000   USED=0000
TXT:   FREE=0000   USED=04F9
```

Kommando-Manager

Grob unterteilt besteht das Betriebssystem aus zwei Teilen, nämlich dem Ein/Ausgabe-System und den eigentlichen Systemprogrammen. Letztere wiederum kann man noch einmal unterteilen in den Zweig, der Kommandos von der Tastatur entgegen nimmt und sie ausführt sowie die Kommandos selber. Den ersten Zweig nennt man Kommando-Manager, im Englischen spricht man vom CCP (consol command processor). Sie kennen bereits drei Kommandos des Kommando-Managers:

EDIT, MEMORY, PACK.



Mit diesen drei Kommandos könnten Sie Ihren Computer schon fast als Textverarbeitungs-System verwenden, aber nur fast, denn es fehlt noch etwas: der fertige Text soll ja auch ausgedruckt werden. Das kann mit einem weiteren Kommando geschehen, geben Sie TYPE ein. Wenn ein Drucker am System angeschlossen ist, rührt er sich nicht. Der Text ist auf dem Bildschirm ausgegeben worden, weiter nichts. Geben Sie TYPE/L ein, danach selbstverständlich, wie nach jedem Kommando,RETURN (Taste CR bzw.Ret). Wenn der Drucker richtig angeschlossen ist, rattert er jetzt los. Der im Textspeicher stehende Text wird ausgedruckt. Die Ausgabe kann mit einer beliebigen Taste jederzeit angehalten werden.

Zusammenfassung

Das Betriebsystem kümmert sich unter anderen um die Speicherverwaltung, mit der Fehlermeldung *FULL* wird verhindert, daß außerhalb des vorgesehenen Textspeichers möglicherweise Speicherinhalte überschrieben werden. Ein Teil des Betriebssystems ist der Kommando-Manager. Er interpretiert eingegebene Kommandos und führt sie aus. Von diesen Kommandos haben Sie bisher MEMORY zur Speicherübersicht, EDIT zum Aufruf des Texteditors, PACK zur Veränderung des Textspeichers und TYPE zur Ausgabe des Textes kennengelernt.

Fragen:

1. Sie geben PACK 100000 und bei einem anderen Versuch PACK 60000 ein. Im ersten Fall erhalten Sie die Fehlermeldung "?", im zweiten Fall "O". Worauf führen Sie das zurück?

2. Wie kann man den Inhalt des Textspeichers auf dem Bildschirm ausgeben, ohne über das Kommando EDIT in den Texteditor zu gehen?

3. Wie könnte man ohne Taschenrechner, Papier, Bleistift und andere Hilfsmittel den Computer "mißbrauchen", um eine dezimal dargestellte Zahl in ihre sedezimale Darstellung umzuwandeln?

(Die Antworten zu diesen Fragen finden Sie auf Seite G2.)

Systemfunktionen

Beim ersten einfachen Programmbeispiel sollen einige Zeichen auf dem Bildschirm ausgegeben werden. Das ist scheinbar einfach, denn das Betriebssystem hat einen Speicherbereich als Bildschirmspeicher, dessen Inhalt ständig auf dem Bildschirm angezeigt wird. Wenn man mit LD-Befehlen entsprechende ASCII-Codes in den Bildschirmspeicher schreibt, dann müßten die zugehörigen Buchstaben auf dem Bildschirm stehen. Das ist wohl der Fall, aber man weiß nicht, welche anderen Zeichen überschrieben wurden. Auch die Position des Cursors ist nicht bekannt. Sie bestimmt ja, wo das nächste Zeichen geschrieben wird.

Wenn man dieseUmstände mit berücksichtigt, wird das scheinbar einfache Programm recht kompliziert, mit einigen LD-Befehlen ist es jedenfalls nicht getan. Zum Glück ist es aber nicht notwendig, sich um diese Dinge zu kümmern, denn das Betriebssystem enthält interne Programme, die dafür da sind, Zeichen auszugeben und den Cursor richtig zu positionieren. Auf diese internen Funktionen des Betriebssystems wird zugegriffen, und dann wird die Sache mit der Ausgabe von Zeichen auf dem Bildschirm tatsächlich einfach.

Frage:

1. Sehen Sie sich noch einmal die Zeichnung über die Speicheraufteilung (Bild A2.1) an.Wo sehen Sie eine Schwierigkeit, wenn Sie Programme innerhalb des Betriebssystems aufrufen wollen?

(Die Antwort zu dieser Frage finden Sie auf Seite G2.)

Die Schwierigkeit spielt keine Rolle, man muß gar nicht wissen, wo die Programmteile stehen, das Betriebssystem kann sie sich selber suchen. Es fehlt lediglich ein Einsprung ins Betriebssystem und eine Vereinbarung darüber, wie dem Betriebssystem mitgeteilt wird, welcher Programmteil, oder, wie man allgemein sagt, welche Systemfunktion durchzuführen ist.

Die Einsprungadresse für alle Systemfunktionen ist die Adresse 0005H. An diese Adresse schreibt sich das Betriebssystem selber einen Sprung an die richtige interne Stelle, wo die Systemfunktionen ausgeführt werden. Welche Systemfunktion ausgeführt werden soll, wird dem Betriebssystem durch eine Zahl im C-Register mitgeteilt, eventuelle Parameter stehen im E-Register oder im DE-Register-

paar. Auf diese Weise ist es möglich, Systemfunktionen aufzurufen zur Konsol-Ein-und Ausgabe, zur Druckerausgabe, zur Aus- und Eingabe von und an die seriellen Ports und zur Beendigung eines Benutzerprogramms -nach der Ausführung eines Programms soll sich ja wieder das Betriebssystem melden, es soll ein sogenannter Warmstart erfolgen.

Dieses Verfahren spart nicht nur Arbeit, es bringt noch weitere Vorteile. Es wird nämlich nicht irgendeine Konvention für die Aufrufe der Systemfunktionen verwendet, sondern die des Betriebssystems CP/M. Daher können Programme, die alle Ein- und Ausgaben über diese Systemfunktionen abwickeln, ohne Änderungen auf jedem CP/M-Computer laufen.

Für unser erstes Programm, das Zeichen auf dem Bildschirm ausgeben soll, werden zwei Systemfunktionen gebraucht: die Warmstart- und die Konsolausgabefunktion.

Erstere ist die Funktion 0, es wird 00H ins C-Register geschrieben und dann die Adresse 0005H aufgerufen:

Funktion 0 -Warmstart

Parameter beim Aufruf:	Register C: 00H
Parameter beim Rücksprung:	keine (kein Rücksprung)

Die Funktion zur Konsolausgabe ist die Funktion 2, es wird also 02H ins C-Register geladen, ferner das auszugebende Zeichen ins E-Register und dann die Adresse 0005H aufgerufen.

Funktion 2 -Konsolausgabe

Parameter beim Aufruf:	Register C: 02H
	Register E: ASCII-Zeichen
Parameter beim Rücksprung:	keine

Zusammenfassung

Bei unserem Betriebssystem lassen sich Systemfunktionen über eine Einsprungadresse aufrufen. Da dies nach den gleichen Konventionen wie bei CP/M geschieht, laufen Programme, die mit diesen Systemfunktionen arbeiten, auch auf anderen CP/M-Computern. Für das erste Versuchsprogramm werden die Systemfunktionen 0 (Warmstart) und 2 (Konsolausgabe) benötigt.

Systemfunktionen

Das erste Programm "Zeichenausgabe" hat gezeigt, daß die Ausgabe von Zeichen auf dem Bildschirm mit der Hilfe des Betriebssystems kein Problem ist. Wenn es aber darum geht, längere Texte auf dem Bildschirm auszugeben, dann wird die im Programm 1 gezeigte Methode doch recht umständlich.Es müßte ein Programm geschrieben werden, das einen längeren Text aus dem Speicher ausgeben kann, oder man muß eine geeignete Systemfunktion verwenden. In den nächsten Programmen werden Sie beides kennenlernen.

Eine solche Systemfunktion, die für die Ausgabe längerer Texte geeignet ist, gibt es: die Systemfunktion 9 für Stringausgabe (als Strings bezeichnet man hier Zeichenketten aus ASCII-Zeichen).Im C-Register wird, wie bereits gewohnt, der Funktionscode (also 9) übergeben. Ein ganzer String, der möglicherweise aus sehr vielen ASCII-Zeichen besteht, kann aber nicht in den Registern des Mikroprozessors übergeben werden.

Deshalb muß anders vorgegangen werden. Im Registerpaar DE wird dem Betriebssystem lediglich die Anfangsadresse des Strings mitgeteilt. Das Ende des Strings muß mit dem Zeichen "$" markiert sein.Das Betriebssystem gibt dann die Inhalte aller Speicherzellen ab der Adresse, die das DE-Registerpaar enthält, bis zum Zeichen "$" auf dem Bildschirm aus. Das Zeichen "$" selber wird nicht mit ausgegeben.

Funktion 9 -Stringausgabe

Parameter beim Aufruf: Register C: 09H
Registerpaar DE: Anfangsadresse des String. Ende des String mit mit "$" markiert

Parameter beim Rücksprung: keine

Wenn es um längere Texte geht, kann man auch den Inhalt des gesamten Textspeichers mit dem Kommando TYPE/L ausgeben lassen. Auch in diesem Fall sollte es nicht schwer sein, ein kleines Programm zu schreiben, das den gleichen Zweck erfüllt.

Die Funktion zur Stringausgabe schafft das nicht, denn das Ende des Textspeichers ist nicht mit dem Zeichen "$"

markiert, außerdem kann dieses Zeichen im Text vorkommen, ohne die Bedeutung "Stringende" zu haben.

Es muß jedes Zeichen des Textspeichers einzeln ausgegeben werden. Dazu muß man wissen, wo der Textspeicher anfängt und wo er aufhört. Das Ende ist wie das Ende aller ASCII-Dateien mit dem ASCII-Zeichen EOF (End of file=Ende der Datei) =1AH bezeichnet. Der Anfang des Textspeichers kann irgendwo im Speicher sein, das weiß am besten das Betriebssystem, das ja die Speicheraufteilung organisiert.

Damit das Betriebssystem die Frage beantworten kann, enthält es (auch noch für andere Fälle) spezielle Systemfunktionen. Diese erweiterten Systemfunktionen gibt es nur in unserem Betriebssystem, sie bewirken auf anderen CP/M-Computern nichts. Es hat z.B. keinen Sinn, auf einem Computer, der gar keinen speziellen Textspeicher hat, nach der Anfangsadresse des Textspeichers zu fragen.

Diese erweiterten Funktionen unseres Betriebssystems haben Funktionscodes ab 240. Die Funktion zur Abfrage des Textspeichers hat den Code 247 und gibt die Anfangsadresse im HL-Registerpaar zurück.

Funktion 247 -Hole Textspeicheranfang

Parameter beim Aufruf:	Register C: 0F7H
Parameter beim Rücksprung:	Registerpaar HL: Anfangsadresse des Textspeichers

Zusammenfassung

Für die Ausgabe von Strings auf dem Bildschirm gibt es die Systemfunktion 9 -Stringausgabe. Die Ausgabe des gesamten Textspeichers kann mit der erweiterten Systemfunktion 247 erreicht werden. Beide Systemfunktionen werden in gleicher Weise wie die anderen aufgerufen.

Frage:

1. Wodurch unterscheiden sich die normalen und die erweiterten Systemfunktionen?

(Die Antwort zu dieser Frage finden Sie auf Seite G2.)

Systemfunktionen

Als erste Systemfunktionen hatten Sie gleich am Anfang diejenigen zur Bildschirmausgabe (Konsolausgabe) kennengelernt. Sie werden sich denken können, daß im Betriebssystem auch Funktionen zur Eingabe vorhanden sind. Die einfachste trägt die Funktionsnummer 1, es ist die normale Konsoleingabe-Funktion.

Wird diese Funktion aufgerufen, dann wartet das Betriebssystem zunächst ab, bis ein Zeichen von der Tastatur kommt, bis eine Taste gedrückt wurde. Das dann von der Tastatur kommende Zeichen wird auf den Bildschirm geschrieben und dem aufrufenden Programm im A-Register übergeben.

Funktion 1 - Konsoleingabe

Parameter beim Aufruf: Register C: 01H

Parameter beim Rücksprung: Register A: ASCII-Zeichen

Auf den nachfolgenden Seiten des Fachgebiets PROGRAMMIERUNG werden Sie sich mit einem Programm zur Buchstabenstatistik befassen. In diesm Programm werden noch zwei weitere Systemfunktionen benötigt, die Sedezimalzahlen auf dem Bildschirm ausgeben.

Diese beiden Funktionen gehören zu den erweiterten Systemfunktionen, deren wichtigstes Kennzeichen im vorliegenden Fall die Möglichkeit ist, auf den Textspeicher zurückgreifen zu können.

Die Funktion 240 gibt den Inhalt des E-Registers als zweistellige Sedezimalzahl auf dem Bildschirm aus.

Funktion 240 - Byte ausgeben

Parameter beim Aufruf: Register C: 0F0H

Parameter beim Rücksprung: Register E: auszugebende zweistellige Sedezimalzahl

Die Funktion 241 gibt den Inhalt des DE-Registerpaars als vierstellige Sedezimalzahl auf dem Bildschirm aus.

Funktion 241 - Wort ausgeben

Parameter beim Aufruf:	Register C: 0F1H Registerpaar DE: auszugebende vierstellige Sedezimalzahl
Parameter beim Rücksprung:	keine

Zusammenfassung

Für die Eingabe eines Zeichens auf der Tastatur gibt es die Systemfunktion1 - Konsoleingabe. Für die Ausgabe von zweistelligen und vierstelligen Sedezimalzahlen gibt es die erweiterten Systemfunktionen 240 - Byte ausgeben und 241 - Wort ausgeben.

Frage:

1. Über welche Einsprungadresse ins Betriebssystem werden Systemfunktionen aufgerufen?

(Die Antwort zu dieser Frage finden Sie auf Seite G4.)

Systemfunktionen

Im ersten Teil des Lehrgangs haben Sie bereits die zwei Systemfunktionen kennengelernt, mit denen auf die Tastatur- und Bildschirmschnittstelle zugegriffen wird, ohne daß man sich dabei um die eigentliche Schnittstelle kümmern muß. Diese beiden System-Funktionen werden aber nicht Tastatureingabe- oder Bildschirmausgabe-Funktion genannt. Vielmehr spricht man von Konsoleingabe- und Konsolausgabe-Funktion. Dies ist verständlich, weil Tastatur und Bildschirm normalerweise die Bedienkonsole eines Computers darstellen.

Das war nicht immer so, früher wurden bei grossen Computern auch Fernschreiber als Konsole verwendet und auch heute noch gibt es diese Möglichkeit. Fest steht: die Konsol-Systemfunktionen bedienen in einem Computer immer das Gerät (die Geräte), das als Konsole angeschlossen ist. Man spricht dann von einem logischen Gerät, das die Funktion einer Konsole erfüllt, im Gegensatz zu dem tatsächlich angeschlossenen physikalischen Gerät.

Bei Ihrem Computer sind dem logischen Gerät Konsole die beiden physikalischen Geräte Tastatur und Bildschirm zugeordnet. Diese Zuordnung könnte jedoch im Betriebssystem oder durch spezielle Software geändert werden.

