Schieb · Weiler

COMMODORE 128

Das CP/M-Buch

EIN DATA BECKER BUCH

DAS STEHT DRIN:

Endlich CP/M beherrschen! Von grundsätzlichen Erklärungen zur Speicherung von Zahlen, Schreibschutz oder ASCII über Anwendung von CP/M-Hilfsprogrammen bis zu CP/M intern für Fortgeschrittene findet hier jeder Commodore-128-Anwender schnell die notwendigen Hilfen und Informationen zur Arbeit mit CP/M.

Aus dem Inhalt:

- Die Aufgabe von CP/M
- Die System-Diskette
- Regeln für Dateinamen
- Eingebaute Befehle
  USER, DIR, ERASE
- Transiente Befehle
  SET, PROTECT, SHOW, SUBMIT
- Alles über PIP
- Mehrere Dateien hintereinander drucken
- Alle 128er-spezifischen CP/M-Befehle
- Kommentiertes Z80-ROM-Listing

UND GESCHRIEBEN HABEN DIESES BUCH:

Jörg Schieb ist erfahrener Programmierer von Dateiprogrammen, gehört zum Autoren-Team des C128 Intern und kennt die Maschinenspracheprogrammierung des Commodore 128 in- und auswendig.
Elmar A. Weiler ist freier Journalist und Autor der DATA BECKER Bücher zu WORDSTAR und dBASE II.

ISBN 3-89011-116-5

Schieb · Weiler

COMMODORE 128

Das CP/M-Buch

EIN DATA BECKER BUCH

ISBN 3-89011-116-5

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

INHALTSVERZEICHNIS

Vorwort

Der Commodore 128, das ist wohl klar, ist ein Supercomputer. Er verfügt über drei total unabhängige Modi, die Sie sicherlich schon kennen: Den 64er-Modus, den 128er-Modus und den CP/M-Modus. Jeder für sich ist sehr interessant und bedarf genauerer Betrachtung. Vorgänger des Commodore 128 ist der Commodore C-64, für den es einen Berg an Software gibt, die dank der 64er-Kompatibilität im Commodore 128 ohne Probleme zum Laufen kommt. Für den zweiten, den 128er-Modus, muß erst einmal Software erstellt werden, die aber sicherlich nicht mehr lange auf sich warten lassen wird. Nun gibt es noch den CP/M-Modus, der nicht so sehr beachtet wird, dennoch aber sehr wichtig ist.

Auch für den Commodore 64, so haben Sie vielleicht gehört oder irgendwo gelesen, gibt es ein CP/M-Modul. Kein Mensch redet aber davon. Ist CP/M deswegen Käse? Ganz und garnicht, lieber Leser. Das CP/M für den Commodore 64 ist ein mühsam aus dem Boden gestampftes CP/M 2.2, mit katastrophalen Ausführungszeiten. Die 1541 tut ihren Teil dazu.

Der Commodore 128 mit seinen 128 KBytes RAM ermöglicht es, CP/M 3.0 (auch CP/M Plus genannt) zu "fahren". Dieses CP/M 3.0 ist ungleich leistungsstärker als CP/M 2.2. Der Commodore mit seiner fantastischen Tastatur eignet sich hier besonders für längeres Arbeiten. Ein gutes Konzept erkennt man auch daran, daß beispielsweise die Tasten **F** und **J** mit einem kleinen Punkt in der Mitte versehen sind, so daß man leichter seine zehn Finger auf der Tastatur plazieren kann. Commodore hat eben doch etwas mehr Erfahrung als einige "Newcomer", die zwar eine Maus zu bieten haben, dafür aber eine miserable Tastatur, die zur Eingabe langer Texte ebenso wenig einlädt, wie ein schmutziges Lokal zum dinnieren.

Die 1571 sowie die etwas preiswertere 1570 haben ihre Ladegeschwindigkeit gegenüber der 1541 erheblich verbessert. Unter uns: ein Arbeiten mit CP/M 3.0 und der 1541 ist zwar möglich, aber eine Zumutung - es sei also davon abgeraten.

Wer CP/M wegen des sich breit machenden GEM totsagt, der muß sich wohl noch ein paar Jahr gedulden. Unter CP/M existiert nun mal eine Masse an Software, die auf diese Weise für einen neuen Rechner - mit CP/M ausgestattet - schon sehr bald zur Verfügung steht. Die 68.000er-Rechner tun sich hier zumindest anfangs noch recht schwer, da Software *komplett* neu entwickelt werden muß.

Wir wollen Ihnen mit diesem Buch das Arbeiten mit CP/M nahe bringen, so daß es Ihnen ein Vergnügen sein wird, die CP/M-Diskette einzulegen und zu "booten". Begriffe wie BIOS, TPA, BDOS und CCP gehören dann nicht mehr der Welt unanständiger Begriffe an, ein PIP vernimmt man dann nicht mehr ausschließlich aus dem Schnabel Ihres Kanarienvogels. Alle gängigen CP/M-Kommandos sowie Spezialitäten des Commdore-CP/Ms werden erläutert, auf den Commodore 128 speziell zugeschnitten.

Wir wünschen Ihnen viel Freude mit dem CP/M-Buch für den Commodore 128 und viel Erfolg bei der Arbeit.

Düsseldorf, im November 1985

E.A. Weiler und J. Schieb

I. Der Computer

Um ehrlich zu sein: Ich beneide Sie, weil Sie schon wissen, was Sie nicht wissen. Ich dagegen kann nur versuchen mir vorzustellen, wie weit Ihr Wissensstand in Beziehung auf Computer und alles, was dazugehört, gediehen ist. Da ich nun leider nicht jeden einzelnen fragen kann, fange ich in diesem Kapitel sozusagen bei "Adam und Eva" an, damit wir alle eine gleiche Ausgangsbasis haben und uns nachher richtig verstehen. Solche Vorgehensweisen gibt's überall, sehen Sie sich nur die DIN-Normen an oder versuchen Sie einmal, ein Programm für Ihren Computer zu schreiben. Immer greifen die Menschen auf festgelegte Vereinbarungen zurück, um sich richtig zu verstehen.

Also los. Erst einmal, was ist eigentlich ein Computer? Getreu einer hochwissenschaftlichen Methode wollen wir nun diese Frage beantworten.

Dazu stellen wir uns erst einmal ganz dumm und sagen: "Ein Computer ist ein Kasten, in dem etwas passiert". Glücklicherweise weist der englische Name "Computer" schon darauf hin, was da passiert. "Compute" läßt sich mit "rechnen" übersetzen und der Computer ist demnach ein Rechner. Solch einen Zahlenjongleur kann man hervorragend einsetzen, um große Zahlenmengen in der Buchhaltung oder bei der Auswertung von Meßreihen abarbeiten zu lassen. Schüler und Studenten finden sicherlich andere praktische Anwendungen eines elektronischen Rechenknechtes.

I.1 Die Tastatur

Aber - da ist ein Problem. Wir haben nur einen Kasten vor uns, der rechnen kann. Wie aber geben wir ihm ein, was er rechnen soll? Eine Möglichkeit wäre, ihm etwas ins Ohr zu flüstern. Aber diese Art der Verständigung mit Computern ist zur Zeit noch Sache der Science-Fiction und für uns nicht brauchbar. Deshalb entsinnen wir uns der guten alten Tastatur und schließen ein solches "Buchstaben- und Zeichen-Eingabe-Gerät" an den Kasten an.