Die Systemfunktion 1 holt ein Zeichen von der Konsole (Tastatur) und gibt gleichzeitig dieses Zeichen auf der Konsole (Bildschirm) wieder aus, man sieht also die eingegebenen Zeichen. Falls noch keine Taste gedrückt wurde, wartet die Systemfunktion 1 solange, bis das geschehen ist. In manchen Fällen ist dies recht unpraktisch, denn in der Wartezeit könnte der Computer unter Umständen etwas anderes tun. Dann empfiehlt es sich, die Funktion 1 erst aufzurufen, wenn sie tatsächlich benötigt wird, wenn also ein Zeichen eingelesen werden kann. Eine weitere Systemfunktion sagt dem aufrufenden Programm, wann ein Zeichen bereit ist zum Einlesen:

Funktion 11 - Hole Konsolstatus

Parameter beim Aufruf: Register C: 0BH

Parameter beim Rücksprung: Register A: Status= 0FFH, wenn Zeichen bereit
000H, wenn kein Zeichen bereit

Frage:

1. Es wurde vorher gesagt, daß bei der Systemfunktion 1 der Computer sozusagen Kaffeepause hat, bis von der Tastatur ein Zeichen kommt. Während der Zeit könnte er etwas anderes tun. Probieren Sie das mit einem kleinen Programm einmal aus. Wir geben Ihnen dazu einige Anregungen:

   Während auf das Zeichen von der Tastatur gewartet wird, soll ununterbrochen ein Zeichen auf dem Bildschirm ausgegeben werden. Damit Sie diese "Nebenarbeit" deutlich von den eingegebenen Zeichen (auf welche die aufgerufene Systemfunktion 1 wartet) auf dem Bildschirm unterscheiden können, soll es das Zeichen "Punkt" sein. Der Bildschirm wird also voll mit Punkten geschrieben und jedesmal, wenn Sie irgendeine Taste drücken, steht das entsprechende Zeichen zwischen der Flut von Punkten.

   Wie schon gewohnt, müssen am Anfang des Programms die Systemfunktionen definiert werden, hier also Konsol-Ein- und Ausgabe sowie die Konsolstatus-Abfrage. Letztere wird nach dem Label Start geladen und nach der Abfrage entweder mit der Konsolausgabe-Funktion ein Punkt auf den Bildschirm gebracht oder mit der Konsoleingabe-Funktion das Zeichen von der betätigten Tastatur abgeholt. Ein Hinweis: die Auswertung des Konsolstatus geschieht mit einem Sprungbefehl, der das Zerobit im Flagregister abfragt. Durch einen OR A- oder AND A-Befehl muß es entsprechend gesetzt werden.

   Schließlich soll das Programm nicht mit dem Reset-Schalter angehalten werden, sondern mit einem bestimmten Zeichen, vielleicht mit CTRL-C oder ähnlich. Viel Spaß bei dieser kleinen Programmieraufgabe, die Lösung auf Seite G6 zeigt Ihnen, wie wir uns das vorgestellt haben.

Die mit der Systemfunktion 2 ausgegebenen oder mit der Systemfunktion 1 eingelesenen Zeichen werden sofort automatisch auf dem Bildschirm wieder ausgegeben. Im Tester können sie sogar nach dem Betätigen von CTRL-P auf einem angeschlossenen Drucker mitgeschrieben werden. Beides ist in vielen Fällen garnicht praktisch. Stellen Sie sich vor, daß z.B. im Editor alle eingegebenen Kontrollzeichen zur Steuerung des Cursors auf dem Bildschirm mit erscheinen - das Arbeiten mit dem Texteditor wäre praktisch unmöglich.

Es wird also für viele Gelegenheiten eine Konsol-Ein- und Ausgabe-Funktion benötigt, die nur direkt zur Konsole

geht und bei der es kein Echo (auf dem Bildschirm) der eingegebenen Zeichen gibt. Diese Systemfunktion hat die Nummer 6, bei ihr ist natürlich auch kein direktes Mitdrucken wie bei den Funktionen 1 und 2 möglich.

Funktion 6 - Direkte Konsoleingabe

Parameter beim Aufruf: Register C: 06H
Register E: 0FFH

Parameter beim Rücksprung: Register A: ASCII-Zeichen, oder 0, wenn kein Zeichen bereit

Funktion 6 - Direkte Konsolausgabe

Parameter beim Aufruf: Register C : 06H
Register E : ASCII-Zeichen (nicht 0FFH!)

Parameter beim Rücksprung: keine

Mit der Systemfunktion 6 sind demnach Eingabe, Ausgabe und Statusabfrage der Konsole möglich, ohne dabei irgendwelche zusätzlichen Funktionen auszuführen. Mit den Systemfunktionen 1,2 und 11 sind Eingabe, Ausgabe und Statusabfrage der Konsole möglich, wobei zusätzliche Funktionen (Bildschirmanzeige, Ausdruck) erfolgen können.

Die Druckerschnittstelle wird in gleicher Weise wie die Konsolschnittstelle mit einer Systemfunktion angesprochen, wobei das zu druckende Zeichen ebenfalls im E-Register übergeben wird:

Funktion 5 - Druckerausgabe

Parameter beim Aufruf: Register C: 05H
Register E: ASCII-Zeichen, das gedruckt werden soll

Parameter beim Rücksprung: keine

Leider ist im Betriebssystem CP/M keine Systemfunktion für die Abfrage des Druckerstatus vorgesehen. Aus Gründen der Kompatibilität gibt es diese Funktion deshalb auch in Ihrem Betriebssystem nicht. Wir werden aber später noch einen Weg aufzeigen, den Druckerstatus über einen direkten Sprung in die I/O-Routinen abzufragen.

B

24

Escape-Sequenzen

Auf den vorangegangenen Seiten wurde dargelegt, daß über die Systemfunktionen an angeschlossene Geräte wie Konsole oder Drucker ASCII-Zeichen im E-Register übergeben werden können. Das sind in der Regel druckbare Zeichen, also Buchstaben, Ziffern und Satzzeichen , aber auch Steuerzeichen: 0DH (CR =Carriage Return) für neue Zeile oder 0AH (LF= Line Feed) für Zeilenvorschub.

Manchmal werden noch weitere Steuerzeichen benötigt. Bei einem Drucker muß beispielsweise das Farbband oder der Schreibschritt umgeschaltet werden; bei einem Sichtgerät soll ein Teil der Bildschirmdarstellung gelöscht oder der Cursor an eine bestimmte Stelle gesetzt werden.

Diese zusätzlichen Steuerzeichen werden fast immer durch sogenannte Escape-Sequenzen realisiert. Damit sind Zeichenfolgen gemeint, die mit dem ASCII-Zeichen ESC (Escape) = 1BH beginnen. Die nach ESC folgenden Zeichen können von Gerät zu Gerät recht unterschiedlich sein, jeder Hersteller macht das nach den Gegebenheiten seines Geräts - leider. Bei den Sichtgeräten gibt es allerdings einige Sequenzen, die allgemein üblicher Standard sind und deshalb auch in Ihrem Computer vorgesehen wurden; diese sind nachstehend angegeben. Die Escape-Sequenzen für Drucker dagegen sind so unterschiedlich, daß es keinen Sinn hat, sie hier anzugeben. Sie müssen bei Bedarf den jeweiligen Handbüchern der Hersteller entnommen werden. Bei Bedarf, daß heißt, wenn Sie später einmal soweit in die Assembler- Programmierung eingestiegen sind, daß Sie darangehen können, z.B. Textverarbeitungsprogramme an Ihren Computer anzupassen

Escape-Sequenzen für das Sichtgerät

ESC T	Löschen aller Zeichen von der Cursorposition bis zum Ende der Zeile; der Cursor bleibt an seiner Stelle stehen.

ESC Y	Löschen aller Zeichen von der Cursorposition bis zum Ende des Bildschirms; der Cursor bleibt an seiner Stelle stehen.
ESC R	Löschen der Zeile, in der der Cursor steht; alle Zeilen darunter werden hochgeschoben, die unterste Zeile des Bildschirms wird leer.
ESC E	Einfügen einer Zeile an der Cursorposition; alle Zeilen darunter werden nach unten geschoben, die unterste Zeile des Bildschirms geht verloren.
ESC= m+20H n+20H	Cursor wird auf Zeile m und Spalte n gesetzt. Linke obere Ecke des Bildschirms hat die Koordinaten (0,0). Um den Cursor nach links oben zu setzten, ist also einzugeben: 1BH, 3DH, 20H, 20H ESC = Blank Blank

Die letzte Escape-Sequenz erscheint in dieser summarischen Aufstellung etwas undurchsichtig, Sie werden gleich anschliessend mit einigen Beispielen hinter die Positionierung des Cursors kommen. Dazu wird ein kurzes Programm gebraucht, das Sie selber entwerfen sollen.

Frage:

1. Wie sieht ein Programm aus, das Tastatureingaben mit der Systemfunktion 6 einliest und anschließend auf dem Bildschirm gleich wieder ausgibt? Bei der Eingabe von CTRL-C soll ein Warmstart erfolgen. Mit diesem Programm lassen sich die Escape-Seqenzen des Sichtgeräts ausprobieren.

Vergleichen Sie Ihren Ansatz mit unserer Lösung auf Seite G6. Geben Sie danach einige Zeilen irgendwelchen Text ein, mit deren Hilfe Sie dann die Auswirkungen der Escape-Seqenzen ESC E, ESC R, ESC Y und ESC T auf der Bildschirm-Darstellung verfolgen können.

Bei der Escape-Sequenz zur Positionierung des Cursors gibt es zuvor einige Überlegungen. In der obenstehenden Aufstellung ist als Beispiel die Position in der linken

oberen Ecke angegeben. Zuerst werden die den Koordinaten zugeordneten Werte in die sedezimale Darstellung umgesetzt:

m+20H entspricht 0+20H =20H
n+20H entspricht 0+20H =20H
Für ESC steht (im ASCII-Code) 1BH, für das Zeichen = steht 3DH.

Die Eingabe sieht dann so aus: 1BH, 3DH, 20H, 20H

In der ASCII-Tabelle müssen Sie nun nachschauen, welche Zeichen diesen Sedezimalwerten im ASCII-Code zugeordnet sind und dann diese Zeichen über ihre Tastatur eingeben. Bei den beiden ersten ist das klar, wir haben sie ja gerade zuvor aus Gründen der einheitlichen Darstellung umgewandelt, es sind die Tasten "ESC" und "=". Dem Wert 20H entspricht die Leertaste. Geben Sie also ein :

ESC = (zweimal die Leertaste) ,

der Cursor springt in Position 0/0, an den Anfang des Bildschirms. Bei der Berechnung anderer Positionen gilt, daß von der Dezimaldarstellung in die sedezimale Darstellung umgerechnet werden muß. Beispiel: die Koordinate der 17. Zeile ist:
m= 17D+20H= 11H+20H=31H
31H in ASCII entspricht der Taste "1".

Wobei ferner zu beachten ist, daß der Cursor im angeführten Beispiel auf der 18. Zeile steht, weil bei der Koordinatenrechnung die nullte Zeile mitgerechnet wird.

Zusammenfassung

Die Systemfunktionen 1 und 2 zur Eingabe und Ausgabe von Zeichen über die Konsole (Konsole = Tastatur und Bildschirm) werden ergänzt durch die Systemfunktion 11, die den Konsolstatus abfragt. Mit der Systemfunktion 6 sind direkte Ein- und Ausgabe ohne Echo und ohne zusätzliche Funktionen (z.B.Druckerausgabe) möglich. Mit der Systemfunktion 5 wird die Drucker-Schnittstelle bedient.

Für die Steuerung von angeschlossenen Geräten (z.B. Drucker) werden als Steuerzeichen Escape-Sequenzen verwendet. Ihre Zuordnung ist nur bei Sichtgeräten einigermassen einheitlich. Die Kenntnis dieser Steuerzeichen und ihrer Zuordnungen ist wichtig bei der Anpassung angeschlossener Geräte oder auch bei der Anpassung von Anwender-Softwarepaketen an das Betriebssystem.

Stringeingabe

Beim Life-Game-Programm war die Eingabe des Spielfelds ganz einfach: es wurde dafür der Editor benutzt und das fertige Spielfeld aus dem Textspeicher geholt. Es wurde bereits gesagt, daß in diesem Heft eine mathematische Aufgabe als Programmierbeispiel dient (es geht um die Berechnung von Fakultäten). Für diesen Zweck wird kein so kompliziertes Eingabeprogramm benötigt. Es müssen ja keine Spielfelder, sondern nur Zahlen eigegeben werden. Dafür genügt ein Programm, das eine Textzeile einliest. Für den Fall, daß es dabei Tippfehler gibt, sollten Korrekturen erlaubt sein. Anschließend kann dann die als Textzeile (ASCII-Zeichenkette, String) vorliegende Zahl in das interne Datenformat für die Rechnung umgewandelt werden.

Die Eingabe einer Textzeile ist eine sehr häufig vorkommende Aufgabe. Das Betriebssystem braucht selbst ein geeignetes Programm, um Kommandozeilen einzulesen und anschließend zu interpretieren. Es liegt auf der Hand, daß es auch für diese Aufgabe eine Standard-Systemfunktion gibt. Bei ihr ist allerdings die Kommunikation mit dem Betriebssystem etwas schwieriger als bei den Ihnen bisher bekannt gewordenen Systemfunktionen.

Beim Aufruf dieser Systemfunktion muß dem Betriebssystem mitgeteilt werden, an welcher Stelle im Speicher die eingegebenen Zeichen abgelegt werden sollen, denn im HL-Registerpaar könnte es ja nur zwei Zeichen zurückgeben. Außerdem muß das Betriebssystem noch wissen, wie viele Zeichen im Höchstfall von der Tastatur eingelesen werden sollen. Es könnte ja sonst passieren, daß mehr Zeichen eingelesen werden, als Speicherplatz für die Eingabe reserviert ist. Beim Rücksprung muß das Betriebssystem dem aufrufenden Programm lediglich mitteilen, wieviele Zeichen tatsächlich eingegeben wurden.

Im DE-Registerpaar, das bei Systemfunktionen für Mitteilungen ans Betriebssystem reserviert ist, können nicht gleichzeitig die Adresse für die einzugebenden Zeichen und deren maximale Anzahl übergeben werden. Deshalb wird für die Kommunikation zwischen Betriebssystem und aufrufendem Programm ein Datenbereich im Speicher eingerichtet. Dieser Bereich enthält die maximale Anzahl der einzugebenden Zeichen, die tatsächliche Anzahl der eingegebenen Zeichen und die eingebenen Zeichen selbst - alles in einer festgelegten Reihenfolge. Beim Aufruf der Systemfunktion wird dem Betriebssystem im DE-Registerpaar lediglich die Adresse dieses "Kommunikations-Datenbereichs" mitgeteilt. Die Systemfunktion zur Stringeingabe enthält die folgenden Parameter:

Funktion 10 - Stringeingabe

Parameter beim Aufruf: Register C: 0AH
Registerpaar DE: Adresse Datenbereich

Parameter beim Rücksprung: Zeichen c1 bis cn im Datenbereich

Der Inhalt des Datenbereichs ist folgendermaßen angeordnet:

Im DE-Registerpaar angegebene Adresse	+0	+1	+2	+3	+4 . . . .+n
	mx	nc	c1	c2	c3 . . . .cn

mx = maximale Anzahl der einzugebenden Zeichen
nc = tatsächliche Anzahl der eingegebenen Zeichen
c1 bis cn = eingegebene ASCII-Zeichen

Die Systemfunktion 10 liest Zeichen von der Tastatur ein und legt sie in dem Datenbereich ab, der zwei Adressen (DE+2) oberhalb der Adresse beginnt, auf die das DE-Registerpaar beim Aufruf zeigt. In diese Adresse (DE+0) wird die gewünschte Größe des Datenbereichs eingetragen und in die nächste (DE+1) vermerkt das Betriebssystem die Anzahl der tatsächlich eingegebenen Zeichen.

Das klingt recht kompliziert, am besten probieren Sie es einmal praktisch aus. Dazu ist ein kurzes Programm notwendig, das Sie sich selbst schreiben sollten und dann mit dem nachstehenden Listing vergleichen können.