Da der Computer aber sehr viel mehr "drauf" hat als eine Schreibmaschine, muß auch die Tastatur etwas anders aussehen. Zwischen 60 und über 100 Tasten, je nach Preislage und Komfort, bietet eine Computer-Tastatur. Ihr Commodore 128 verfügt über 92 Tasten, die verschiedenen Funktionsblöcke sind sauber voneinander getrennt. Wie es sich für einen komfortablen Computer gehört, verfügt auch der Commodore 128 über eine leider überhaupt nicht selbstverständliche Zehnertastatur, beispielsweise zur Eingabe von Zahlenkolonnen. Der kleine Punkt in der Mitte der 5-Taste ist kein Fehler beim Ausstanzen, sondern dient der Orientierung Ihrer Finger bei der Benutzung dieser Zehnertastatur während einer Blindeingabe. Übrigens finden Sie diesen kleinen Punkt ebenfalls auf den beiden Tasten "F" und "J"; *Blindtipper* wissen warum: Auf diesen beiden Tasten müssen die beiden Zeigefinger liegen.

Ob ein Computer eine deutsche oder eine amerikanische Tastatur besitzt, können Sie leicht mit einem Blick feststellen. Steht in der obersten Buchstaben-Reihe von links das Wort QWERTZ, haben Sie einen Rechner mit einer Eingabeeinheit nach deutschem Geschmack. Finden Sie dort QWERTY, haben Sie es unter Umständen mit vielen Computer-Programmen leichter, weil die zur Zeit meist aus Amerika

kommen, müssen dafür aber auf die hübschen deutschen Buchstaben "ÖÄÜß" verzichten. Man kann eben nicht alles haben. Als Commodore-128-Besitzer sind Sie jetzt sicherlich verwirrt, denn Sie verfügen über beide "Wörter", auch wenn der Buchstabe Z nur in hellgrau auf der Tastatur aufgedruckt ist. Wir behaupten nun: man kann *doch* alles haben, wie man sieht. Sie können durch die CAPS-LOCK-Taste (vierte von links in der ersten Tastenreihe) auswählen, ob Sie eine amerikanische Tastatur (ASCII-Tastatur) wünschen, dann muß die Taste losgelassen sein, oder ob Sie eine deutsche Tastatur bevorzugen - die CAPS-LOCK-Taste ist dann gedrückt.

Entsprechend Ihrer Wahl ändert sich auch der Zeichensatz auf dem Bildschirm. Bei der deutschen Tastatur gelten, sofern vorhanden, die hellgrauen Aufdrucke auf der Tastatur Ihres Commodore, so daß Ihnen auch Umlaute am gewohnten Platz zur Verfügung stehen. Allerdings wird im CP/M-Modus immer nur mit der amerikanischen Tastatur gearbeitet - aus Kompatibilitätsgründen. Lediglich den Zeichensatz können Sie auf diese Weise wählen, bevor Sie CP/M laden.

Ob deutsch oder englisch, es gibt ein paar Tasten, die für die Computerei sehr wichtig sind und oft auf beiden Tastatur-Arten gleich oder ähnlich beschriftet sind. Weil diese Ausnahmen für den Rest des Buches und bei allen Programmen wichtig sind, erfahren Sie jetzt, um welche Tasten es sich dabei handelt. Dabei sollen die meistverwendeten Bezeichnungen für diese Tasten genannt werden, damit Sie vor einem anderen Rechner als dem Commodore sitzend, nicht auf dem Schlauch stehen. Die für den Commodore-Rechner zutreffende Bezeichnung ist jeweils unterstrichen:

WAGENRÜCKLAUF

Sie kennen diese Taste von der Schreibmaschine her, verwenden sie aber beim Computer meist in einem völlig anderen Sinne. Die Abkürzungen auf dieser Taste sehen meist so aus:

RETURN (engl. für "Wagenrücklauf")

ENTER (Abschluß einer Eingabe)

CR ("Carriage return" = engl. Wagenrücklauf)

<--- (erklärt sich von selbst)

RÜCKWÄRTSSCHRITT

BS (kommt von "Back Step")

<- (Cursor rechts/links oder Cursor-links)

LÖSCH- ODER KORREKTURTASTE

DELETE (engl. für "löschen")

DEL (Abkürzung für DELETE)

RUB OUT (engl. für "ausradieren")

UMSCHALTUNG

SHIFT (schaltet auf Großbuchstaben um)

FESTSTELLER

SHIFT LOCK (schaltet dauerhaft auf Großbuchstaben um)

Die genaue Funktion dieser letzten Taste hängt von der Bauart Ihrer Tastatur ab. Teilweise wird mit dieser Taste nicht die gesamte Tastatur umgeschaltet, sondern wirklich nur auf die Großbuchstaben. Das kann eine erhebliche Arbeitserleichterung sein. Beim Commodore 128 allerdings ist die SHIFT-LOCK-Taste für alle Tasten von Bedeutung und mit dem Niederhalten der SHIFT-Taste gleichbedeutend. Auf einigen Tastaturen gibt's für die Umschaltung der Buchstaben noch eigene Tasten, die mit

ALPHA LOCK oder

CAPS LOCK

benannt sind. Leider wird beim Commodore 128 der Zustand der SHIFT-LOCK-Taste nicht angezeigt, dafür rastet sie aber ein, und dies kann man schon wahrnehmen.

Ganz wichtig beim Computern ist die Taste CONTROL, links außen auf Ihrer Commodore-Tastatur, die für viele Steuerbefehle in Verbindung mit einer Buchstaben- oder Zifferntaste gedrückt werden muß. Sie ist in der Regel mit:

CTRL oder

CONTROL

beschriftet.

I.2 Der Bildschirm

Richtig, eigentlich müßte die Überschrift für den Commodore 128 hier im Plural lauten: die Bildschirme, denn man kann ja bekanntlich zwei Bildschirme gleichzeitig anschließen. Dennoch: nachdem wir nun die Möglichkeit haben, Daten in den Kasten einzugeben, brauchen wir natürlich auch einen Weg, auf dem die Daten wieder zu uns gelangen können. Es nützt uns ja herzlich wenig, wenn der Rechner die tollsten Berechnungen ausführt, uns aber nicht mitteilen kann, was er herausbekommen hat. Recht umweltfreundlich funktioniert die Datenausgabe über einen Bildschirm, weil es erstens schnell geht und zweitens keinen Lärm macht. Daten und Fehlermeldungen des Computers werden schnell sichtbar, und wenn man sich seine Fehler aufschreiben möchte, kann immer noch ein Drucker in Aktion treten.

Ein Bildschirm ist im Grunde ein "umgebauter" Fernseher, der nach dem gleichen Prinzip funktioniert. Nur werden auf dem "Computer-Fernseher", häufig auch Monitor genannt, meist keine laufenden Bilder dargestellt, sondern Buchstaben, Ziffern und sonstige Zeichen. Deshalb sind die Anforderungen an einen Monitor auch andere als an einen Fernseher. Die Auflösung sollte deutlich besser sein, damit man bei den meist stundenlangen Sitzungen keine "Mattscheibe" bekommt und die Bindehaut des Auges nicht über die Maßen strapaziert wird.

I.3 Der Drucker

In der frühen Zeit der Groß-Computer mit heute winzigen Leistungen dienten Fernschreiber als Ausgabemedium für Daten in dauerhafter Form. Der Computer schickt also seine Daten

zum Drucker und verbraucht dort in der Regel eine große Menge Papier. Weiterer Nachteil neben dem großen Papierhunger ist die (zumeist) lautstarke Art, wie Drucker ihre Zeichen zu Papier bringen. Aber - zur Zeit bestehen noch viele Leute auf Informationen in geschriebener Form und man kommt um einen Drucker nicht herum. Weil es so viele verschiedene gibt, hier ein kurzer Überblick:

Preise und Leistungen schwanken in einer Breite, die selbst von Spezialisten kaum noch zu übersehen ist. So kosten Drucker zwischen 400 und 10 000 Mark und bieten dafür Schreibgeschwindigkeiten zwischen 15 und 800 Zeichen pro Sekunde (und mehr).

I.3.1 Nadeldrucker

Besonders gut eingeführt als Mitarbeiter von Micros sind die sogenannten Nadeldrucker. Sie haben im Markt die größte Bedeutung und aus ihren Reihen kommen sowohl die preisgünstigsten wie auch die schnellsten Exemplare. Begonnen hat alles mit Nadel-Druckern, die fein säuberlich Punkt neben Punkt setzten und damit eine Schrift produzierten, die alles andere als säuberlich aussah. Das liegt am Druck-Prinzip der "frühen" Jahre (etwa 1980). Einzeln steuerbare Nadeln drücken auf ein Farbband und dann auf's Papier, so daß hier ein kleiner Punkt abgebildet wird. Aus sieben Punkten läßt sich schon etwa ein Buchstabe oder eine Zahl erkennen. Nur fällt das Lesen längerer Texte recht schwer.

Aber - diese Zeiten sind vorbei. Außer bei ganz billigen Geräten bieten Nadeldrucker heute eine Schreibbreite von 80 Zeichen, etwa 100 Zeichen pro Sekunde an Geschwindigkeit und ein Schriftbild, das zwar nicht für Korrespondenz, aber für nebensächlichen Schriftverkehr vertretbar ist. Großer Vor-

teil der Nadel-Drucker: Sie bieten in der Regel eine Vielzahl von Schriften und auch frei lad- oder programmierbare Zeichensätze an, die einfach umgeschaltet werden können. So wird Schriftverkehr mit Griechenland oder auch Japan sehr erleichtert.

Hochleistungs-Drucker erreichen Geschwindigkeiten bis zu 800 Zeichen pro Sekunde bei einer Schreibbreite von 132 Zeichen, kosten allerdings zwischen 8000 und 10 000 Mark. Relativ neu sind die Nadeldrucker der "neuen Art".
Sie besitzen meist 24 Drucknadeln und können damit, bei verringerter Geschwindigkeit, Schreibmaschinen-Qualität erreichen. Übliche Werte sind hier rund 70 Zeichen pro Sekunde für Korrespondenz und 140 bis 150 Zeichen im Schnellschreib-Modus.

I.3.2 Typenraddrucker

Zur zweiten Drucker-Kategorie gehören die Typenrad-Drucker. Mit hierzu zählen noch die Typenkorb und Kugelkopf-Drucker. Typenkorb und Kugelkopf-Maschinen basieren meist auf früheren Schreibmaschinen-Konstruktionen und bedürfen in der Regel eines hohen Wartungsaufwandes. Zudem ist die Herstellung recht teuer. Typenrad-Drucker ersetzten den Typenkorb oder den Kugelkopf durch eine runde Scheibe, auf der die verschiedenen Buchstaben und Zeichen federnd angebracht sind. Ein kleiner Hammer schlägt die Type auf's Papier, ein Unterschied zu einer Schreibmaschine ist nicht festzustellen. Wenn also die Korrespondenz "wie gedruckt" aussehen soll, kommt ein Typenrad-Drucker in Frage.

Geschwindigkeit: bei den billigen Modellen meist gerade 20 Zeichen pro Sekunde, bei Spitzenmodellen 80 Zeichen.

I.3.3 Tintenstrahldrucker

Deutlich billiger geben sich die neuartigen Tintenstrahl-Drucker. Wie der Nadeldrucker erstellen sie Buchstaben und Zeichen aus einzelnen Punkten. Dabei findet aber keine mechanische Berührung zwischen Druck-Kopf und Papier statt, sondern es werden winzige Farbtröpfchen auf's Papier geschossen. Die Qualität des Schriftbildes ist mit der von einfachen Nadeldruckern vergleichbar. Besondere Vorteile der Tintenstrahl-Drucker: Sehr geringer Geräuschpegel bei etwa 45 dbA (Dezibel A), trotzdem 150 Zeichen pro Sekunde. Nachteil: Kopien sind nur über mehrmaliges Drucken möglich.

I.3.4 Thermodrucker

Den gleichen Nachteil bieten die sogenannten Thermodrucker. Sie erzeugen die Zeichen durch Wärmebeeinflussung der Papieroberfläche. Selbstverständlich ist hierzu ein besonderes Papier notwendig, das mit der Zeit im Licht vergilbt. Dafür sind die Drucker, die auch im Pünktchen-Verfahren arbeiten, extrem preiswert und für Protokoll-Funktionen durchaus noch zu gebrauchen. In Taschenrechnern unterhalb der 100-Mark-Preisgrenze sind die einfachen Thermodrucker heute schon im Einsatz.

Das Optimum für den schnellen, sauberen Druck per Personal-Computer sind die Laser-Drucker. Deren Ausstoß ist von einer professionell gesetzten und gedruckten Arbeit nicht zu unterscheiden.

Ach ja, da fällt mir etwas ein. Unser Kasten, der mittlerweile in der Beschreibung einem Computer schon recht ähnlich ist, sollte natürlich auch einen Schalter haben, um

die Stromzufuhr unterbrechen zu können. In der an sich guten Absicht, versehentliches Ein- oder Auschalten zu erschweren, kamen viele Computerhersteller auf den Dreh, daß ein Einschaltknopf irgendwo versteckt sein muß. Wenn Sie an einem Computer den Schalter noch nicht gefunden haben, suchen Sie an einer Stelle, wo ihn kein Mensch vermuten würde. Er ist bestimmt da.

I.4 Datenspeicher

Wir können mit unserem theoretischen Computer nun schon ziemlich viel machen. Der Computer selbst ist da, die Tastatur, ein Bildschirm, und einen Drucker haben wir auch schon. Was noch fehlt, ist ein Platz, an dem die Daten gespeichert werden können. Einmal innerhalb des Computers und - für längere Zeit - auch außerhalb. Grundsätzlich kann man sagen, ist die Art Daten zu speichern in jedem Medium des Computers gleich. Ob im sogenannten Arbeitsspeicher, das ist der, der jetzt gerade meinen hier geschriebenen Text aufbewahrt, oder auf einer Diskette oder einer Hard-Disk. Wieso und wie der Computer etwas speichern kann, sollen Sie jetzt erfahren.

Weil das ein wenig theoretisch ist, holen Sie sich vielleicht ein Täschen Kaffee, nehmen einen guten Schluck und lesen dann aufmerksam weiter. Ich warte, bis Sie wieder da sind...