```
        TITLE   STRING

        ORG     100H

SYSTEM  EQU     005H

START:  CALL    LESE            ;String einlesen
        JP      0               ;Programm-Ende

LESE:   LD      C,0AH           ;Stringeingabe
        LD      DE,080H         ;Anfang Datenbereich
        LD      A,10            ;Länge Datenbereich
        LD      (DE),A          ;-> (DE)+0
        CALL    SYSTEM          ;Funktionsaufruf
        RET                     ;Ende
```

Nach der Assemblierung steht auf dem Bildschirm:

```
>ASM

Z80-Assembler
pass 1:
+
pass 2:
+

no fatal error(s)


SUM=0435
CRC=818D
```

Gehen Sie danach in den Tester und füllen Sie den Speicherbereich von 080H bis 0FFH mit Nullen, um später beim Speicherausdruck eine bessere Übersicht zu haben, was geschehen ist. Die dazu gebrauchte Anweisung heißt F (Fill) und benötigt als Parameter den Anfang des Speicherbereichs, das Ende des Speicherbereichs und die Zeichen, die in die Speicherstellen "gefüllt" werden sollen (hier also 00).

```
>TEST
*F 0080,00FF,00
```

Danach mit G Programmstart und Eingabe der Zeichen, hier als Beispiel 123456:

```
*G 100
123456
```

Das Programm ist gelaufen, schauen Sie nach, ob die eingegebenen Zeichen an der gewünschten Stelle im Speicher stehen und was sonst noch dort steht:

```
>TEST
*D 0080
0080  0A 06 31 32 33 34 35 36 00 00 00 00 00 00 00 00 ..123456........
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00A0
```

. . . usw., der Rest der Darstellung interessiert hier nicht.

In der Speicherstelle 0080 steht 0A, das ist die sedezimale Darstellung von Zehn, welche als Wert für mc angegeben war. In der nächsten Speicherstelle steht die Anzahl der tatsächlich eingegebenen Zeichen, es waren sechs. Darauf folgen die 6 Zeichen. Probieren Sie aus, was geschieht, wenn andere Werte benutzt werden. Geben

Sie soviele Zeichen ein, wie mit mc vorgegeben sind, dann muß für nc der Wert 0 eingetragen sein. Und vor allem, probieren Sie das aus, was diese Systemfunktion interessant macht: die Möglichkeit des Editierens, also eingegebene Zeichen mit DEL bzw. BS zu löschen oder mit CTRL-X die ganze Zeile zu löschen und neu einzugeben. Mit CTRL-P kann übrigens auch die Druckerprotokollierung zugeschaltet werden. Daß die Eingabe mit RET bzw. CR beendet wird, haben Sie bereits bei der Eingabe des Programms gemerkt.

Wird am Anfang der Eingabezeile CTRL-C eingegeben, dann erfolgt kein Rücksprung ins rufende Programm, sondern ein Warmstart. Damit besteht eine einfache Möglichkeit, Programme zu verlassen, ohne einen Warmstart im Programm vorzusehen.

Viele Programme verwenden übrigens den Bereich von 0080H bis 00FFH als Datenpuffer. Er ist auch als allgemeiner Pufferbereich (auch für Diskettenoperationen) vorgesehen. Wenn Sie einen Datenbereich innerhalb Ihres Programms definieren, kann dessen maximale Länge auch gleich mit dem Pseudobefehl DB mit assembliert werden.

Zusammenfassung

Mit der Systemfunktion 10 - Stringeingabe kann eine Textzeile eingegeben werden mit der Möglichkeit, Korrekturen anzubringen. Die eingegebenen Zeichen werden in einem Datenbereich abgelegt, dessen Anfangsadresse im DE-Registerpaar steht. Am Anfang dieses Datenbereichs wird seine maximale Größe angegeben, darauf folgt die tatsächliche Zeichenzahl und dann die eigentlichen Zeichen.

Fragen:

1. Was geschieht, wenn Sie mehr Zeichen eingeben, als mit dem Wert von mc festgelegt wurde?

2. Wie lang kann der Datenbereich, der mit der Systemfunktion 10 angelegt wird, maximal sein ?

(Die Antworten zu diesen Fragen finden Sie auf Seite G 10)

BIOS-Einsprungtabelle

In allen Programmen dieses Lehrgangs war zu sehen, wie mit Hilfe der Systemfunktionen des Betriebssystems Ein- und Ausgabeoperationen programmiert werden können. Man übergibt dem Betriebssystem ein ASCII-Zeichen, befiehlt ihm mit einem Funktionscode, dieses Zeichen auf den Bildschirm zu schreiben oder zu drucken, und braucht sich um nichts weiter zu kümmern - die Sache funktioniert. Und das sogar auf einem anderen Computer, wenn das gleiche Betriebssystem installiert ist.

Trotzdem wollen wir an dieser Stelle einmal ergründen, was das Betriebssystem mit den übergebenen Zeichen im einzelnen anfängt. Das geschieht nicht nur aus reiner Neugierde: Sie werden auf den folgenden Seiten bald merken, was damit bezweckt ist.

Es gibt zur Zeichenausgabe und zur Zeicheneingabe jeweils mehrere Systemfunktionen. Es liegt die Vermutung nahe, daß all diese Systemfunktionen ihre Ein- und Ausgaben letztlich doch über ein einziges Unterprogramm zur Zeicheneingabe und eines zur Zeichenausgabe abwickeln. Solche Unterprogramme, von denen die Ein- und Ausgabefunktionen letztlich abgewickelt werden, gibt es tatsächlich.

Man nennt diesen Teil des Betriebssystems BIOS (Basic-Input-Output-System). Die BIOS-Unterprogramme existieren nicht nur, sie sind für ganz spezielle Fälle sogar auch für den Anwender zugänglich! Außerdem rufen nicht nur die Systemfunktionen diese Unterprogramme auf, sondern auch die anderen Systemprogramme, wie Editor, Tester oder Assembler.

Die BIOS-Unterprogramme sind alle über eine Adressentabelle zugänglich, die man Einsprungtabelle nennt. Das ist eine Auflistung von Sprungbefehlen, die zu den eigentlichen BIOS-Unterprogrammen führen. Diese Einsprungtabelle beginnt immer an einer Adresse, die mit 00H endet, also z.B. 9A00H oder 0AB00H.

Die auf der nächsten Seite stehende Einsprungtabelle enthält anstelle der zunächst nicht bekannten ersten beiden Stellen der Adressen die Zeichen xx.

Zu Ihrer Information haben wir die komplette Einsprungtabelle eines CP/M-Systems angegeben. Selbstverständlich stehen sämtliche Einsprünge für die Diskettenverwaltung in einem residenten Betriebssystem ohne Disketten nicht zur Verfügung.

Adresse	Programm
xx00H	Kaltstartprogramm
xx03H	Warmstartprogramm
xx06H	Konsolstatus - 0FFH in A, wenn Zeichen bereit, sonst 00H in A
xx09H	Konsoleingabe - Zeichen von Konsole in A
xx0CH	Konsolausgabe - Zeichen in C an Konsole
xx0FH	Druckausgabe - Zeichen in C an Drucker
xx12H	Stanzerausgabe - Zeichen in C an Stanzer (oder serielle Schnittstelle)
xx15H	Streifenlesereingabe - Zeichen von Leser (oder serieller Schnittstelle) in A
xx18H	Home - Kopf auf Spur 0 (nur für Diskettensysteme)
xx1BH	Laufwerk - Laufwerk (C) ansprechen (nur für Diskettensysteme)
xx1EH	Spur - Spur (BC) wählen (nur für Diskettensysteme)
xx21H	Sektor - Sektor (BC) wählen (nur für Diskettensysteme)
xx24H	DMA - Transferadresse einstellen (nur für Diskettensysteme)
xx27H	Read - Sektor lesen (nur für Diskettensyst.)
xx2AH	Write - Sektor schreiben (nur für Diskettensysteme)
xx2DH	Druckerstatus - 0FFH, wenn Drucker bereit,, sonst 00H in A (nicht immer implementiert!)
xx30H	Sektorübersetzung (nur für Diskettensysteme)

Es wurde zuvor gesagt, daß die BIOS-Unterprogramme auch für den Benutzer zugänglich sind. Sie werden fragen, wie das laufen soll, sind doch die kompletten Adressen der an sich recht praktischen Einsprungtabelle nicht bekannt - was heißt "xx"? Oder doch? - denken Sie mal nach, eine Adresse kennen Sie: es ist der zweite Einsprung von oben, der Warmstart. Sie wissen, daß ein Sprung zur Adresse 0 in Ihrem Systen einen Warmstart auslöst.

Rufen Sie den Tester auf und sehen Sie mit dem L-Kommando nach, was an der Adresse 0 tatsächlich passiert (L 0). Sie werden einen Sprungbefehl vorfinden und es wird Ihnen auffallen, daß die Zieladresse des Sprungs mit 03H endet. Ein Blick in die BIOS-Einsprungtabelle zeigt: der Warmstartsprung hat dort die Adresse xx03H. Das deutet darauf hin, daß der an der Adresse 0 stehende Sprung in die BIOS-Einsprungtabelle geht.

Auch das läßt sich nachprüfen: die Zieladresse des Sprungs bei 0 wird ebenfalls mit einem L-Kommando dissassembliert und es kommt eine Liste von Sprungbefehlen zum Vorschein: da ist die BIOS-Einsprungtabelle! Sie läßt

sich also in jedem System finden, indem man die an den Systemadressen 0001H und 0002H stehende Zieladresse des Warmstartsprungs liest.

Wozu soll das Ganze aber nun gut sein, was kann man mit den BIOS-Unterprogrammen anfangen, wenn sich doch alle Ein- und Ausgaben bequem über die Systemfunktionen abwickeln lassen? Man kann z.B. hingehen und (mit viel Mut und Vorsicht!) einen der Tabelleneinträge (nehmen wir einmal den für die Konsoleingabe) dahingehend ändern, daß nicht mehr in das BIOS-Unterprogramm, sondern dafür in ein selbst geschriebenes Programm gesprungen wird. In diesem Fall erhält der Editor z.B. seine Zeichen nicht mehr von der Tastatur, sondern von dem selbst geschriebenen Programm angeliefert. Das Ganze klingt schon etwas riskant, eine etwas haarige Sache - im Anschluß lernen Sie ein Beispiel für eine solche "haarige" Sache kennen.

Beim Spielen mit dem Primzahlen-Programm stellt sich heraus, daß bei den höheren Primzahlen die Suche doch einige Zeit braucht. Vielleicht wollen Sie auch einfach mal sehen, wie sich der Computer mit einem solchen "Dauerprogramm" abfindet (er sucht und sucht und sucht, länger, als Sie Geduld zum Zugucken haben).

Während dieser Zeit sitzen Sie vor dem Bildschirm und würden das Programm am liebsten abbrechen, weil Sie ja vielleicht noch einen Brief mit dem Editor schreiben wollen. Da wäre es ideal, wenn das Primzahlenprogramm im Hintergrund weiterarbeiten könnte, während im Editor geschrieben wird. Sie wissen ja: kein Mensch kann so schnell tippen, daß der Computer zu 100 Prozent ausgelastet ist. Genau dieser Idealfall soll verwirklicht werden: das Primzahlenprogramm soll immer dann rechnen, wenn der Editor auf eine Eingabe wartet.

Multitasking, Multi-User

Die Aufgabe eines Multi-Tasking-Betriebssystems ist es, den gleichzeitigen Ablauf mehrerer Programme zu organisieren (mit dem englischen Wort "Task" wird hier ein Programm, eine Aufgabe für den Computer bezeichnet). Meist sind diese Multi-Tasking-Betriebssysteme gleichzeitig auch Multi-User-Betriebssysteme. Es können dann nicht nur mehrere Programme gleichzeitig laufen, sondern auch mehrere Konsolen für verschiedene Benutzer (User) von einem einzigen Computer bedient werden.

Dabei kann jeder Benutzer seine Konsole verwenden, als ob er der einzige Benutzer wäre, dem der ganze Computer

allein zur Verfügung steht. In Wirklichkeit beschäftigt sich der Computer nur während eines Bruchteils seiner gesamten Rechenzeit mit der einzelnen Konsole. Er verteilt die verfügbare Rechenzeit auf alle Benutzer bzw. Programme, die Rechenzeit anfordern.

Die eine Anwendung derartiger Betriebssysteme sind größere Computer, an denen mehrere Konsolen betrieben werden sollen.

Eine andere, nicht weniger wichtige Anwendung sind Prozeßrechensysteme, die in Echtzeit (Real-Time) irgendwelche technischen Abläufe (Prozesse) steuern. In solchen Ablaufsteuerungen müssen oft eine Vielzahl von Aufgaben gleichzeitig erledigt werden: Meßwerte zu bestimmten Zeitpunkten einlesen, mit diesen Werten Berechnungen durchführen, Stellglieder verstellen und schließlich auf einem Drucker den gesamten Ablauf protokollieren.

Jede dieser Aufgaben ist eine eigene "Task" und hat zwei wichtige Eigenschaften: Priorität und Dauer. In der Regel haben kurze Tasks eine hohe Priorität und umgekehrt. So dauert vielleicht das Erstellen des Druckerprotokolls wegen damit verbundener Berechnungen ziemlich lange, ist aber auch nicht zeitkritisch, weil es nicht darauf ankommt, ob es einige Sekunden früher oder später fertig ist.

Das Einlesen eines Meßwertes geht dagegen sehr schnell, muß aber auch sofort erfolgen, wenn der Meßwert vorliegt. Das bedeutet: die Task "Meßwert" bekommt eine hohe Priorität und kann die Task "Protokoll" jederzeit unterbrechen. Während dieser Unterbrechungen müssen natürlich jedesmal die Inhalte des Programmzählers und der Register gespeichert werden, damit die unterbrochene Task anschließend wieder richtig fortgesetzt werden kann.

Eine Task hat aber neben den festen Eigenschaften Priorität und Dauer des weiteren einen Zustand, in dem sie sich gerade befindet. Sie kann z.B. ablaufen, oder darauf warten, ablaufen zu können, weil gerade noch eine andere Task mit höherer Priorität abläuft. Eine Task kann aber auch ruhen, nicht aktiviert sein, weil beispielsweise kein Meßwert vorliegt. Sie kann auch in einem Wartezustand sein, weil eine Tastatureingabe oder noch weitere Meßwerte für die Protokollerstellung benötigt werden.

Das Multi-Tasking-Betriebssystem hat nun die Aufgabe, zu bestimmen, wann welche Task ablaufen darf und den jeweiligen Rechenstand der einzelnen Tasks in Zwischenspeichern zu erfassen. Dabei muß das Betriebssystem die Prioritäten aller Tasks berücksichtigen und die notwendige Rechenzeit anfordern.

Es muß aber auch an andere Nebenbedingungen denken: beispielsweise darf eine Task mit hoher Priorität, die gerade eine noch nicht getätigte Tastatureingabe benötigt, selbstverständlich trotz der hohen Priorität nicht gestartet werden.

Die Priorität ist überhaupt die Bedingung, um die sich alles dreht. Bei der Programmierung von Prozeßrechensystemen muß man sich sehr genau überlegen, welche Task welche Priorität bekommen darf. Gibt man einer Task eine zu hohe Priorität, dann kommen womöglich andere wichtige Aufgaben nicht mehr zum Zuge und das ganze System funktioniert nicht.

Ein einfaches Beispiel soll noch demonstrieren, wie in einem solchen Prozeßrechensystem die Zeitaufteilung aussehen kann. Angenommen, in dem System gibt es 3 Tasks:

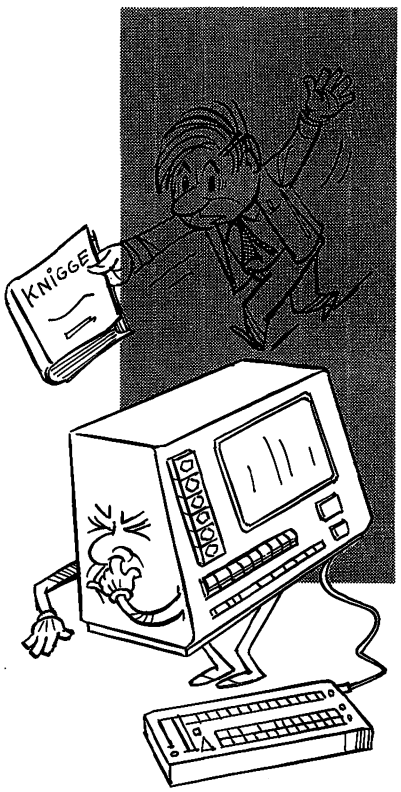
1. Meßwerte einlesen, 5 mal pro Sekunde, je 0,05 Sekunden
2. Meßwerte verarbeiten, 1 mal pro Sekunde, 0,5 Sekunden
3. Stellglied nachregeln, 2 mal pro Sekunde, 0,1 Sekunden

Task 1 erhält die höchste Priorität, da sie am wenigsten Zeit (0,05 Sekunden Dauer) beansprucht. Die zweithöchste Priorität erhält Task 3 und die lange dauernde Task 2 die niederste Priorität.

Die Aufteilung der Rechenzeit innerhalb des Zeitraums von einer Sekunde wird dann so aussehen:

```
Zehntel-Sekunden:   0 1 2 3 4 5 6 7 8 9

Task 1 :            *   *   *   *   *
Task 2 :              * *** *   *** **
Task 3 :             **       **      *
```



Task 1 kommt mit ihrer höchsten Priorität immer sofort an die Reihe. Task 3 darf nur dann arbeiten, wenn Task 1 fertig ist, und Task 2 schließlich kommt nur dann zum Zuge, wenn weder Task 1 noch Task 3 Rechenzeit anfordern. Es wird deutlich, daß der Computer ganz schön zwischen den drei Aufgaben hin- und herspringen muß. Nur in dem kleinen Zeitraum von 0,05 Sekunden innerhalb der ganzen Sekunde hat er nichts zu tun, denn er ist ja 5 * 0,05 + 2 * 0,1 + 0,5 = 0,95 Sekunden beschäftigt gewesen. Das bedeutet eine Auslastung von 95 Prozent, er könnte also kaum noch weitere Aufgaben erledigen.

Von Ihrem Computer können Sie das nicht sagen, der hat lange Zeitabschnitte, während denen er in der N... , naja, was kleine Buben manchmal tun, wenn sie Langeweile haben. Diese Feststellung bringt uns von dem kleinen Exkurs in die Prozeßrechentechnik wieder zu unseren Re-

alitäten zurück. Wir haben nämlich kein Multi-Task-Betriebssystem zur Verfügung, das die Aufteilung der Rechenzeit auf verschiedene Aufgaben verwalten kann.

Trotzdem soll unser Computer in den für ihn langen Pausen zwischen den Tastatureingaben noch etwas anderes tun, er soll gleichzeitig Primzahlen berechnen, während wir mit ihm anderweitig arbeiten wollen. Wir werden uns selbst bei diesem Multi-Tasking helfen, wobei es zuerst Probleme auf dem Bildschirm gibt. Es geht nämlich nicht, daß der Editor und das Primzahlenprogramm gleichzeitig Zeichen auf den Bildschirm schreiben.

Diese Probleme lassen sich auf einfache Weise umgehen: das Primzahlenprogramm gibt seine Ergebnisse nicht auf dem Bildschirm, sondern auf einem angeschlossenen Drucker aus. Als zweite Möglichkeit ist vorgesehen, daß die Ergebnisse des Primzahlen-Suchens laufend abgespeichert werden. Während der Arbeit mit dem Editor kann man dann immer mal nachschauen, wie weit das Primzahlenprogramm inzwischen gekommen ist.

Diese zweite Möglichkeit ist bei weitem nicht so eindrucksvoll wie das gleichzeitige Ausgeben auf einem Drukker. Aber, wenn kein solcher vorhanden ist, geht es eben nicht anders.

Zusammenfassung

Der Teil des Betriebssystems, der sich mit der Ein- und Ausgabe von Zeichen befaßt, wird BIOS genannt. Die verschiedenen BIOS-Unterprogramme sind über eine Einsprungtabelle auch für Benutzer zugängig. Diese Tabelle findet man über die Sprungadresse, die an der Adresse 0 für den Warmstart angegeben ist.

Mit Multi-Tasking und Multi-User können einerseits mehrere Aufgaben gleichzeitig von andererseits mehreren Benutzern gleichzeitig erledigt werden. Neben Großrechenanlagen wird Multi-Tasking vor allem in der Prozeßrechner-Technik eingesetzt.

Diskettenbetrieb

Bei der Beschäftigung mit Ihrem Computer werden Sie schon oft festgestellt haben, daß der Umgang mit den Cassetten (das Cassetten-Handling, wie es in Expertenkreisen heißt) doch recht umständlich und vor allem zeitaufwendig ist. Man sollte dabei auch immer wissen, welches Programm auf welcher Cassette gespeichert ist und dann auch noch wo, denn das Absuchen einer Seite einer C60-Cassette dauert immerhin 30 Minuten.

So ist es kein Wunder, wenn trotz der damit verbundenen Kosten früher oder später jeder, der sich ernsthaft mit der Computerei befaßt, zum Diskettenbetrieb übergeht. Auch Sie sind diesen Weg gegangen und geniessen jetzt die Vorteile gegenüber vorher: das Laden und Abspeichern von Dateien dauert nur noch Bruchteile von Sekunden und der Inhalt der Disketten ist ebenfalls kein Geheimnis: mit einem entsprechenden Kommando kann der blitzschnell auf dem Bildschirm angezeigt werden.

Von den meisten Benutzern werden die Disketten allerdings nur zum Speichern von Programmen oder Texten verwendet. Das geschieht mit vorhandenen Systemprogrammen, anders gesagt: ein Texteditor oder etwa ein BASIC-Interpreter verwendet die Disketten als schnellen Massenspeicher. Eigene Programme in Maschinensprache, die auf die Diskette zugreifen, Daten lesen, verändern und wieder auf die Diskette abspeichern, werden recht selten geschrieben. Der Horror vor den scheinbar undurchschaubaren Systemaufrufen des Disketten-Betriebssystems (DOS, Disk Operating System) ist zu groß.

Diese Hürde zu überwinden, das ist der eine Zweck des vorliegenden Heftes. Nachdem wir Ihnen in den vorangegangen 4 Heften die Angst vor der Assemblerprogrammierung hoffentlich genommen haben, werden Sie uns glauben, daß der Zugriff auf Disketten mit Programmen in Maschinensprache recht einfach ist und dabei eine Fülle von Möglichkeiten eröffnet.

Die andere Aufgabe dieses Heftes ist die Einführung in den Umgang mit dem Betriebssystem CP/M, soweit es für die Arbeit mit Ihrem Computer notwendig ist. CP/M heißt Control Program for Mikrocomputers und ist das Standard-Betriebssystem für die 8-Bit-Rechner der 8080- und Z80-Serie.

Was ein Betriebssystem tut, wissen Sie aus den vorangegangenen Heften schon. Zu den dort vorgestellten Auf-

gaben (Speicherverwaltung, Ein- und Ausgabefunktionen) kommen bei einem Disketten-Betriebssystem noch einige Funktionen, mit denen die Ein- und Ausgabe von Daten auf Disketten ermöglicht wird. Davon wird noch ausführlich die Rede sein.

Im Fachgebiet PROGRAMMIERUNG werden Sie ein einfaches Beispiel-Grundprogramm kennenlernen. Es wird in unterschiedlichen Versionen einige der Möglichkeiten aufzeigen, wie man in Assemblerprogrammen mit Disketten umgehen kann.

Sie haben sich vielleicht schon einmal gewundert, daß die Programmtexte in den Listings der vorangegangenen Hefte immer in Großbuchstaben geschrieben sind, die Kommentare jedoch auch in Kleinbuchstaben. Das macht die Programme lesbarer und wir hoffen, daß Sie unseren fleißigen Umgang mit der Shifttaste für die Großschreibung gehörig bewundert haben. Zu Unrecht, denn die Programmtexte wurden alle in Kleinbuchstaben eingetippt und dann von einem kleinen Programm bearbeitet, das die Kommentare und die in Anführungsstrichen stehenden Strings erkennt und unbehelligt läßt, den Programmtext hingegen in große Buchstaben umwandelt.

Genau dieses Programm ist das vorher erwähnte Beispiel, das Ihnen die verschiedenen Verfahren zeigen soll, mit denen Sie auf die Diskettenaufzeichnungen zugreifen können.

Beim Kennenlernen des CP/M-Betriebssystems wird Ihnen vieles bekannt vorkommen: die Speicher-Organisation und die Konventionen für die Systemaufrufe sind die gleichen wie die des ZEAT-Betriebsystems. Selbstverständlich war CP/M zuerst da und wir haben ZEAT so angelegt, daß es kompatibel zu CP/M ist. Wenn Sie also mit ZEAT ein Assemblerprogramm geschrieben haben, das seine Ein- und Ausgaben brav über die Systemfunktionen durchführt, dann läuft dieses Programm ohne Änderung auf einem beliebigen CP/M-Computer. Es gibt auch unter CP/M die Ihnen schon bekannten Systemfunktionen.

Was es bei CP/M allerdings nicht gibt, sind die erweiterten Systemfunktionen, die beispielsweise den Textspeicher betreffen. Bei CP/M gibt es nämlich keinen reservierten Speicherbereich für einen Textspeicher. Leider verfügt dieses Betriebssystem auch nur über einen sehr einfachen Texteditor (er heißt hier ED), gegen den der Editor des ZEAT fast schon ein richtiges Textverarbeitungssystem ist.

Außerdem stammt CP/M aus einer Zeit, zu der es den Z80-Prozessor noch garnicht gab. Deshalb wird bei CP/M auch

kein Z80-Assembler mitgeliefert, sondern eine einfacherer 8080-Assembler, der mit den Z80-Programmen nichts anfangen kann.

Dies alles sind Gründe, warum das Ihnen vorliegende CP/M-Betriebssystem in einigen Punkten hinter dem ZEAT-Betriebssystem zurücksteht. Das ist jedoch für Sie kein Nachteil: ZEAT und CP/M werden zusammen in den Speicher geladen und sind dann gleichzeitig aktiv, sie stellen sozusagen ihre gesammelte Leistung zur Verfügung. Wie das genau aussieht, wird in diesem Heft dargelegt.

Zusammenfassung

Die Diskettenversion von ZEAT läuft unter dem Betriebssystem CP/M, wobei dessen DOS benutzt wird. Diese Kombination vereint die Vorteile von ZEAT -komfortabler Editor, Assembler und Tester- mit denjenigen des altbewährten und weitverbreiteten CP/M.

Fragen:

1) Abgesehen von der erwähnten vorteilhaften Kombination CP/M-ZEAT: welchen wichtigen Vorteil bringt Ihnen die Tatsache, daß Ihr Computer mit dem Betriebssystem CP/M läuft, noch?

(Die Antwort zu dieser Frage finden Sie auf Seite G 19.)

Diskettenbetrieb unter CP/M

Was eine Diskette (Englisch Floppy Disk oder kurz Floppy) ist, brauchen wir Ihnen sicher nicht mehr im einzelnen zu erklären. Diese runden Scheiben aus weichem Kunststoff sind ähnlich wie Tonbänder auf einer oder beiden Seiten mit einem magnetisierbarem Material beschichtet. Während die Aufzeichnung bei Bändern in einer fortlaufenden Spur geschieht, werden die Daten auf den sich drehenden Disketten in einzelnen, ringförmigen und getrennten Spuren aufgezeichnet. Der Schreib/Lesekopf steht nicht immer an der gleichen Stelle, sondern er wird von innen nach außen und umgekehrt in Schritten bewegt (man hört das deutlich), um die einzelnen Spuren lesen oder beschreiben zu können.

Im Heft 4 auf Seite E 14 wird im Zusammenhang mit der Datenübertragung gesagt, daß es für die Datensicherheit vorteilhaft ist, Blöcke zu bilden und diese zur Kontrolle mit Prüfsummen zu versehen. Aus dem gleichem Grund werden bei der Aufzeichnung auf Diskette die Spuren nicht durchgehend beschrieben, sondern noch einmal in einzelne Sektoren unterteilt. Wie das im einzelnen gemacht wird, ist dabei nicht so interessant. Wichtig ist, daß das Betriebssystem mit der Unterteilung klarkommt und die gespeicherten Daten auch wieder findet.

Als Beispiel für eine solche Unterteilung kann das einzige standardisierte CP/M-Diskettenformat angeführt werden: auf den einseitigen 8-Zoll-Disketten in einfacher Dichte sind 77 Spuren zu je 26 Sektoren mit 128 Bytes untergebracht. Auf einer solchen Diskette haben demnach theoretisch 77*26*128 = 256256 Bytes Platz. In Wirklichkeit ist die benutzbare Kapazität etwas kleiner, weil das Betriebssystem und das Inhaltsverzeichnis auch Platz benötigen.

"Inhaltsverzeichnis" ist ein Stichwort für die folgenden Zeilen, bei denen es nun um den praktischen Umgang geht. Bei einem Disketten-Betriebssystem wie CP/M lassen sich nicht nur Daten wie auf einer Tonbandcassette abspeichern und wieder einlesen, wobei jeder Datenblock, der ein Programm oder der ASCII-Inhalt des Textspeichers sein kann, einen Namen bekommt. Vielmehr kann ein Datenblock unter Angabe seines Namens ohne Suchen wieder eingelesen werden, der Anwender braucht sich überhaupt nicht darum zu kümmern, wo der Datenblock auf der Diskette abgelegt ist.

Den Datenblock nennt man üblicherweise eine Datei (Englisch File), der zugehörige Name ist der Dateiname (Filename). Das Betriebssystem legt auf der Diskette ein Disketten-Inhaltsverzeichnis an, in dem der Name jeder Datei und die Information darüber, auf welchen Spuren und Sektoren sich die Datei befindet, eingetragen ist. Wird eine Datei nicht mehr benötigt, dann kann das Betriebssystem die Eintragung im Inhaltsverzeichnis einfach löschen und die freigewordenen Sektoren können von anderen Dateien belegt werden.

Ein Dateiname besteht bei CP/M aus drei Teilen: einem Haupt-Dateinamen, der aus bis zu 8 Buchstaben oder Ziffern (oder bestimmten Zeichen, wir kommen darauf noch zurück) bestehen kann, einem Punkt und einer Erweiterung des Dateinamens, die aus bis zu 3 Ziffern oder Buchstaben bestehen kann. Die Erweiterung wird normalerweise zur Kennzeichnung des Dateityps verwendet. Alle Programme werden als Typ "COM" bezeichnet; Ihre Briefe könnten Sie z.B. mit dem Typ "BRF" kennzeichnen. Gültige Dateinamen wären beispielsweise:

```
ZEAT.COM        -ZEAT-Programm
ADRESS.DAT      -Adressen-Datei
BRIEF1.TXT      -Brief
LIFEGAME.ASM    -Assembler-Listing eines Programms
```

Frage:

1) Sind die folgenden Namen gültige Dateinamen, und wenn nicht, warum?

   FRAGE?.BRF
   LEHRBRIEF.TXT
   PROGRAMM.2
   TESTER

(Die Antwort zu dieser Frage finden Sie auf Seite G 19.)

Im Prinzip können die Dateinamen also frei gewählt werden, allerdings mit einer schon erwähnten Einschränkung: alle Programm-Dateien (lauffähige Programme im Maschinencode) müssen die Erweiterung (den Typ) "COM" erhalten. Das CP/M-Betriebssystem und die Diskettenversion von ZEAT sind damit nämlich in der Lage, Programm-Dateien zu erkennen und diese Programme zu starten, wenn nur ihr Name eingegeben wird. Das Betriebssystem lädt das Programm ab Adresse 100H in die TPA und springt dann zur Adresse 100H. Wie das Anwender-Programm die Kontrolle an

das Betriebssystem zurückgeben kann (Warmstart), wissen Sie ja bereits vom residenten ZEAT-Betriebssystem. Nachdem jetzt feststeht, wie ein Dateiname auf Diskette aussehen muß, können Sie etwas erledigen, was Ihnen sicher schon lange auf den Nägeln brennt: die mühsam auf der Cassette gesammelten Programme auf die bequeme Diskette zu übertragen.

Dateienübertragung von Cassette nach Diskette

Lassen Sie noch einmal durch die Eingabe von "3" nach dem Anfangsmenü das residente ZEAT booten. Nach dem Aufruf des Testers geben Sie das Kommando RF ein. Das Betriebssystem fragt nach dem Dateinamen mit "Filename?" Sie geben den Namen, unter dem die Datei auf der Cassette steht, an, z.B. "1001". Danach wird die Datei in den Speicher geladen und die nächste freie Adresse angegeben (vgl. Seite 22 des ZEAT-Handbuchs). Auf dem Bildschirm könnte also folgendes stehen:

```
>TEST
*RF
FILENAME ? 1001
NEXT: 06C9
```

Diese nächste freie Adresse ist wichtig für das nachfolgende CP/M-Kommando SAVE. Aber verlassen Sie erst einmal das residente ZEAT mit RESET und gehen nach dem Anfangsmenü über "1" ins CP/M. Dort wird die in der TPA stehende Datei mit dem Kommando SAVE auf die Diskette abgespeichert:

```
A>SAVE 6 LIFEGAME.ASM
```

Beim SAVE-Kommando wird die Anzahl der auf die Diskette zu kopierenden Bytes in 256-Byte-Blöcken angegeben. Dies geschieht in dezimaler Darstellung nach einem Space hinter dem Kommando. Im Beispiel steht hier "6". Diese Zahl stammt aus dem HOB der nächsten freien Adresse, die vom Test-Kommando RF bei ZEAT angegeben wurde: 06C9. (Eine recht praktische Sache, oder?) Schließlich wird nach einem weiteren Space noch der Name angehängt, unter dem die Datei auf der Diskette stehen soll. Im Beispiel ist es ein Programm-Listing, das noch nicht assembliert ist, also noch in ASCII-Zeichen abgelegt ist.

So, jetzt ist das erste Programm von der Cassette auf die Diskette gerettet- sehen Sie gleich mal nach, ob es auch wirklich dort steht. Das Inhaltsverzeichnis hat selbst-

verständlich auch einen englischen Namen : Directory. Entsprechend heißt das Kommando zum Anzeigen der Directory auf dem Bildschirm

A>DIR (und danach RETURN bzw. CR)

Nachdem die Datei im Inhaltsverzeichnis erscheint, sollten Sie es doch genau wissen wollen, rufen Sie ZEAT auf:

A:0> PACK 20000

Danach mit EDIT in den Editor und dort ESC-A:

*APPEND ?LIFEGAME.ASM

Wenn jetzt rechts "NOT FOUND" erscheint, dann haben Sie irgend etwas falsch gemacht. Sicher aber erscheint die gewünschte Datei auf dem Bildschirm, sie steht tatsächlich auf der Diskette!

Formatieren von Disketten

Ehe Sie nun Ihren sämtlichen Dateien von der Cassette auf die Diskette übertragen, lesen Sie noch den folgenden Abschnitt. Es könnte ja sein, daß Sie sich gleich zwei Laufwerke zugelegt haben. In diesen Fall erscheint es als zweckmäßig, die Dateien nicht auf die im Laufwerk A befindliche "Systemdiskette" (mit den Betriebssystemen), sondern auf die Diskette im Laufwerk B zu schreiben. Aber auch mit nur einem Laufwerk ist es sinnvoll, eine weitere Diskette zum Abspeichern von Dateien anzulegen.

Nun genügt es nicht, wenn Sie eine Diskette kaufen und in das Laufwerk B stecken, das Betriebssystem kann mit ihr nichts anfangen. Die Magnetschicht ist zunächst ein unbeschriebenes Blatt, es sind weder Spuren noch Sektoren irgendwie gekennzeichnet. Die Markierung dieser Grundeinteilung auf der Cassette nennt man Formatieren, auf der CP/M-Systemdiskette gibt es dafür ein Formatierprogramm, das mit UFORM.COM bezeichnet ist.

Dieses Formatierprogramm ist recht bedienungsfreundlich angelegt, es erklärt sich quasi selber. Legen Sie also die neue Diskette ins Laufwerk B ein und geben Sie auf der Tastatur UFORM ein, danach CR. Es erscheint der Programmtitel und danach die Aufforderung, die zu formatierende Diskette einzulegen und die Systemdiskette herauszunehmen. Letzteres muß geschehen, wenn nur ein Laufwerk

vorhanden ist. Nach dem gezeigten Menü wählen Sie aus:

NDR- DD DS 80 SPUR =5

Drücken Sie also "5", danach kommt wieder ein Menue, bei dem Sie

FLO2-Maxi/Mini (OCOh) =1

die "1" wählen. Im nächsten Menü wird das Laufwerk gewählt, also die "2" für Laufwerk B oder, wenn nur ein Laufwerk vorhanden ist, "1" für dieses Laufwerk A. Der letzte Text auf dem Bildschirm informiert über den Formatiervorgang , weist noch einmal auf das benutzte Laufwerk hin und verlangt die Eingabe von "J" (Ja), wodurch das Programm gestartet wird.

Eine langsam wachsende Reihe von "F" und das hörbare Geräusch des Schrittmotors für den Schreib/Lesekopf zeigt an, daß Spur um Spur formatiert wird. Anschließend wird die Sache noch einmal überprüft, auch Spur für Spur, was jeweils mit einem "V" auf dem Bildschirm angezeigt wird. Nach getaner Arbeit erscheint wieder das letzte Menü, die Eingabe von "0" und danach "CR" beendet das Programm.

Um mit der frisch formatierten Diskette, die ja zunächst als Datenträger für Programme, Texte usw. gedacht ist, auch schon mal booten zu können, empfiehlt es sich, gleich noch das Betriebssystem zu kopieren. Für das eigentliche Betriebssystem sind die Spuren 0 und 1 reserviert, auf die man bei normalem Betrieb nicht zugreifen kann. Um das Betriebsystem zu kopieren, gibt es auf Ihrer Systemdiskette ein Programm SYSGEN80.COM. Geben Sie also ein:

```
A>SYSGEN80
SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP)
```

Es wird nach dem Quell-Laufwerk gefragt, geben Sie "A" ein:

```
SOURCE DRIVE NAME (OR RETURN TO SKIP)A
SOURCE ON A, THEN TYPE RETURN                ("CR" eingeben)
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO SKIP)
```

Jetzt wird nach dem Ziellaufwerk gefragt, geben Sie "B" ein (oder, wenn nur ein Laufwerk vorhanden ist, nehmen Sie die Systemdiskette heraus und legen Sie die neue Diskette ein und geben dann natürlich "A" als Ziellaufwerk ein:

```
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)B
DESTINATION ON B, THEN TYPE RETURN          ("CR" eingeben)
```

Nach dem (kurzen) Kopiervorgang erscheint die gleiche Frage noch einmal, die jetzt mit "CR" beantwortet wird, es erscheint wieder der Prompt "A>".

Jetzt können Sie die frisch fabrizierte Diskette in das A-Laufwerk stecken, einen RESET veranlassen und nach dem Anfangsmenü mit "1" booten -das Betriebssystem ist auf dieser Diskette vorhanden und Sie können weitere Dateien von der Cassette auf diese Diskette schreiben.

CP/M-Kommandos

Mit SAVE, UFORM, und SYSGEN80 haben Sie schon einige CP/M-Kommandos bzw. Betriebsprogramme kennengelernt. Bevor es ans Programmieren geht, werden noch einige Teile von CP/M vorgestellt, die Sie beim Umgang mit der ZEAT-Diskettenversion immer wieder benötigen.

Das Kommando DIR haben Sie bereits benutzt, es gibt das Inhaltsverzeichnis einer Diskette auf dem Bildschirm aus. Wenn die Diskette mit dem CP/M-Betriebssystem und dem dazu gefügten ZEAT im Laufwerk A steckt, dann erscheint nach der Eingabe von "DIR" eine stattliche Liste:

```
A>DIR
A: ZEAT    COM : MOV80CPM COM : SYSGEN80 COM : UFORM   COM
A: SPEED   MAC : SPEED    COM : PIP      COM : SUBMIT  COM
A: XSUB    COM : ED       COM : ASM      COM : DDT     COM
A: LOAD    COM : STAT     COM : SYSGEN   COM : DUMP    COM
A: DUMP    ASM : BIOS     ASM : CBIOS    ASM : DEBLOCK ASM
A: DISKDEFLIB : BOOT80    ASM : BIOS80   ASM : NDR80   ASM
A: INHALT BAK : ASSIGN    COM : SER      COM : INHALT  TXT
```

(Diese Directory muß bei Ihrer Diskette nicht unbedingt gleich aussehen.)

Es kann ja sein, daß Ihnen beim Anblick eines solchen Inhaltsverzeichnisses ein Dateiname nicht gefällt. Aus irgendeinem Grund paßt er nicht mehr , wird vielleicht für eine anderere Datei benötigt und sollte geändert werden. Das englische Wort Rename bedeutet umbenennen, das Kommando, mit dem ein Dateiname umbenannt wird, heißt dementsprechend "REN".

Probieren Sie das aus, aber bitte nicht mit den obenstehenden CP/M-Dateien, sondern beispielweise mit einer der von der Cassette übernommenen Dateien. Das sieht dann so aus:

A>REN LIFEGAME.ASM=PROGRAMM.ASM

Die Datei LIFEGAME.ASM soll also jetzt PROGRAMM.ASM heissen. Wenn die Datei auf einer Diskette im Laufwerk B steht, heißt es:

A>REN B: LIFEGAME.ASM=PROGRAMM.ASM

Nach dem Ausprobieren läßt sich selbstverständlich die alte Ordnung wiederherstellen: mit einem weiteren "REN" können Sie die Datei wieder umtaufen, damit der alte Name wieder da steht.

Ein weiteres Kommando wird benötigt, wenn eine Datei gelöscht werden soll. Auch hier ist der Name des Kommandos die Abkürzung des englischen Ausdrucks. "ERA" kommt von Erase = ausstreichen. Beim zuvor angegebenen Beispiel hiesse es:

A>ERA PROGRAMM.ASM oder:

A>ERA B: PROGRAMM.ASM

Dieses Kommando sollten Sie nur dann ausprobieren, wenn tatsächlich eine Datei vorhanden ist, die nicht mehr benötigt wird.

Ein wenig Vorsicht oder besser gesagt Überlegung vor Gebrauch empfiehlt sich auch beim Kommando "TYPE". Dieses CP/M-Kommando gibt den Inhalt einer ASCII-Datei auf dem Bildschirm aus. Diese Datei kann ohne weiteres mit Hilfe des ZEAT-Editors geschrieben sein. Beispiel:

A>TYPE PROGRAMM.ASM

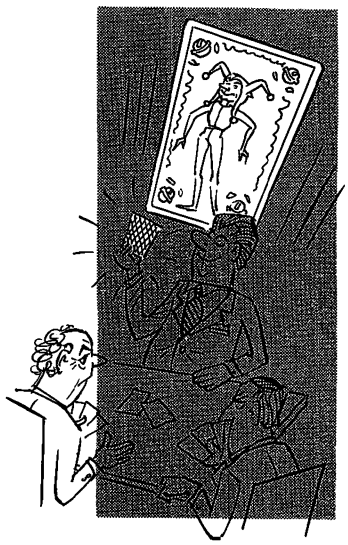
Wenn Sie vor dem TYPE-Kommando noch CTRL-P eingeben, erfolgt sogar die gleichzeitige Ausgabe auf einem angeschlossenen Drucker (mit einem weiteren CTRL-P kann der Drucker jederzeit wieder angehalten werden). Es muß aber eine ASCII-Datei sein, bei einem COM-File (lauffähiges Programm xxx.COM) gibt's Kakao auf dem Bildschirm, weil das System versucht, aus dem Binärcode ASCII-Zeichen zu entziffern.

Weiterhin wichtig ist der Unterschied zum ZEAT-Kommando "TYPE" bzw. "TYPE/L". Hier wird keine Datei von der Diskette, sondern der Inhalt des (bei ZEAT angelegten) Textspeichers ausgegeben.

Auf Seite B 41 wurde gesagt, daß Dateinamen bei CP/M aus dem eigentlichen Namen, einem Punkt und der Erweiterung bestehen, wobei jeweils Buchstaben oder Ziffern erlaubt sind. Darüber hinaus sind auch Zeichen erlaubt, allerdings nicht die folgenden Zeichen:

< > . , ; : = ? * sowie eckige Klammern

Zu den erlaubten Zeichen gehören also beispielsweise Bindestriche, Schrägstrich, das &-Zeichen und das !-Zeichen.



Bei einigen Kommandos gibt es zur Erleichterung ihrer Handhabung noch zwei spezielle Zeichen, nämlich den Stern (*) und das Fragezeichen (?). Die englische Bezeichnung für die beiden Zeichen lautet "Wildcard" und läßt sich sinngemäß mit Joker übersetzen, denn wie ein Joker können sie für andere Karten spielen: das Fragezeichen ersetzt einzelne Buchstaben im Dateinamen und der Stern einen ganzen Namensteil.

Wenn man z.B. von einer Diskette nicht das ganze Inhaltsverzeichnis braucht, sondern nur wissen will, welche Dateien mit der Kennzeichnung ASM vorhanden sind, dann lautet die Eingabe:

```
A>DIR *.ASM
```

Oder, das größte Unheil, das Sie anrichten können (falls es nicht Absicht ist), passiert mit

```
A>ERA *.*
```

Da wird bequem mit einem einzigen Kommando eine ganze Diskette gelöscht!

Das Fragezeichen kann als Joker eingesetzt werden, wenn Dateien vorhanden sind, die durch ihre Namensgebung Seriencharakter haben, z.B.

```
TEST-10.ASM
TEST-12.ASM
TEST-14.ASM
```

Mit dem Kommando "DIR TEST-??.ASM" wird diese Dateienserie erfaßt.

Die beiden Wildcard-Zeichen * und ? können bei den bisher vorgestellten CP/M-Kommandos "DIR" und "ERA" benutzt werden. Sehr gebräuchlich sind sie bei dem Kommando PIP, das meist zum Kopieren von Dateien verwendet wird und das wir Ihnen noch ausführlich vorstellen werden. An dieser Stelle sei noch auf eine grundsätzliche Unterscheidung der CP/M-Kommandos hingewiesen.

Es ist Ihnen sicher aufgefallen, daß mit dem Kommando UFORM oder auch SYSGEN80 (Seite B 43 bzw. B 44) ein lauf-fähiges Programm in den Speicher geladen wurde. Anders verhält es sich mit dem Kommando SAVE. Zur Ausführung der von ihm erwarteten Aktivitäten braucht kein COM-File in den Speicher geladen zu werden. Die zum Abspeichern einer Datei auf Diskette notwendigen Befehle sind bereits im Betriebssystem selbst enthalten. Das SAVE-Kommando ist eines der fünf residenten Kommandos, die im Betriebssystem sozusagen fest eingebaut sind. Alle fünf sind auf diesen Seiten vorgestellt worden: DIR, REN, ERA, SAVE, TYPE.

Ein weiteres Kommando gehört zu dieser Gruppe der residenten auch noch dazu, nämlich das Kommando zur Umschaltung des jeweils angesprochenen Laufwerks. Das Umschalten wird einfach durch die Eingabe der Laufwerkbezeichnung mit nachfolgendem Doppelpunkt und (wie bei allen CP/M-Kommandos) anschließender Betätigung der Taste CR (bzw. RETURN) erreicht:

```
A>B:
B>
```

UFORM und SYSGEN80 sind Beispiele für die transienten Kommandos, deren ausführende Befehle nur vorübergehend in den Speicher geholt werden. Von dieser Art CP/M-Kommandos werden Sie, wie schon erwähnt, noch einige kennenlernen.

Zusammenfassung

Ein Diskette muß vor ihrer Benutzung als Datenspeicher formatiert und zweckmäßigerweise mit einem Betriebssystem versehen werden. Die dazu benötigten CP/M-Kommandos gehören zu den residenten (im Betriebssystem enthaltenen) und den transienten (als eigene Programmdateien vorhandenen) Kommandos. Mit Hilfe dieser Kommandos ist es auch möglich, Dateien von Cassette auf Diskette zu übertragen. Dabei muß der Dateiname den von CP/M verlangten Spezifikationen entsprechen.

Frage:

1) Es könnte in seltenen Fällen notwendig werden, Dateien von Diskette auf Cassette zu übertragen. Überlegen Sie sich, wie das vor sich geht!

(Die Antwort zu dieser Frage finden Sie auf Seite G 19.)

Systemfunktionen für Diskettenbetrieb

In der auf Seite B 32 (Heft 4) stehenden Bios-Einsprungtabelle gab es auch Eintragungen für Unterprogramme, die das Lesen und Schreiben von Disketten erledigen:

Adresse	Programm
xx00H:	Kaltstartprogramm
xx03H:	Warmstartprogramm
xx06H:	Konsolstatus - 0FFH in A wenn Zeichen bereit, sonst 00H in A
xx09H:	Konsoleingabe - Zeichen von Konsole in A
xx0CH:	Konsolausgabe - Zeichen in C an Konsole
xx0FH:	Druckausgabe - Zeichen in C an Drucker
xx12H:	Stanzenausgabe - Zeichen in C an Stanze (oder serielle Schnittstelle)
xx15H:	Streifenlesereingabe - Zeichen von Leser (oder serieller Schnittstelle) in A
xx18H:	Home - Kopf auf Spur 0 (nur für Diskettensysteme)
xx1BH:	Laufwerk - Laufwerk (C) ansprechen (nur für Diskettensysteme)
xx1EH:	Spur - Spur (BC) wählen (nur für Diskettensysteme)
xx21H:	Sektor - Sektor (BC) wählen (nur für Diskettensysteme)
xx24H:	DMA - Transferadresse einstellen (nur für Diskettensysteme)
xx27H:	Read - Sektor lesen (nur für Diskettensysteme)
xx2AH:	Write - Sektor schreiben (nur für Diskettensysteme)
xx2DH:	Druckerstatus - 0FFH, wenn Drucker bereit, sonst 00H in A nicht immer implementiert!!
xx30H:	Sektorübersetzung (nur für Diskettensysteme)

Man könnte hergehen und mit diesen Bios-Unterprogrammen ein Laufwerk, eine Spur, einen Sektor und eine Transferadresse wählen. Anschließend könnte man z.B. mit einem Lesebefehl diesen Sektor an die vorher definierte Transferadresse im Speicher des Mikroprozessors bringen. Wird dieser Vorgang Sektor für Sektor wiederholt, dann lassen sich ganze Dateien über diese Einsprünge einlesen. Aber nur scheinbar, den genau das wollen wir nicht tun!

Bei dieser Methode müßten wir uns ja selbst merken, auf welchen Spuren und Sektoren eine Datei abgelegt ist. Dabei hat das Betriebssystem doch ein Inhaltsverzeichnis auf der Diskette angelegt, in dem diese Informationen bereits verschlüsselt sind. Keine Angst, um diese Ver-

schlüsselung müssen wir uns nicht kümmern. Das Betriebssystem stellt uns nämlich Systemfunktionen zur Verfügung, welche die richtigen Sektoren einer Datei auch ohne unser Zutun finden. Von diesen Systemfunktionen wird auf den folgenden Seiten die Rede sein.

Wie bei den anderen Systemfunktionen greift das Betriebssystem auch bei den Diskettenfunktionen letztendlich auf die Bios-Unterprogramme zurück. Darauf ist es auch zurückzuführen, daß im CP/M-Betriebssystem immer nur 128 Bytes auf einmal gelesen oder geschrieben werden können - das ist ein Sektor im ursprünglichen CP/M-Standardformat. Falls die Sektoren einer Diskette größer sind, muß das Bios die tatsächlichen Sektoren aus den für das CP/M-System benötigten 128 Bytes zusammen- und auseinander-"basteln". In der Fachsprache nennt man das "Sector-Blocking-Deblocking".

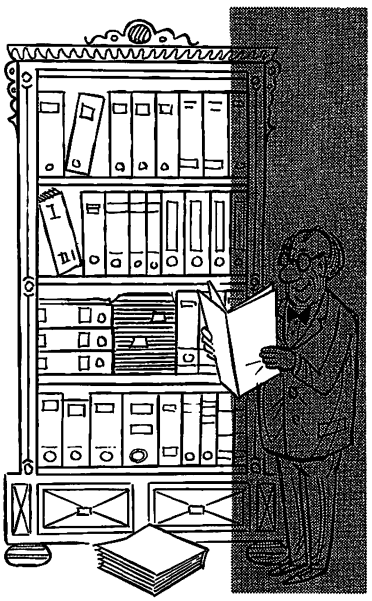
Dies hat übrigens nichts zu tun mit dem Unterprogramm für Sektorübersetzung, das in der Bios-Einsprungtabelle aufgeführt ist. Dessen Aufgabe ist eine andere: normalerweise sind die Sektoren einer Spur auf der Diskette (man sagt "physikalisch" und meint "wirklich dort vorhanden") durchgehend numeriert. Viele Dateien sind nun in aufeinanderfolgenden Sektoren auf der Diskette abgelegt, obwohl das nicht so sein muß. Es hat aber Vorteile hinsichtlich der Geschwindigkeit, wenn nicht dauernd die Spur gewechselt werden muß.

Hat das Betriebssystem einen Sektor gelesen, verarbeitet und will den nächsten Sektor lesen - schwupp, schon hat sich die Diskette während der Verarbeitungszeit weitergedreht und der zu lesende Sektor ist längst am Kopf vorbei. Jetzt muß wohl oder übel eine ganze Umdrehung der Diskette abgewartet werden, bis der Sektor wieder zum Kopf kommt. Um das zu vermeiden, werden die Sektoren vom Programm für Sektorübersetzung einfach umnumeriert. Es stehen dann z.B. zwischen dem ersten und zweitem ("logischen") Sektor fünf andere Sektoren: während die durchlaufen, ist der erste Sektor verarbeitet und der zweite kann gelesen werden. Auf der Diskette wäre dann logischer Sektor 1 gleich physikalischer Sektor 1, logischer Sektor 2 aber ist physikalischer Sektor 6. Damit wird das System wesentlich schneller.

Es ist ganz interessant, wen man sich solche Zusammenhänge einmal klar macht. Aber die Systemfunktionen, um die es hier geht, sind ja gerade dazu da, um die Einzelheiten der Diskettenaufzeichnung vergessen zu machen.

Grundsätzlich kann man den Ablauf der Diskettenoperationen recht schnell verstehen, wenn man die Datenverwaltung auf der Diskette als Denkmodell mit derjenigen in

einem Aktenschrank vergleicht. Letzterer entspricht dabei der Diskette und die mit einem Namen versehenen einzelnen Aktenordner den Dateien auf der Diskette. Soll nun eine Datei (ein Ordner) gelesen werden, dann muß diese Datei (Ordner) zunächst gesucht und dann geöffnet werden. Dieser Vorgang wird tatsächlich "Datei öffnen" ("Open File") genannt.



Danach können die einzelnen Datensätze der Datei (die Blätter des Ordners) nacheinander entnommen und irgendwo auf dem Tisch abgelegt werden. Dieser Vorgang wird sequentielles Lesen ("Read Sequential") genannt. Anders geht es beim ebenfalls möglichen wahlfreien Lesen ("Read Random"): hier können die Datensätze (die Blätter) in beliebiger Reihenfolge entnommen werden. Zum Abschluß des ganzen Vorgangs muß der Aktenordner wieder geschlossen werden. Dasselbe gilt auch für die Datei auf der Diskette: sie wird geschlossen ("Close File").

Stören Sie sich nicht daran, daß der Vergleich ein wenig hinkt: die von der Diskette geholten Daten sind selbstverständlich nur kopiert, sie stehen also noch dort. Der aus dem Aktenschrank geholte Ordner bzw. die aus ihm entnommenen Blätter müßten also kopiert und die Original umgehend wieder abgelegt werden. Aber so genau wollen wir es nicht nehmen, es soll ja nur ein sinnfälliger Vergleich zum besseren Verständnis sein.

Wird jetzt eine neue Datei angelegt, dann kann natürlich nicht die Funktion zum Öffnen verwendet werden - klar, ein noch nicht vorhandener Ordner kann auch nicht geöffnet werden. Hier gibt es eine Funktion zum Anlegen einer Datei ("Make File"), nach deren Aufruf die einzelnen Datensätze ("Records", je 128 Bytes) nacheinander auf die Diskette geschrieben werden, entsprechend dem Ablegen der Blätter im Aktenordner.

Mit der ebenfalls vorhandenen Funktion zum wahlfreien Schreiben ("Write Random") können die Datensätze statt nacheinander auch in beliebiger Reihenfolge abgelegt werden. Nach dem Abschluß muß der Ordner natürlich ordentlich wieder geschlossen werden: die neu geschriebene Datei wird mit "Close File" geschlossen.

Sie werden es schon ahnen, daß die Systemfunktionen für Diskettenoperationen etwas aufwendiger sind als z.B. diejenigen für Konsoleingabe. Das hat zwei Gründe:

Erstens werden im CP/M-Betriebssystem immer nur Blöcke zu 128 Bytes ("Records") auf einmal gelesen oder geschrieben. Für die Übertragung dieser Daten muß irgendwo ein Speicherbereich reserviert werden, denn 128 Bytes lassen sich nicht auf einmal in CPU-Registern übermitteln. Die

"Records" entsprechen den Blättern im Aktenordner und der Speicherbereich für die Übertragung einer Stelle auf dem Schreibtisch, wo die Blätter abgelegt werden sollen.

Zweitens muß dem Betriebssystem der Name der gewünschten Datei mitgeteilt werden. Das ist ebenfalls nicht mit den CPU-Registern zu machen.

Das erste Problem wird im CP/M-Betriebssystem ganz einfach gelöst: es wird vor jeder Lese- oder Schreiboperation eine Systemfunktion aufgerufen, mit der eine Transferadresse definiert wird.

Diese Funktion wird "Set DMA Adress" genannt, weil in vielen Systemen der Datentransfer von einer DMA-Logik durchgeführt wird. DMA heißt "Direct Memory Access" (direkter Speicherzugriff). Ein DMA-Baustein kann ganze Datenblöcke im Speicher verschieben oder von einem externen Gerät einlesen, wobei die CPU nicht benötigt wird - das führt zu einem schnelleren Ablauf.

Ob der Datentransfer nun tatsächlich durch einen DMA-Baustein oder durch die CPU erledigt wird, braucht uns nicht zu interessieren. Es muß nur definiert werden, wohin die von der Diskette gelesenen Daten gebracht werden bzw. von wo die auf die Diskette zu schreibenden Daten geholt werden sollen.

Die Systemfunktion zum Setzen der Transferadresse hat die Nummer 26 und übergibt die Transferadresse im DE-Registerpaar:

Funktion 26 - Setze DMA-Adresse (Set DMA)

Parameter beim Aufruf:	Register C: 1AH Registerpaar DE: Transferadresse für Diskettenoperationen
Parameter beim Rücksprung:	keine

Mit dieser Funktion kann also vor einer Leseoperation bestimmt werden, an welcher Adresse im Hauptspeicher der Datensatz von 128 Bytes gelesen werden soll, bzw. vor einer Schreiboperation, von welcher Adresse im Hauptspeicher der Datensatz auf Diskette geschrieben werden soll.

Bei einer Leseoperation muß selbstverständlich dafür ge-

sorgt werden, daß mit der eingestellten Adresse keine Bereiche eines Programms oder gar des Betriebssystems überschrieben werden.

Wenn keine DMA-Adresse mit der Funktion 26 gewählt wurde, verwendet CP/M die Adresse 0080H für diesen Zweck, die Daten werden dann von bzw. an Adresse 0080H bis 00FFH transferiert. Bei manchen Anwendungen genügt dieser Bereich, in diesen Fällen braucht die Funktion 26 gar nicht aufgerufen zu werden.

Das zweite Problem, die Übermittlung des Dateinamens, wird auf ähnliche Weise gelöst, wie es bei der Systemfunktion 10 (Seite B 28, Heft 3) geschieht. Zur Übermittlung der notwendigen Informationen wird ein bestimmter Datenbereich eingerichtet. In den entsprechenden Systemfunktionen braucht dann nur noch die Adresse dieses Datenbereichs angegeben zu werden, dann weiß das Betriebssystem, was es zu tun hat.

Bei Diskettenoperationen heißt dieser Datenbereich "File Control Block", also Datei-Steuer-Block. In ihm steht nicht nur der Dateiname, sondern auch die übrigen Informationen, die für die Bearbeitung einer Datei notwendig sind: z.B. muß sich das Betriebssystem irgendwo merken, wie viele Datensätze einer Datei es schon bearbeitet hat, was also der nächste zu bearbeitende Datensatz ist.

Die Bezeichnung "Datei-Steuer-Block" ist kaum gebräuchlich, in allen Unterlagen verwendet man die Abkürzung der englischen Bezeichnung und spricht vom "FCB".

Zweierlei ist nach dem Vorhergesagten also zu überlegen, bevor irgendwelche Diskettenoperationen ausgeführt werden können: erstens, welche Transferadresse angegeben wird, und zweitens muß für jede Datei ein FCB eingerichtet werden. Dieser hat folgendes Aussehen:

FCB	lw	n1	n2	....	n8	t1	t2	t3	ex	s1	s2	rc	d0	....	dn	cr	r0	r1	r2
Adressen	00	01	02	....	08	09	10	11	12	13	14	15	16	....	31	32	33	34	35

Der FCB besteht aus 36 Bytes, wobei die drei letzten bei normalen Dateioperationen auch weggelassen werden können. Die einzelnen Felder des FCB haben folgende Bedeutung:

lw Laufwerk: 0=angemeldetes Laufwerk, 1=A, 2=B usw

n1 .. n8 Dateiname, 8 ASCII-Zeichen (keine Kleinbuchst.)

t1 .. t3 Dateityp, 3 ASCII-Zeichen (keine Kleinbuchst.)

ex	"extent number"
s1,s2	nur intern benutzt
rc	"record count"
d0 .. dn	nur intern benutzt
cr	"current record"
r0 .. r2	"random record number" bei wahlfreien Schreib-Lese-Operationen

In den Feldern 0 bis 11 des FCB muß die Bezeichnung des Laufwerks und der Dateiname der gewünschten Datei eingetragen werden. Um die genaue Bedeutung der Felder 12 bis 23 brauchen Sie sich nicht zu kümmern. Wichtig ist lediglich, daß einige davon (ex, s2, cr) vor einem Aufruf der Funktionen "Datei öffnen", "Datei anlegen" und "Datei suchen" auf Null gesetzt werden müssen. Die Felder 33 bis 35 werden nur bei wahlfreien Schreib/Leseoperationen benötigt - damit beschäftigen wir uns erst später.

Sie brauchen sich also von den vielfältigen Bezeichnungen der Felder des FCB nicht verwirren lassen. Es kann überhaupt nichts schiefgehen, wenn beim Umgang mit FCB's folgendes beachtet wird:

> Vor dem Öffnen ("Open") oder Anlegen ("Make") einer Datei gewünschtes Laufwerk in Feld 0 eintragen. Dateiname und Typ in Felder 1 bis 11 eintragen, restliche Felder auf Null setzen und nachher nichts mehr am FCB ändern!

Ein FCB kann in einem Programm z.B. wie folgt definiert werden:

```
FCB:
   DB   0              ;angemeldetes Laufwerk verwenden
   DB   'BRIEF  TXT'   ;8 Bytes Dateiname, 3 Bytes Datei-
                       ;typ
   DS   24,0           ;24 Bytes auf Null setzen
```

Meist kann der FCB aber bei der Programmentwicklung noch nicht vollständig definiert werden. Der Name der Datei z.B. wird erst beim Ablauf des Programms gewählt und sollte nicht schon beim Entwurf des Programms festgelegt werden, weil das Programm dann wenig flexibel ist. Deshalb wird in der Regel für den FCB zunächst nur der benötigte Speicherplatz reserviert und der gewünschte Inhalt dann vom Programm eingetragen. Dafür gibt es von CP/M und ZEAT auch noch Unterstützung.

Ist ein FCB erst einmal angelegt, dann ist das Lesen und Schreiben von Dateien völlig problemlos möglich. Man verwendet die folgenden Systemfunktionen für Datei öffnen ("Open"), Datei lesen ("Read") oder schreiben ("Write") und schließlich Datei schließen ("Close"). Wir geben hier bewußt oft die englischen Bezeichnungen an, weil sie Ihnen beim Umgang mit fremden Programmen häufiger begegnen werden als die deutschen Übersetzungen.

Funktion 15 - Öffne Datei (Open File)

Parameter beim Aufruf: Register C: 0FH
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0 bis 3, falls oK,
0FFH, falls Datei nicht gefunden

Dem System wird die Adresse eines FCB übergeben, in dem die Laufwerkbezeichnung, Name und Typ der gewünschten Datei eingetragen und die restlichen Felder (mindestens aber ex, s2, cr) auf Null gesetzt sind. Das System trägt in den FCB die für die weitere Verarbeitung notwendigen Daten ein und kehrt mit Register A gleich 0, 1, 2 oder 3 zurück, falls die gewünschte Datei gefunden wurde. Sonst ist Register A gleich 0FFH beim Rücksprung. Im Dateinamen und Typ sollten keine Fragezeichen vorkommen, da sonst die erste Datei verwendet wird, deren Name in den restlichen Stellen mit dem FCB übereinstimmt.

Die geöffnete Datei kann anschließend gelesen aber auch beschrieben werden (wobei die alten Datensätze überschrieben werden).

Programmbeispiel:

```
SYSTEM  EQU   0005H
OPENF   EQU   15

        LD    DE, FCB       ;FCB übergeben
        LD    C, OPENF      ;Funktion wählen
        CALL  SYSTEM        ;und System aufrufen
        CP    0FFH          ;prüfen ob Fehler ...
        JR    Z, ...
```

Funktion 22 - Datei anlegen (Make File)

Parameter beim Aufruf: Register C: 16H
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0bis 3, falls oK,
OFFH, falls Inhaltsverz. voll

Dem System wird die Adresse eines FCB übergeben, in dem die Laufwerksbezeichnung sowie Name und Typ der gewünschten Datei eingetragen sind. Das System legt die Datei auf der Diskette an, trägt in den FCB die für die weitere Verarbeitung notwendigen Daten ein und kehrt mit Register A gleich 0, 1, 2 oder 3 zurück, wenn die gewünschte Datei angelegt werden konnte. Wenn das Inhaltsverzeichnis voll ist, steht in Register A beim Rücksprung OFFH. Im Dateinamen und Typ dürfen keine Fragezeichen vorkommen und der Dateiname darf auf der Diskette noch nicht vorhanden sein, sonst gibt es zwei Dateien mit dem gleichen Namen.

Um dies zu verhindern, kann mit einer "Open"-Funktion erst einmal geprüft werden, ob der gewünschte Dateinamen schon vorhanden ist. Mit einer (noch zu beschreibenden) Löschfunktion ("Delete File") kann eine eventuell vorhandene Datei mit dem gleichen Namen einfach gelöscht werden. Dies ist natürlich mit Vorsicht zu genießen - gelöscht ist gelöscht!

Programmbeispiel:

```
SYSTEM  EQU   005H
MAKEF   EQU   22

        LD    DE,FCB        ;FCB übergeben
        LD    C,MAKEF       ;Funktion wählen
        CALL  SYSTEM        ;und System aufrufen
        CP    OFFH          ;prüfen ob Fehler ...
        JR    Z,  ...
```

Funktion 20 - Datensatz lesen (Read Sequential)

Parameter beim Aufruf: Register C: 14H
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0,falls ok, ungleich 0, falls Dateiende erreicht

Dem System wird die Adresse eines FCB übergeben, der vorher mit einer "Open"- oder "Make"-Funktion aktiviert wurde. Das System liest den nächsten Datensatz (Record) an die mit der Funktion 26 festgelegte Adresse und erhöht die Eintragungen in den entsprechenden Feldern des FCB (cr und ex), damit beim nächsten Aufruf der nächste Datensatz gelesen wird. Nach einem erfolgreichen Lesevorgang ist das Register A gleich Null. Falls das Dateiende erreicht wurde und deshalb kein Datensatz mehr gelesen werden konnte, enthält es einen Wert ungleich Null. Das ist beispielsweise auch beim Lesen einer gerade mit "Make" neu angelegten und noch leeren Datei der Fall.

Wird eine ganze Datei eingelesen, dann muß vor jedem neuen Aufruf der Lesefunktion der zuletzt gelesene Datensatz von der Transferadresse wegtransportiert oder diese geändert werden. Im anderen Fall werden die Daten laufend überschrieben.

Außerdem, wie bereits erwähnt, muß vor dem Lesevorgang sichergestellt sein, daß die Transferadresse (DMA-Adresse) so gewählt ist, daß kein Programm oder vom Programm benötigte Datenbereiche oder gar das Betriebssystem überschrieben werden. Das System würde mit Sicherheit abstürzen!

Programmbeispiel:

```
SYSTEM  EQU  0005H
READF   EQU  20

        LD   DE, FCB        ;FCB übergeben
        LD   C, READF       ;Funktion wählen
        CALL SYSTEM         ;und System aufrufen
        OR   A              ;prüfen, ob Fehler  ...
        JR   Z, ...
```

Funktion 21 - Datensatz schreiben (Write Sequential)

Parameter beim Aufruf: Register C: 15H
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0 falls ok,ungleich 0, falls Diskette voll

Dem System wird die Adresse eines FCB übergeben, der vorher mit einer "Open"- oder "Make"-Funktion aktiviert wurde. Das System schreibt die 128 Bytes, die an der mit Funktion 26 festgelegten Transferadresse stehen, als nächsten Datensatz auf die Diskette und erhöht den Inhalt

der entsprechenden Felder im FCB (cr und ex), damit beim nächsten Aufruf der nächste Datensatz beschrieben wird. Nach einem erfolgreichen Schreibvorgang ist das Register A gleich Null. Wenn die Diskette voll war und deswegen der Datensatz nicht mehr geschrieben werden konnte, enthält es einen Wert ungleich Null.

Analog zum Vorgehen beim Lesen einer Datei mußder Inhalt des Speicherbereichs bei der Transferadresse vor jedem Aufruf der Schreibfunktion neu vorbereitet oder die Transferadresse geändert werden. Sonst werden immer wieder die selben Daten auf die Diskette geschrieben!

Progammbeispiel:

```
SYSTEM  EQU  0005H
WRITEF  EQU  20

        LD   DE, FCB        ;FCB übergeben
        LD   C,WRITEF       ;Funktion wählen
        CALL SYSTEM         ;und System aufrufen
        OR   A              ;prüfen,ob Fehler ...
        JR   Z ...
```

Funktion 16 - Datei schließen (Close File)

Parameter beim Aufruf: Register C: 10H
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0 bis 3, falls ok, ungleich 0 falls Dateiende erreicht

Dem System wird die Adresse eines FCB übergeben, der vorher mit einer "Open"- oder "Make"-Funktion aktiviert und mit "Read" oder "Write" verwendet wurde. Das System schreibt eventuell noch in Puffern stehende Daten auf die Diskette und vervollständigt die Eintragung im Inhaltsverzeichnis. Nach einem erfolgreichen Aufruf ist das Register A gleich 0, 1, 2 oder 3. Falls die im FCB angegebene Datei nicht gefunden wurde, ist das Register A gleich OFFH. Normalerweise kann dieser Fehler nicht auftreten, da eine mit "Open" oder "Make" geöffnete oder angelegte Datei auch im Inhaltsverzeichnis vorhanden ist.

Falls die betreffende Datei nur gelesen wurde, kann das Schließen entfallen. Aber immer, wenn eine Datei geschrieben wurde, muß sie unbedingt auch geschlossen werden, damit der Eintrag der Daten auf der Diskette vollständig ist.

Programmbeispiel:

```
SYSTEM  EQU  0005H
CLOSEF  EQU  20
        LD   DE, FCB        ;FCB übergeben
        LD   C, CLOSEF      ;Funktion wählen
        CALL SYSTEM         ;und System aufrufen
        CP   0FFH           ;prüfen ob Fehler ...
        JR   Z ...
```

Jetzt kennen Sie schon die wichtigsten Funktionen, die für das Programmieren von Diskettenoperationen notwendig sind: FCB anlegen, Datei öffnen, Daten lesen und schon ist eine Datei im Speicher. Der einzige Punkt, der etwas Aufwand beim Programmieren erfordern kann, ist Punkt 1: "FCB anlegen". Damit auch dies unproblematisch wird und Sie sich voll auf das eigentliche Programm konzentrieren können, gibt es im ZEAT noch eine nützliche erweiterte Systemfunktion. Mit ihrer Hilfe wird aus einem String, der einen Dateiname enthält, automatisch ein korrekter FCB erzeugt. Hat der String z.B. den Inhalt

```
B:PROG.ASM
```

dann erzeugt diese Systemfunktion die Folge von Sedezimalzahlen

```
02 50 52 4F 47 20 20 20 20 41 53 4D
B: P  R  O  G              A  S  M
```

wie sie im Dateisteuerblock (FCB) benötigt wird. Gleichzeitig werden dabei auch eventuell vorhandene Kleinbuchstaben in Großbuchstaben gewandelt.

Funktion 250 - Dateisteuerblock (FCB) erzeugen

Parameter beim Aufruf:
- Register C: 0FAH
- Registerpaar DE: Zeiger zu Dateibezeichnung in einen Textpuffer
- Registerpaar IY: Zeiger z.Textpuffer (wie bei Funktion 10)
- Registerpaar IX: Zeiger zu FCB

Parameter beim Rücksprung: Carry-Flag: normal gesetzt, bei Fehler gelöscht

Dem Betriebssystem wird im Registerpaar IY die Adresse eines Textpuffers übergeben, der nach der Konvention des Funktionsaufrufs 10 aufgebaut ist und normalerweise auch mit dem Aufruf der Funktion 10 gesetzt wird. Das Registerpaar DE zeigt auf die Stelle im Textpuffer, an der sich die Dateibezeichnung befindet. Falls sich im Textpuffer nur eine Dateibezeichnung befindet, also am Anfang des Inhalts, dann ist dies IY+2. Das Registerpaar IX zeigt zu der Adresse, wo der FCB aufgebaut werden soll. Wenn das Programm eine gültige Dateibezeichnung findet und einen FCB an der in IX bezeichneten Stelle aufbauen konnte, dann ist beim Rücksprung das Carry-Flag gesetzt, im Fehlerfall ist es gelöscht.(Beim Aufbau des FCFB wird der Inhalt des Textpuffers verschoben.)

Beachten Sie, daß dieses Programm auch * und ? richtig verarbeitet, also auch zweideutige FCB aufbauen kann. Die eingegebenen Dateinamen müssen also eindeutig sein, oder es muß nach dem Aufruf der Funktion 250 geprüft werden, ob sich im FCB Fragezeichen befinden.

Zusammenfassung

Im CP/M-Betriebssystem gibt es eine Reihe von Systemfunktionen für den Diskettenbetrieb, beispielsweise zum Anlegen, Öffnen, Lesen, Schreiben, Schließen einer Datei. Der Datentransfer geschieht dabei immer in Blöcken zu 128 Bytes, für die jeweils eine Speicheradresse als Transferadresse angegeben werden muß. Außerdem muß ein Dateisteuerblock (FCB) erzeugt werden. Er enthält den Namen der Datei und Informationen für die Bearbeitung. Transferadresse und FCB werden ebenfalls mit Systemfunktionen angegeben. Im ZEAT-Betriebssystem gibt es noch eine erweiterte Systemfunktion, die das Anlegen des FCB erleichtert.

Fragen:

1) Wie muß das Programm-Stück aussehen, in dem mit der Funktion 250 das FCB einer Datei Dat.dat auf einer Diskette angelegt werden soll?

2) Was wird dabei von der Funktion 250 gleich mit erledigt?

(Die Antworten zu diesen Fragen finden Sie auf Seite G20)

CP/M-Kommandos

Wenn man im letzten Programmbeispiel die Buchstabenumwandlung wegließe, dann könnte das Programm auch zum Kopieren von Textdateien benutzt werden. Dafür gibt es aber einen bequemeren Weg, nämlich das Systemprogramm PIP, mit dem beliebige Dateien kopiert werden können. PIP ist eines der auf Seite B 48 angekündigten weiteren transienten Kommandos. Es steht im Gegensatz zu den internen Kommandos (z.B. DIR) als .COM-Datei auf der Diskette und wird bei Bedarf in die TPA geladen. Das Kommando PIP kann übrigens auch unter ZEAT benutzt werden.

Mit dem Kommando PIP kann nicht nur zwischen Diskettendateien kopiert werden, sondern auch von Diskettendateien auf Computerschnittstellen und umgekehrt. Als Beispiel zeigen wir Ihnen das mit der Konsole. Übrigens, PIP kann noch viel mehr als nur Kopieren. Sie sollten sich daraufhin einmal mit dem CP/M-Handbuch beschäftigen.

Als erstes sehen wir uns das Inhaltsverzeichnis der Diskette im Laufwerk A an:

```
A>DIR
A: ZEAT    COM : MOV80CPM COM : SYSGEN80 COM : UFORM   COM
A: SPEED   MAC : SPEED    COM : PIP      COM : SUBMIT  COM
A: XSUB    COM : ED       COM : ASM      COM : DDT     COM
A: LOAD    COM : STAT     COM : SYSGEN   COM : DUMP    COM
A: DUMP    ASM : BIOS     ASM : CBIOS    ASM : DEBLOCK ASM
A: DISKDEFLIB : BOOT80    ASM : BIOS80   ASM : NDR80   ASM
A: INHALT BAK : ASSIGN    COM : SER      COM : INHALT  TXT
```

Das Inhaltsverzeichnis wird natürlich bei Ihnen anders als unser Beispiel aussehen, aber auch dort sind es sicher die .COM-Dateien, die auf dieser Diskette vorherrschen. Es soll nun mit PIP ein kurzer Text in eine Datei geschrieben werden, wobei dieser Text einfach von der Konsole (CON:) in die Datei kopiert wird.

Bei PIP wird zuerst das Ziel des Kopiervorgangs, ein Gleichheitszeichen und danach die Quelle angegeben. Sie sehen das an der nachstehenden Eingabe, bei der auch gleich ein Muster für den kurzen Text gezeigt ist. Wie alle CP/M-Kommandos wird das ganze mit CR (bzw. RETURN) abgeschlossen:

```
A>PIP DATEI1.TXT=CON:
```

Dies wird eine kleine Textdatei, die nur aus zwei Zeilen besteht und deren Eingabe mit CTRL-Z abgeschlossen wird:

```
^Z
```

Nach dieser Eingabe sieht das Inhaltsverzeichnis so aus:

```
A>DIR
A: ZEAT    COM : MOV80CPM COM : SYSGEN80 COM : UFORM    COM
A: SPEED   MAC : SPEED    COM : PIP      COM : SUBMIT   COM
A: XSUB    COM : ED       COM : ASM      COM : DDT      COM
A: LOAD    COM : STAT     COM : SYSGEN   COM : DUMP     COM
A: DUMP    ASM : BIOS     ASM : CBIOS    ASM : DEBLOCK ASM
A: DISKDEFLIB : BOOT80    ASM : BIOS80   ASM : NDR80    ASM
A: INHALT BAK : ASSIGN    COM : SER      COM : INHALT   TXT
A: DATEI1.TXT
```

Das Spielchen läßt sich fortsetzen, "DATEI1.TXT" wird in eine weitere "DATEI2.TXT" kopiert und dandach wieder ins Inhaltsverzeichnis geschaut:

```
A>PIP DATEI2.TXT=DATEI1.TXT

A>DIR
A: ZEAT    COM : MOV80CPM COM : SYSGEN80 COM : UFORM    COM
A: SPEED   MAC : SPEED    COM : PIP      COM : SUBMIT   COM
A: XSUB    COM : ED       COM : ASM      COM : DDT      COM
A: LOAD    COM : STAT     COM : SYSGEN   COM : DUMP     COM
A: DUMP    ASM : BIOS     ASM : CBIOS    ASM : DEBLOCK ASM
A: DISKDEFLIB : BOOT80    ASM : BIOS80   ASM : NDR80    ASM
A: INHALT BAK : ASSIGN    COM : SER      COM : INHALT   TXT
A: DATEI1.TXT : DATEI2.TXT
```

Das Ganze geht auch anders herum, z.B. von "DATEI2.TXT" auf die Konsole:

```
A>PIP CON:=DATEI2.TXT
```

Dies wird eine kleine Textdatei, die nur aus zwei Zeilen besteht und deren Eingabe mit CTRL-Z abgeschlossen wird:

Man sieht, daß das ASCII-Zeichen CTRL-Z (1AH), mit dem das Textende markiert ist, nicht mit ausgegeben wird. Als letztes Beispiel wollen wir noch eine Datei aus mehreren Teilen zusammenbasteln. Anschließend wird das Ergebnis im Inhaltsverzeichnis nachgeschaut :

```
A>PIP DATEI3.TXT=DATEI1.TXT,CON:,DATEI2.TXT
*** Diese Zeile kommt zwischen die anderen Dateien ***
^Z
A>DIR
A: ZEAT    COM : MOV80CPM COM : SYSGEN80 COM : UFORM    COM
A: SPEED   MAC : SPEED    COM : PIP      COM : SUBMIT   COM
A: XSUB    COM : ED       COM : ASM      COM : DDT      COM
A: LOAD    COM : STAT     COM : SYSGEN   COM : DUMP     COM
A: DUMP    ASM : BIOS     ASM : CBIOS    ASM : DEBLOCK ASM
A: DISKDEFLIB : BOOT80    ASM : BIOS80   ASM : NDR80    ASM
A: INHALT BAK : ASSIGN    COM : SER      COM : INHALT   TXT
A: DATEI1 TXT : DATEI2    TXT : DATEI3   TXT
```

Der Inhalt der zusammengebastelten DATEI3.TXT kann auf die Konsole kopiert und so besichtigt werden:

```
A>PIP CON:=DATEI3.TXT
Dies wird eine kleine Textdatei, die nur aus zwei Zeilen
besteht und deren Eingabe mit CTRL-Z abgeschlossen wird:
*** Diese Zeile kommt zwischen die anderen Dateien ***
Dies wird eine kleine Textdatei, die nur aus zwei Zeilen
besteht und deren Eingabe mit CTRL-Z abgeschlossen wird:
```

Wir haben mit PIP drei Textdateien erzeugt, die mit DIR auch richtig angezeigt werden, über die man aber allein aus dem Inhaltsverzeichnis nicht viel erfährt. Man kann sie mit dem transienten PIP-Kommando auf den Bildschirm (CON:) bringen oder den gleichen Effekt mit dem internen TYPE-Kommando erzielen (nicht unter ZEAT, dort hat TYPE eine andere Bedeutung). Bei Programmdateien geht das nicht, diese bestehen ja nicht aus ASCII-Zeichen. Genug der Beispiele zum Kommando PIP, obwohl gerade PIP noch über andere Fähigkeiten verfügt, die -wie schon gesagt- den entsprechenden Handbüchern entnommen werden können.

Mit einem anderen transienten Kommando, dem STAT-Kommando, läßt sich mehr erfahren:

```
A>STAT ZEAT.COM

 Recs  Bytes  Ext Acc
  138   18k    2 R/W A:ZEAT.COM
Bytes Remaining On A: 522k
```

Wie haben hier eine einzelne Datei als Beispiel herausgegriffen. Sie können bei Ihrem Computer "STAT *.*" eingeben, dann erhalten Sie eine alphabetisch sortierte Liste mit allen Dateien der angesprochenen Diskette. Am Schluß steht immer, egal ob bloß eine Datei wie oben oder mehrere oder alle angefordert werden, wieviel Platz auf der Diskette noch frei ist (im obigen Beispiel 522 kBytes).Das Kommando zeigt an, aus wievielen Datensätzen ("Records") oder Bytes eine Datei besteht. Es fällt auf, daß die Länge der Dateien immer ein Vielfaches von 2 kByte beträgt: das Betriebssystem vergibt den Speicherplatz auf der Diskette in diesem Laufwerk immer in Gruppen zu 2 kByte. Wichtig ist die Angabe, ob die Datei schreibgeschützt ist oder nicht. Im ersten Fall heißt es R/O ("Read/Only"), im zweiten Fall R/W ("Read/Write").

Mit dem Kommando STAT können wichtige Dateien schreibgeschützt werden:

```
A>STAT DATEI.TYP $R/O
```

Mit dem Parameter $R/W kann der Schreibschutz wieder aufgehoben werden, mit $SYS kann das Auflisten einer Datei im Inhaltsverzeichnis unterdrückt und mit $DIR wieder erlaubt werden. Beachten Sie, daß die Parameter mit dem Zeichen $ anfangen. Mit den beiden "Wild Card"-Zeichen (* und ?) können wie bei den anderen Kommandos auch mehrere Dateien auf einmal angesprochen werden .Es kann auch eine ganze Diskette schreibgeschützt werden:

```
A>STAT B:=R/O
```

Die Aufhebung des Schreibschutzes erfolgt in diesem Fall mit CTRL-C. Eine weitere Variante des STAT-Kommandos teilt die Parameter der aktiven Laufwerke mit:

```
A>STAT DSK:
    A: Drive Characteristics
 2448: 128 Byte Record Capacity
  306: Kilobyte Drive  Capacity
   64: 32  Byte Directory Entries
    0: Checked  Directory Entries
  256: Records/ Extent
   16: Records/ Block
   16: Sectors/ Track
    7: Reserved Tracks

    B: Drive Characteristics
 6320: 128 Byte Record Capacity
  790: Kilobyte Drive  Capacity
  128: 32  Byte Directory Entries
  128: Checked  Directory Entries
  128: Records/ Extent
   16: Records/ Block
   80: Sectors/ Track
    1: Reserved Tracks
```

Bei Ihnen ergeben sich an dieser Stelle ganz bestimmt andere Angaben, das gezeigte Beispiel stammt von einem speziellen Computer. Das angesprochene Laufwerk B ist hier ein 80-Spur Laufwerk mit einer Kapazität von 790 kByte. Das Laufwerk A ist sogar kein richtiges Diskettenlaufwerk, sondern ein sehr großer RAM-Speicher, dessen Inhalt in Gruppen zu 128 Bytes gelesen oder geschrieben werden kann, als ob es sich um eine Diskette handelte. Man nennt das eine "RAM-Disc" oder "Pseudo-Floppy". Der Vorteil ist die viel größere Verarbeitungsgeschwindigkeit (die Kopfpositionierung und das Suchen der Sektoren fällt weg). Der Nachteil ist derjenige aller RAM-Speicher: bei Stromausfall ist alles Gespeicherte zerstört.

Zurück zu PIP und STAT, mit Hintergedanken wurde hier das Kapitel über die zwei transienten Kommandos eingeschoben , Sie werden es gleich sehen.

Systemfunktionen

Die Systemprogramme PIP, STAT und andere Kommandos können sich ihre Befehlsparameter offensichtlich von der Kommandozeile holen, mit der sie aufgerufen werden. Das ist viel praktischer als die nachträgliche Eingabe jedes einzelnen Parameters oder Dateinamens. Also, was PIP kann, muß unser Programm auch können! Dafür muß ein Programm entworfen werden, das beispielsweise unter dem Namen "GROSS.COM" auf der Diskette abgelegt wird. Mit:

```
A>GROSS VERSUCH1.TLC
```

soll es veranlaßt werden, mit der Datei "VERSUCH1.TLC" die Buchstabenwandlung durchzuführen.

Das Betriebssystem (CP/M und ZEAT mit CP/M-Erweiterung) gibt den Anwenderprogrammen sogar zwei Möglichkeiten, den Rest der beim Aufruf des Programms stehenden Kommandozeile (im Beispiel "VERSUCH1.TLC") auszuwerten. Zunächst einmal legt das Betriebssystem die Länge und den Inhalt dieses Restes der Kommandozeile bei der Adresse 0080H ab. Dabei werden die Buchstaben gleich in Großbuchstaben gewandelt, was beim Aufbau eines FCB oder der Suche nach bestimmten Buchstaben die Arbeit erleichtert. Bei unserem Beispiel stünden nach dem Aufruf die folgenden Bytes an dieser Adresse:

```
*D 80,8F
0080 0D 20 56 45 52 53 55 43 48 31 2E 54 4C 43 00 00 . VERSUCH1.TLC
```

Die erste Stelle enthält die Anzahl der Zeichen (0DH=13). In den restlichen Stellen stehen die eingegebenen ASCII-Zeichen in Großschreibung, wobei der notwendige Zwischenraum zwischen Programmnamen (GROSS) und Dateinamen (VERSUCH1.TLC) mit übertragen wird. Es werden übrigens auch - abgesehen von der Wandlung in Großbuchstaben- nicht alle Zeichen unverändert übertragen. Ein Gleichheitszeichen erhält zusätzlich je einen Zwischenraum vor- und nachgestellt. Das zeigt als Beispiel der Aufruf eines PIP-Kommandos, mit dem alle Dateien vom Laufwerk A: auf das Laufwerk B: kopiert werden sollen:

```
A>PIP B:=A:*.*
```

übergibt an der Adresse 0080H:

```
0080  0B 20 42 3A 20 3D 20 41 3A 2A 2E 2A 00 00 00 00 . B: = A:*.*...
```

Auf diese Weise wird dem Kommando PIP die Analyse der Kommandozeile etwas erleichtert.

Offensichtlich ist diese Form der Übergabe einer Kommandozeile sehr gut geeignet, jede Form von Parametern zu übergeben: Dateinamen, sonstige verschlüsselte oder nicht verschlüsselte Anweisungen wie beispielsweise bei PIP.Im Fall unseres Programms, in dem nur ein Dateinamen übergeben werden soll, wäre es noch praktischer, wenn dieser Name vom Betriebssystem gleich in der für den FCB benötigten Form aufbereitet wird: eine vorangestellte Laufwerksangabe (B:) auswerten, den Dateinamen bis zur Länge von 8 Zeichen mit Zwischenräumen auffüllen. Genau das tut das Betriebssystem: es bereitet nicht nur einen, sondern sogar bis zu zwei Dateinamen aus der Kommandozeile entsprechend auf.

Der erste Dateiname wird vom Betriebssystem in aufbereiteter Form an der Adresse 005CH abgelegt, der zweite (falls vorhanden) an der Adresse 006CH. Falls ein Anwenderprogramm nur einen Dateinamen benützt, kann es die Adresse 005CH ohne Änderung als FCB verwenden, da vom Betriebssystem auch die entsprechenden Bytes schon auf Null gesetzt werden. Werden zwei Dateinamen benutzt, muß der zweite bei der Adresse 006CH stehende zuerst an eine andere Stelle verschoben werden. Beim Öffnen der an der Adresse 005CH stehenden Datei würde dieser sonst überschrieben werden.

Zwei Beispiele sollen das zeigen:

```
A>xyz versuch1.tlc
```

ergibt als Dateisteuerblock bei 005CH bzw. Textzeile bei 0080H:

```
0050  FF FF 00 00 FF FF 00 00 FF FF 00 00 00 56 45 52 .............VER
0060  53 55 43 48 31 54 4C 43 00 00 80 01 00 20 20 20 SUCH1TLC.....
0070  20 20 20 20 20 20 20 20 00 00 00 00 01 00 00 00         ........

0080  0D 20 56 45 52 53 55 43 48 31 2E 54 4C 43 00 00 . VERSUCH1.TLC..
```

```
A>xyz datei1.txt b:datei2.txt
```

ergibt zwei vorbereitete Dateisteuerblöcke und eine längere Textzeile:

```
0050  FF FF 00 00 FF FF 00 00 FF FF 00 00 00 44 41 54 .............DAT
0060  45 49 31 20 20 54 58 54 00 00 80 01 02 44 41 54 EI1  TXT.....DAT
0070  45 49 32 20 20 54 58 54 00 00 00 00 01 00 00 00 EI2  TXT........

0080  18 20 44 41 54 45 49 31 2E 54 58 54 20 42 3A 44 . DATEI1.TXT B:D
0090  41 54 45 49 32 2E 54 58 54 00 00 00 00 00 00 00 ATEI2.TXT.......
```

Beachten Sie, wie im zweiten Beispiel die Laufwerksangabe beim zweiten Dateinamen ("B:") in das Byte 02 übersetzt wurde.

Systemfunktionen

Für die erweiterte Version des Programms zur Buchstabenwandlung werden noch zwei Systemfunktionen benötigt und zwar zum Löschen und zum Umbenennen von Dateien.

Funktion 19 - Datei Löschen (Delete File)

Parameter beim Aufruf: Register C: 13H
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0 bis 3, falls ok,
0FFH, falls Datei
nicht gefunden

Dem System wird die Adresse eines FCB übergeben, der den Namen der Datei enthält, die gelöscht werden soll. Nach einem erfolgreichen Aufruf (Datei gefunden und gelöscht) ist das Register A gleich 0, 1, 2 oder 3. Falls die im FCB angegebene Datei nicht gefunden wurde, steht 0FFH im Register A.

Bei der Löschfunktion darf der Dateiname auch Fragezeichen (ASCII 3FH) enthalten, um mehrere Dateien gleichzeitig zu löschen. Es werden dann alle Dateien gelöscht, die in den restlichen Stellen des Namens übereinstimmen. Die mit Fragezeichen angegebenen Positionen dürfen beliebig sein, nur in der Laufwerkbezeichnung darf kein Fragezeichen angegeben werden. Nachstehend ein Beispiel für den Aufruf der Funktion 19:

```
SYSTEM  EQU     0005H
DELETEF EQU     20
        LD      DE,FCB      ;FCB übergeben
        LD      C,DELETEF   ;Funktion wählen
        CALL    SYSTEM      ;und System aufrufen
        CP      0FFH        ;prüfen ob Fehler ...
        JR      Z, ...
```

Funktion 23 - Datei umbenennen (Rename File)

Parameter beim Aufruf: Register C: 17H
Registerpaar DE: Adresse FCB

Parameter beim Rücksprung: Register A: 0 bis 3 falls ok,
0FFH falls Datei
nicht gefunden

Dem System wird die Adresse eines FCB übergeben, in dem der Name der Datei steht, die umbenannt werden soll. In den zweiten 16 Bytes des FCB wird ein zweiter Dateiname angegeben, den die Datei erhalten soll. Nach einem erfolgreichen Aufruf (Datei gefunden und umbenannt) steht im Register A eine 0, 1, 2 oder 3. Wenn die erste, im FCB angegebene Adresse nicht gefunden wurde, steht 0FFH im Register A.

Das Laufwerk, auf dem die Umbenennung geschehen soll, wird mit dem ersten Namen angegeben, beim zweiten Namen soll Null als Laufwerkangabe stehen. Vor dem Umbenennen muß sichergestellt sein, daß der neue Dateiname auf dem Laufwerk noch nicht vorhanden ist. Ein Beispiel für den Aufruf der Funktion 23:

```
SYSTEM  EQU   0005H
RENAMF  EQU   20

FCB:    DB    2,'PROGRAMMTLC'   ;"B:PROGRAMM.TLC" in
        DS    4,0               ;(bis Byte 15 auffüllen)
        DB    0,'PROGRAMMBAK'   ;"PROGRAMM.BAK" umbenen-
                                ;nen
        DS    5,0               ;(bis Byte 32 auffüllen)

        LD    DE,FCB            ;FCB übergeben
        LD    C,RENAMF          ;Funktion wählen
        CALL  SYSTEM            ;und System aufrufen
        CP    0FFH              ;prüfen, ob Fehler ...
        JR    Z, ...
```

Die wichtigsten Systemfunktionen für die Diskettenoperationen kennen Sie jetzt. Die Funktionen zum wahlfreien Lesen und Schreiben (Random Read, Random Write) wurden nicht behandelt. Bei diesen Funktionen wird in den Feldern r0 und r1 des FCB zusätzlich die Nummer des gewünschten Datensatzes angegeben. Bei den normalen Lese- und Schreib-Funktionen werden die Datensätze einer nach dem anderen bearbeitet, während z.B. beim wahlfreien Lesen etwa der letzte Datensatz einer langen Datei gelesen werden kann, ohne vorher die ganze Datei zu lesen.

Die genaue Anwendung dieser Funktionen können Sie anhand der Erfahrungen mit den bisher verwendeten Funktionen sicher leicht dem CP/M-Handbuch entnehmen. Die dort stehenden ausführlichen Daten sind in unserem beschreibenden Lehrgang nicht enthalten.