I.5 Eins oder Null - Grundlagen der Binärarithmetik

Sie werden nicht abstreiten, daß ein Computer ein elektrisches Gerät ist. Wenn dem so ist, dann müssen auch die Informationen innerhalb des Computers in einer elektrischen Form verarbeitet werden. Aber, wie stellen wir das an? Wie sollen wir einem Computer sagen, was wir meinen? Dazu ist eine Art von Darstellung notwendig, die der Computer versteht. Es gibt eine Art, die der Computer kennt - die beiden Zustände: Strom fließt - kein Strom fließt. Nach dem gleichen Prinzip funktioniert auch eine Glühbirne. Bemerkt sie, daß der Strom fließt, fängt sie schleunigst zu brennen an. Fällt ihr auf, daß kein Strom mehr da ist, geht sie einfach aus. Was so eine dumme Glühbirne kann, kann unser Computer schon lange. Seine Erbauer haben ihm in sein Elektronen-Gehirn geschrieben, daß der Zustand "Strom fließt" gleichbedeutend ist mit dem Wert "Eins". So wie für uns die "1" auch einen bestimmten Wert hat.

Fließt kein Strom, bedeutet das für ihn "0". Damit kann er jetzt zwei Zustände unterscheiden, was zu erstaunlichen Ergebnissen führt.

I.6 Binär zählen

Für den täglichen Gebrauch ist die binäre Zählweise, also mit eins und null, nicht sonderlich gut geeignet. Bis Sie auf diese Weise Ihr Geburtsdatum hergesagt haben, bekommen Sie schon eine trockene Zunge. Glücklicherweise hat der Computer keine Zunge und so kann auch nichts trocken werden, wenn er wie ein Wahnsinniger mit den beiden Ziffern hantiert.

Sehen wir uns das Arbeiten mit Zahlen einmal grundsätzlich an. Es gibt verschiedene Zahlensysteme: das binäre, das dezimale, das hexadezimale und noch einige mehr. Das dezimale System beherrschen wir alle aus dem Eff-eff. Wir können kaum anders denken als in Zehnerwerten. Was machen wir aber dabei? Wir denken uns eine Zahl, zum Beispiel die Null. Soll es mehr sein, nehmen wir die Eins und so weiter. Sind wir bei der "9" angelangt, setzen wir die nächste Zahl aus den zwei niedrigsten wieder zusammen, was eine "10" ergibt.

Genau so können wir auch vorgehen, wenn wir nur zwei Ziffern zur Verfügung haben. Der erste Wert ist die Null. Der nächste die Eins. Dann ist Schluß, wir haben keine Zahlen mehr um weiterzuzählen. Also setzten wir die nächste Zahl aus den beiden zusammen und erhalten für den Wert Zwei die Ziffer 10. Die Drei sieht so aus: 11 und für die Vier brauchen wir schon wieder eine neue Stelle, was 100 ergibt.

Zum Verständnis hier nochmal eine Tabelle der binären Zahlen:

```
Null   = 0
Eins   = 1
Zwei   = 10
Drei   = 11
Vier   = 100
Fünf   = 101
Sechs  = 110
Sieben = 111
Acht   = 1000
```

und immer lustig so weiter.

Achtet man nur auf die Stellen dieses Zahlensystems, kann man mit einer Stelle zwei Werte darstellen, mit zwei Werten vier Stellen, mit drei Stellen acht, mit vier Stellen 16 und so weiter. Von Stelle zu Stelle ergibt sich immer die doppelte Möglichkeit. In unserem Dezimal-System kennen wir die Wertigkeit der Stellen ja auch. Nur geht's hier immer gleich um Zehner. Eine Stelle kann 10 Werte darstellen, zwei Stellen hundert, drei Stellen tausend...

Die Geschichte mit den zwei Zahlen nennt sich binäres (zweizahliges) Zahlensystem und hat den Leuten, die uns nicht nur den Hamburger, sondern auch die Computer schmackhaft machten, so gut gefallen, daß sie sich gleich ein paar Fachausdrücke dafür ausdachten.

Eine Stelle einer Zahl heißt im amerikanischen "digit" und zu "binär" sagt die Super-Nation "binary". Beides zusammen nennt sich "binary digit" und wurde kurzerhand auf "bit" abgekürzt.

Nun ist aber eine einzige Stelle als Speicherplatz recht wenig und moderne Computer haben eine ganze Menge Speicherplätze. Läßt man es bei einem Bit, werden die Zahlen für Speicherplatzangaben enorm groß. Aus diesem Grunde hat man Einheiten, "Wörter", geschaffen, die mit einem Wort oder einer Zahl eine ganze Reihe von Bits beschreiben. Mit einem Acht-Bit-Wort kann man so 256 Bits beschreiben und mit einem 16-Bit-Wort 65536.

Sehr häufig kommt die Einheit mit 256 Möglichkeiten vor, weshalb diese Einheit den Namen "Byte" trägt. Es ist allerdings ein Gerücht, daß die amerikanische Fachzeitschrift mit dem Namen Byte aus diesem Grunde nur mit maximal 256 Seiten erscheinen darf.

I.7 Werte speichern

Um richtig arbeiten zu können, muß der Computer die Zahlen auch speichern können. Eine Zahl, wie wir gesehen haben, besteht für den Computer aus einem elektrischen Zustand: geladen oder ungeladen. Faßt man acht solcher Speicherplätze zusammen, erhält man ein Acht-Bit-Wort, das insgesamt 256 Werte darstellen kann. "Zufälligerweise" entspricht das genau einem Byte. Weil aber auch ein Byte noch nicht sehr viel ist, gibt's noch die Bezeichnung Kilobyte. Dabei entspricht aber ein Kilo nicht genau 1000 Bit sondern, entsprechend der binären Zählweise 1024 Bits. Für ganz große Einheiten gibt's dann noch das Megabyte.

Mit Hilfe elektronischer Schaltungen ist es nun möglich, die Werte aus einem Byte herauszulesen, weiter zu verarbeiten und auch neue Werte wieder einzulesen. Das ist das ganze Geheimnis des Computers - wenngleich dieses Geheimnis uns auch sehr viel Arbeit abnehmen kann. Sein Vorteil ist, daß er dieses Operationen in sehr kurzen Zeitabständen machen kann und deshalb ein echter und nützlicher Datenverarbeiter ist.

Da jede Speicherstelle wirklich vorhanden sein muß, hat jeder Computerspeicher nur einen begrenzten Raum, in dem Daten abgelegt werden können. Übliche Größen für den Arbeitsspeicher eines Personal-Computers mit einer 8-Bit-Zentraleinheit, das ist das eigentliche Hirn, sind 64 Kilobyte oder kurz 64 KB. Die neuen 16-Bit-Computer bieten Speichergrößen zwischen 128 KB und rund acht Megabyte. Der Commodore 128 ist zwar auch ein 8-Bit-Computer, bietet aber 128 KBytes - sviel sind für CP/M 3.0 ja auch notwendig. Allerdings kann man ihn bis auf 1 MByte ausbauen, dies ist zumindest im Betriebssystem so vorgesehen.

Etwas unklar ist bis jetzt aber noch, wie man zum Beispiel Texte in einem Computer speichern kann, wo der doch nur Zahlen hin- und herschicken können soll.
In diesem Punkt sind Computer ganz menschlich. Sie benutzen, genau wie wir, wenn wir eine Fremdsprache übersetzen, ein Wörterbuch. Innerhalb des Computers gibt's eine Tabelle, in der für jeden Buchstaben und all die anderen Zeichen ein Zahlenwert festgelegt ist. Kommt nun ein Buchstabe in den Computer rein, schaut der in der Tabelle nach und findet den richtigen binären Zahlenwert. Den wiederum kann er problemlos verarbeiten. Bei der Ausgabe funktioniert die Sache genau umgekehrt.

Damit das sinnvoll nicht nur auf einem Computer eines bestimmten Herstellers klappt, muß man sich über die "Übersetzungstabelle" einigen. Leider, aus deutscher Sicht, geschah diese Einigung in Amerika und umfaßt auch nur die dort gebräuchlichen Zeichen. Das Ganze nennt sich ASCII (American Standard Code for Information Interchange = amerikanischer Standardcode für Informations-Austausch) und beinhaltet eben die deutschen Sonderzeichen nicht. In diesem ASCII-Code sind 128 Zeichen erfaßt, von denen jedes seinen eigenen Wert hat. Im Anhang finden Sie eine Tabelle mit all diesen Werten.

Allerdings hat man jetzt auch deutsche und andere ausländische Umlaute vorgesehem, diese allerdings an einen Platz weit "hinten" verbannt. Auf diese Weise haben die amerikanischen Hersteller der Geräte am wenigsten Nachteile dadurch und dennoch ein Pro-Argument für die Europäer zur Hand.

Was aber macht der Computer mit einem Speicher voller Werte? Nichts. Er ist zu passiv, irgendetwas mit diesen Daten anzufangen. Damit etwas passiert, müssen Sie ihm erst einen entsprechenden Befehl geben. Sie können ihm befehlen, den

Text in seinem Speicher mit einem glatten rechten Rand zu versehen oder die Beleuchtung anzuschalten. Erst wenn sie sinnvolle Befehle geben, kann der Computer auch etwas ausrichten. Auf der untersten Ebene sieht das so aus: Sie sagen dem Computer: Nimm das Bit im Speicher "a", addiere es zu dem Bit in Speicher "b" und lege das Ergebnis in Speicher "c" ab.

Eine ganze Reihe von solchen Anweisungen sind bereits in der Central Processing Unit (CPU), dem Rechengehirn des Computers, gespeichert. Aus vielen dieser Operationen lassen sich auch komplizierte Vorgänge zusammenstellen, was die Aufgabe der Maschinen- oder Assembler-Programmierung ist. Das sind Programmier-Sprachen, die die Maschinenfunktionen direkt steuern. BASIC, PASCAL oder FORTRAN steuern die CPU erst nach einer internen Übersetzung des jeweiligen Programms in die Maschinensprache.

Übrigens: Deutschfetischisten akzeptieren die Abkürzung *CPU* nicht - nein, es muß bei den Datenverarbeitungskaufleuten *ZE* für *Zentraleinheit* heißen.

Um eine bestimmte Aufgabe zu lösen, muß dem Computer jeder einzelne Schritt haarklein vorgegeben werden. Das geschieht in der Regel mit Hilfe eines Programms, was nichts anderes ist, als die Aneinanderreihung von Befehlen, die nacheinander abgearbeitet werden. Das Programm selbst muß natürlich auch im Speicher vorhanden sein, damit es ausgeführt werden kann, verbraucht also auch Speicherplatz.

Je nach Anforderungen, kann solch ein Programm zwischen zehn und 120 KByte verbrauchen. Unter Umständen ist der Platz im Speicher allein schon durch das Programm so belegt, daß für die Ausführung des Programms und die Daten des Benutzers kein Platz mehr bleibt. Das bedeutet dann: Dieses Programm ist für diesen Computer zu groß und kann nicht benutzt

werden. Aber diese Fälle sind recht selten. Häufiger kommt es vor, daß die Daten nicht mehr in den eingebauten Speicher passen.

I.8 Massenspeicher

In den seltensten Fällen ist es nötig, ein ganzes Programm auf einmal im Speicher zu haben. Meist genügt es völlig, wenn die wichtigen Teile im Speicher stehen und der Rest nur bei Bedarf nachgeladen wird. Ein auf diese Weise arbeitendes Programm arbeitet in der sogenannten *Overlaytechnik*. Aus diesem Grunde gibt es bei Computern außer dem internen Arbeitsspeicher auch externe Speichermöglichkeiten, die nicht einer so großen Platzbeschränkung unterliegen und außerdem billiger sind. Die sogenannten Massenspeicher können sein: ein Kassettenlaufwerk, eine Floppy-Disk-Station oder eine Festplatte. Alle drei haben ihre Vor- und Nachteile, bieten aber alle reichlich Platz für Daten und können diese in mehr oder weniger erträglicher Zeit mit dem Computer austauschen.

I.8.1 Floppy-Disk

Die Floppy-Disk, auch Floppy oder nur Disk genannt, ist eine Art Schallplatte aus dem Material eines Tonbandes. Ein dünne Scheibe mit magnetisierbarer Oberfläche, eingehüllt in einen Schutzumschlag, gegen die fettigen Finger des Benutzers oder auch Staub- und Schmutzteilchen. Auf der Oberfläche der Floppy fährt, wenn sie im Computer steckt, ein Schreib-Lesekopf entlang und macht verschiedene Stellen magnetisch oder nicht.

Da haben wir wieder die zwei Zustände: magnetisch oder nicht. Das deckt sich mit der Speichermethode im Computer - und das mit Absicht. Die Daten im Computer, die dort als elektrische Zustände vorliegen, lassen sich auf diese Weise auf die Floppy übertragen. Hat ein Bit den Zustand geladen, macht der Schreibkopf der Floppy einen kleinen Punkt der Diskettenoberfläche durch einen Stromstoß magnetisch. Hat das nächste Bit keine Ladung, kommt auch nichts auf die Floppy. Aus der Folge von magnetischen und nicht magnetischen Punkten auf der Floppy können nun ein Programm oder eine Reihe von Daten gelesen und wieder in den Computer übertragen werden. Damit haben wir die Möglichkeit, Daten extern zu lagern.

Die Diskette dreht sich mit 200 bis 300 Umdrehungen pro Minute und würde ein fürchterliches Datenchaos hinterlassen, wenn man die Daten einfach planlos draufschriebe. Also unterteilt man die Diskette in eine bestimmte Anzahl von Spuren, die nebeneinander auf der Diskette liegen wie die Fahrbahnen einer mehrspurigen Autobahn.

Will man Daten ablegen, fährt der Schreibkopf über die entsprechende Spur und schreibt munter drauf los. Nun ist aber eine solche Spur immer noch recht lang und man müßte ganz schön lange warten, bis man seine Daten von solch einer Spur wieder in den Computer gelesen hat.

Um diese Zeit zu verkürzen, unterteilt man die Diskette nochmal in Sektoren (je nach Hersteller zwischen 128 Bytes und 1024 Bytes groß, der Commodore 128 verwendet im CP/M-Modus verschiedene Formate, da er zu mehreren Formaten kompatibel ist; welche Formate dies sind, dazu später mehr) was dann aussieht wie die Stückchen einer Torte.

Diese Methode der Aufzeichnung ist recht praktisch, verhindert aber unter anderem, daß man einfach eine Diskette des

einen Rechners in das Laufwerk eines anderen Rechners reinstecken und die Daten lesen kann. Jeder Hersteller denkt sich nämlich da so seine eigene Methode aus, um uns Anwender zu ärgern. Große Spezialisten auf diesem Gebiet sind die Firmen Apple und Victor.

Wenn Sie neue, unformatierte Disketten kaufen, müssen Sie diese in der Regel erst einmal formatieren. Dazu gibt's bei jedem Computer ein spezielles Programm, das genau die Spuren und Sektoren auf die Diskette schreibt, die der Rechner braucht.

Wenn Sie sich gerne etwas ärgern wollen, formatieren Sie doch mal die Diskette mit Ihren wichtigsten Daten drauf. Sie werden staunen, wie wenig davon noch übrig bleibt. Deshalb geben Sie Acht und machen Sie einen Schreibschutz auf die Kerbe der Diskette. Die kleinen Aufkleber, die jeder Packung Disketten beiliegen, können Ihnen Kopf und Kragen retten. Bei den modernwerdenden 3-Zoll-Disketten wie der Schneider-Computer sie beispielsweise verwendet, verstellt man einen Plastikschieber, was es dem Computer dann unmöglich macht, auf diese Diskette zu schreiben.

Genauso klug ist es übrigens, immer von allen wichtigen Daten eine Sicherheits-Kopie (sogenanntes BACKUP) anzulegen und unerreichbar für Kinder, Hunde, Katzen, Feuer, Einbrecher, Schneestürmen, Erdbeben und sonstigem Ungemach aufzubewahren. Wenn Sie schon Ihre Diamanten nicht in den Safe legen - tun Sie's wenigstens mit Ihren Sicherheits-Kopien. Wie Sie mit Hilfe von CP/M Sicherheits- und andere Kopien machen können, erfahren Sie in einem folgenden Kapitel.

Zur Zeit befinden sich ganz verschiedene Disketten-Formate auf dem Markt. Teilweise findet man noch die altehrwürdige 8 Zoll-Diskette, die einmal den Siegeszug der elektronischen

Datenverarbeitung einläutete. Besonders weit verbreitet sind die 5 1/4 Zoll-Disketten, die normalerweise in Personal Computern verwendet werden. Das Format ist einigermaßen handlich und mittlerweile passen auch schon fast zwei Megabyte auf solch ein Scheibchen.

Besonders aus Japan dringen aber auch 3 Zoll und 3 1/2 Zoll-Disketten auf den Markt, die ein nochmal verbessertes Handling versprechen, weil der Schreib/Leseschlitz durch eine Metall-Lasche verschlossen ist, solange die Diskette nicht im Laufwerk steckt. Außerdem haben die 3er-Disketten eine feste Plastik-Hülle, so daß man sie nur noch mit Gewalt knicken kann.

I.8.2 Hard-Disk

In Computern, die hauptsächlich beruflich genutzt werden, findet man auch noch die Hard-Disk oder den Festplatten-Speicher. Diese Dinger sind in den Abmessungen einer 5 1/4 Zoll-Diskette gleich und nehmen auch den gleichen Raum in einem Computer-Gehäuse ein (Weil die Verkleinerungswelle vor nichts haltmacht, gibt's natürlich auch schon 3-1/2 Zoll-Festplattenlaufwerke). Innendrin aber drehen sich beschichtete Metallplatten mit gut 3000 Umdrehungen pro Minute und die Schreib/Leseköpfe fliegen im Tiefstflug über die Plattenoberfläche. Ein Staubkorn oder Rauchpartikelchen auf dieser Oberfläche hätte die gleiche Wirkung wie das Matterhorn für einen Jumbo, wenn er zu tief fliegt.

Deshalb kommen die Hard-Disks in einem luftdicht verschlossenen Gehäuse, und darin sollte man sie auch lassen. Als Standard-Fassungsvermögen gelten seit Beginn der 80er Jahre 10 MByte für eine Festplatte. Aber, die Preise sinken und die 20 MB-Platte hält langsam ihren Einzug in die Computer-Gehäuse.

Außer dem besonders hohen Fassungsvermögen bietet die Festplatte aber noch einen entscheidenen Vorteil: Sie ist bis zu 20 mal schneller im Datenzugriff, als eine Diskette.

So, wenn ich richtig mitgezählt habe, müßten jetzt alle Teile eines richtigen Computers besprochen sein und Sie sollten die grundlegenden Kenntnisse über dieses hilfreiche und manchmal auch spaßige Gerät intus haben. Im nächsten Kapitel erfahren Sie etwas über das Betriebssystem und wofür man sowas überhaupt braucht.

I.9 Was Sie jetzt schon wissen

* *Sie kennen jetzt die Bestandteile eines Computers, von der Tastatur bis zum Bildschirm*

* *Sie kennen die verschiedenen Druckertypen, ihre Vor- und Nachteile*

* *Sie wissen auch was ein Speichermedium ist*

* *Sie kennen die Zahlen Eins und Null, können binär zählen und wissen, wie die Werte gespeichert werden*

II. Das Betriebssystem

Im vorhergehenden Kapitel haben Sie einen kurzen Einblick in die Hardware eines Computers, die einzelnen Bauteile, erhalten. Wenn Sie den Materialwert eines solchen Computers nehmen, ist das Ding nicht viel wert. Grund: Sie können damit praktisch nichts anfangen. Richtig wertvoll wird ein Computer erst, wenn Sie darauf für Sie wichtige und nützliche Programme laufen lassen können. Damit das klappt, brauchen Sie die sogenannte Software. Auch hier müssen wir aber wieder unter verschiedenen Begriffen unterscheiden.

Um die allgemeine Verwirrung klein zu halten, beschäftigen wir uns mit zwei Typen von Software: Dem Betriebssystem und den Programmen. Sie sollten im Hinterkopf behalten, daß natürlich auch das Betriebssystem ein Programm ist. Der Unterschied den ich meine, kommt von der Funktion der unterschiedlichen Software her und ich will Ihnen das gerne erklären.

Häufig höre ich unerfahrene Leute mit Interesse an einem Computer fragen:" Kann der auch Textverarbeitung?" oder so ähnlich. Diese Leute haben den Unterschied zwischen einem Programm und einem Betriebssystem noch nicht verstanden, weshalb hier eine Erklärung folgt, falls Sie auch schon einmal so gefragt haben.

Stellen Sie sich bitte vor, wir haben es nicht mit einem Computer zu tun, sondern mit dem ganz normalen Leben. In einem solchen ist ein Butterbrot etwas ganz normales. Es besteht aus einer Scheibe Brot, Butter und einem Belag, der möglichst schmackhaft sein sollte. So ein Butterbrot kommt uns gerade recht, um den Unterschied zwischen Betriebssystem und Programm zu verdeutlichen.

Was Sie immer brauchen, ist die Scheibe Brot. Auf den Computer übertragen, ist das der Computer selber. Weil das Butterbrot eben Butterbrot heißt, brauchen Sie auch unbedingt Butter drauf (außerdem quietscht es dann nicht so beim Essen). Übertragen auf den Computer: Wenn Sie einen Computer haben und irgendetwas damit anfangen wollen, brauchen Sie ein Betriebssystem. Was Sie auf Ihr Butterbrot legen, ist Ihre Sache und richtet sich nach Ihrem Geschmack oder Ihrem Hunger. Genauso beim Computer. Welches Programm Sie verwenden, um auf dem Bildschirm zu spielen oder Texte zu verarbeiten oder sonstwas zu machen, liegt an Ihrem persönlichen Bedarf und Geschmack.

Spätestens jetzt werden Sie erkennen, daß die Frage weiter oben falsch gestellt ist. Denn: Wenn der Computer vorhanden ist und ein Betriebssystem vorhanden ist, dann läuft jedes Programm, das für diese Art von Computer, unter diesem Betriebssystem geschrieben wurde.

II.1 Was ist ein Programm?

Grundsätzlich besteht jedes Programm aus nacheinander aufgeschriebenen Befehlen, die vom Computer in der Reihenfolge abgearbeitet werden. Einer nach dem anderen. Der Computer braucht die Befehle in einer Sprache, die er versteht und verarbeiten kann. Weil er selber eine Maschine ist, heißt diese Sprache "Maschinensprache". Programme, die in Maschinensprache geschrieben sind, bleiben für "normale" Menschen unverständlich. Aber der Computer freut sich. Weil diese Sprache dem Computer quasi auf den Leib geschrieben ist, arbeitet er die Befehlsfolgen auch besonders schnell ab.

Für "normale" Menschen gibt es zur Arbeitserleichterung noch die sogenannten Hochsprachen oder Programmiersprachen. Sie kennen sicherlich einige davon. Besonders bekannt ist BASIC oder auch COBOL oder FORTRAN oder PASCAL oder ganz neu auch MODULA 2. Bei diesen höheren Programmiersprachen werden für Menschen verständliche Wörter als Kommandos benutzt, innerhalb des Computers in Maschinensprache übersetzt und vom Computer abgearbeitet.

Ein Programm ist eben nichts anderes als eine Folge von Befehlen, die dem Computer genau sagen, Schrittchen für Schrittchen, was er wann zu tun hat. Die Länge eines solchen Programms kann zwischen zwei Befehlszeilen und fast unendlich vielen Befehlszeilen liegen. Das kommt einfach nur noch auf die Speicherkapazität im Hauptspeicher des Computers und auf den Laufwerken an.

Nun reicht es aber nicht, daß ein Programm die Anforderungen des Benutzers erfüllt, zum Beispiel Textverarbeitung ermöglicht. Es muß auch intern, innerhalb des Computers, eine ganze Reihe von Nebenaufgaben wahrnehmen, damit die Sache überhaupt klappt. Zum Beispiel muß es feststellen, welche Taste auf der Tastatur gerade gedrückt wurde, welches Zeichen auf dem Bildschirm oder dem Drucker darzustellen ist, ob und wo auf dem Massenspeicher die gerade benötigten Daten oder Programme stehen oder Daten von der Diskette in den Arbeitsspeicher des Computers laden.

Weiter muß sichergestellt sein, daß die von der Diskette gelesenen Daten an die richtige Stelle im Arbeitsspeicher gelangen und daß auf der Diskette genügend Platz für neue Aufzeichnungen vorhanden ist. Außerdem muß ein Inhaltsverzeichnis auf der Diskette geführt werden, u.s.w., u.s.w.

II.2 Unterprogramme

Wie Sie sehen, eine ganze Menge Arbeit so quasi nebenbei. Und diese "Nebenbei-Arbeit" muß jedesmal für jedes Programm neu programmiert werden. Für jeden Teil eines Programms muß ganz klar festgelegt sein, was im Computer zu passieren hat. Erschwerend kommt hinzu, daß diese Aufgaben von Computertyp zu Computertyp, so ähnlich sie sich auch sehen mögen, jedesmal anders zu lösen ist. Ein klein bißchen anders als andere ist eigentlich jeder Computer. Würde man nun ein längeres Programm quasi in einem Stück schreiben, müßte es für jeden Computertyp umgeschrieben werden, um vernünftig abzulaufen.

Diese unnötige Arbeit kann man sich sparen, wenn man das Programm in ein Hauptprogramm und mehrere Unterprogramme aufteilt. Die Unterprogramme werden nur dann in den Arbeitsspeicher des Computers geladen, wenn sie wirklich gebraucht werden.

Um das Programm dann an einen anderen Rechnertyp anzupassen, nimmt man einfach das eine Blöckchen Unterprogramm weg, schreibt ein neues dazu, und schon läuft das gesamte Programm auf dem neuen Computer ohne Einschränkung. Ein weiterer Vorteil dieser Unterprogramm-Technik ist, daß das Hauptprogramm gar nicht mehr weiß, wie eine bestimmte Teilaufgabe gelöst wird. Es erteilt einfach einen Befehl, zum Beispiel "Textausgeben", und das Unterprogramm erledigt diese Arbeit. Ob der Text auf dem Bildschirm, einem Drucker oder über einen Fernschreiber ausgegeben wird, kann dem Hauptprogramm völlig gleichgültig sein. Es erteilt einfach nur den Befehl und die Arbeit wird erledigt.

Diese Methode funktioniert, wenn im Hauptprogramm die Übergabe der Daten an die Unterprogramme standardisiert ist. Das kann man sich vorstellen wie einen Staffellauf, wo immer an

genau vorherbestimmten Stellen der ablösende Läufer steht und den Stab übernimmt. In der Computersprache heißt eine solche genormte und genau festgelegte Übergabestelle eine Schnittstelle. Dieser Ausdruck ist durchaus bildhaft zu verstehen, weil man ein Unterprogramm einfach abschneiden und ein neues dransetzen kann, ohne daß die Funktion des Hauptprogrammes darunter leidet oder es diese Veränderung überhaupt bemerkt. Es paßt alles wieder genau aufeinander und funktioniert reibungslos.

Nun ist es ja eigentlich Quatsch, daß jeder Programmierer bei jedem Programm immer wieder aufs Neue die internen Operationen in sein Programm mit aufnimmt. Das macht viel Arbeit, kostet eine Menge Zeit und Geld. Diese Idee hatten die Programmierer von Digital Research auch und machten sich daran, ein Standardprogramm für Mikro-Computer zu schreiben. Sie faßten die häufigsten Unterprogramme, die zur internen Steuerung nötig sind, zusammen, definierten die Schnittstellen und die Art der Datenübergabe zwischen Haupt- und Betriebsprogramm und hatten damit ein gemeinsames Betriebssystem für die verschiedensten Rechnertypen geschaffen.

Einsetzbar ist dieses Betriebssystem natürlich nur auf Computern mit ähnlichen oder gleichen Zentraleinheiten, weil ja schließlich die Maschinenbefehle des Programms verstanden und abgearbeitet werden müssen. Das Betriebssystem, das als erstes die vielen internen Arbeitsoperationen standardmäßig zur Verfügung stellte heißt CP/M. Diese Abkürzung steht für "Control Program for Micro-Processors", was in deutsch schlicht und einfach Steuerprogramm für Mikro-Prozessoren heißt. Dieses Betriebssystem arbeitet mit Mikro-Prozessoren vom Typ 8080, 8085 oder Z80.

II.3 Die Aufgaben von CP/M

Im wesentlichen erledigt CP/M die folgenden Grundaufgaben: Eingabe von Zeichen, Ausgabe von Zeichen, Verwalten des verfügbaren Speicherplatzes auf dem Massenspeicher, Lesen von Disketten-Aufzeichnungen und Schreiben neuer Aufzeichnungen auf Diskette oder in den Massenspeicher. Diese Dienstroutinen können von allen Anwenderprogrammen in einheitlicher und standardisierter Weise benutzt werden.

Damit das klappt, muß CP/M aber zwei verschiedene Arten von Arbeit durchführen, weshalb es in zwei große Blöcke aufgeteilt ist: Der erste Teil, BDOS (Basic Disc Operating System = Grundbetriebssystem für Disketten) kümmert sich um alle Aufgaben, die unabhängig von dem jeweiligen Computersystem immer in der selben Art und Weise zu lösen sind. Der zweite Teil des Betriebssystems, das sogenannte BIOS (Basic I/O-System = Grundsystem zur Ein- und Ausgabe) enthält all die Programmteile, die für ein bestimmtes Computer-System angepaßt und notwendig sind.

Jedes Computermodell hat nämlich so seine eigenen Ansichten und Methoden, wie bestimmte Aufgaben erledigt werden, und man kann nicht einfach ein Betriebssystem von einem Gerät nehmen und es auf einem anderen laufen lassen. Bevor solch ein übernommenes Betriebssystem funktioniert, muß das BIOS auf die neue Maschine angepaßt werden. Normalerweise sollten Sie bei Ihrem Computer damit keine Schwierigkeiten haben, denn jeder Computer wird mit einem angepaßten und passenden Betriebssystem ausgeliefert (hoffentlich).

II.4 CP/M und die verschiedenen Versionen

Jedes Programm, das sich einige Zeit auf dem Computermarkt behaupten kann, erfährt auch Verbesserungen und kommt danach in einer neuen Version auf den Markt. Es gibt nun mal keine fehlerfreien Programme, und einige dieser Fehler werden erst entdeckt, wenn ein Programm in vielfältigem Einsatz ist und wirklich alle Möglichkeiten ausprobiert werden, die es überhaupt bietet.
Treffen genügend Fehlermeldungen bei dem Hersteller des Programms ein, verbessert er die Fehler und entschließt sich irgendwann, eine neue Version des Programms auf den Markt zu bringen. Als nahezu fehlerfrei kann CP/M in der Version 2.2 gelten, die praktisch das Standardbetriebssystem für alle 8-Bit-Rechner ist. Die neuere Version, CP/M 3.0, war also keine Neuausgabe um alte Fehler auszumerzen, sondern erfuhr eine deutliche Leistungssteigerung und erhielt zusätzliche, neue Befehle. Außerdem mußte CP/M an die immer leistungsfähiger werdenden Computer und Computertechniken angepaßt werden.

II.5 Der CP/M-Prompt

So, ich glaube das reicht. Lassen wir mal die Theorie da, wo sie hingehört und wenden uns dem wirklichen Leben, sprich dem Computer und seinem Betriebssystem zu. Was Sie als nächstes brauchen, ist ein funktionsfähiger Computer, ein Floppy-Laufwerk und eine Diskette mit dem CP/M-Betriebssystem drauf. Ist alles vorhanden, schalten Sie Ihren Computer ein, schieben die Diskette mit dem CP/M-Betriebssystem in Laufwerk "A" oder Laufwerk "1" und verriegeln es. Diese Vorgehensweise sollten Sie sich merken. Schalten Sie nämlich den Computer ein oder aus, während eine Diskette im Laufwerk

liegt, kann es durch den Stromstoß zu ungewollten Bewegungen des Schreib/Lesekopfs kommen, und Ihre wertvolle Diskette ist nicht mehr zu gebrauchen.

Also - Sie haben jetzt Ihren Computer eingeschaltet, Ihre CP/M-Systemdiskette eingelegt. Nun soll es losgehen. Es gibt nun drei Wege, wie Sie CP/M auf dem Commodore 128 starten können. Erster - nicht so empfehlenswerter - Weg: Einlegen der CP/M-Systemdiskette und Einschalten des Rechners. Zweiter - empfehlenswerter - Weg: Den RESET-Knopf betätigen. Der dritte Weg ist nicht viel komplizierter, allerdings macht er eine Eingabe erforderlich: Geben Sie das Kommando:

BOOT (RETURN)

ein. Das Betätigen des RESET-Knopfes sollte aber der "Standardweg" zur Erreichung des CP/M-Modus werden, er ist der sicherste und der schnellste.

Das Diskettenlaufwerk beginnt zu rattern und klickern, die Funktionslampe leuchtet auf, und kurz darauf sehen Sie auf dem Bildschirm eine Meldung. Diese sogenannte Anfangsmeldung sieht bei jedem Computer etwas anders aus. Das liegt daran, daß diese Meldung vom BIOS-Teil des Betriebssystems, also jenem Teil, der auf jeden Computer extra angepaßt werden muß, ausgegeben wird. Jeder Computerhersteller paßt sich diesen Teil für seine Maschine an und gestaltet die Anfangsmeldung nach seinen eigenen Vorstellungen.

Bevor diese commodorespezifische Anfangsmeldung auf dem Bildschirm erscheint, erkennen Sie auf dem Bildschirm noch die Meldungen "Booting..." und dann in blauer Schrift "Booting CP/M Plus".

Wie Sie wissen, können Sie an Ihren Commodore 128 einen Bildschirm mit 40 Zeichen/Zeile und einen Bildschirm mit 80 Zeichen/Zeile anschließen. Empfehlenswert ist das Arbeiten mit einem 80-Zeichen-Monitor, aber auch ein 40-Zeichen-Monitor wird von CP/M unterstützt. Mit der 40/80-Display-Taste können Sie auswählen, ob mit dem 40er-Bildschirm oder mit dem 80er-Bildschirm arbeiten wollen. Die Anfangsmeldungen erscheinen noch auf beiden Bildschirmen, nach dem CP/M-Prompt wird allerdings nur noch auf dem ausgewählten Bildschirm angezeigt.

Auf dem 40-Zeichen-Bildschirm wird durch Scrollen in der Horizontalen ein 80-Zeichen-Schirm simuliert. Mittels der TAB-Taste können Sie den Bildschirm verschieben. Wir wollen uns in diesem Buch immer mit dem 80-Zeichen-Bildschirm beschäftigen, sofern es nicht anders erwähnt wird.

Erscheint die Meldung:

```
NO CP/M+.SYS File - HIT RETURN TO RETRY
          DEL TO ENTER C128 MODE
```

dann haben Sie mit Sicherheit die verkehrte Seite der CP/M-Systemdiskette eingelegt. Drehen Sie die Diskette um und betätigen Sie die RETURN-Taste. Ihr Commodore wird dann erneut versuchen, CP/M zu booten.

Es kann aber auch zu einem Ladefehler kommen. Dann erscheint im unteren Teil des Bildschirmes ebenfalls eine Meldung, die aus folgenden Wortlaut besteht:

```
READ ERROR - HIT RETURN TO RETRY
          DEL TO ENTER C128 MODE
```

Wollen Sie, daß ein erneuter Boot-Versuch unternommen wird, so brauchen Sie lediglich die RETURN-Taste zu betätigen. Sind Sie allerdings von diesen Mißversuch so frustriert, daß Sie aufgeben wollen, so betätigen Sie einfach die DEL-Taste. Jede andere Taste wird übrigens ignoriert.

Läuft alles nach Plan, so schillern erst einmal folgende fünf Zeilen im unteren Teil des Bildschirmes aus:

```
BNKBIOS3 SPR  F400  0800
BNKBIOS3 SPR  CA00  1600
RESBOOS3 SPR  EE00  0600
BNKBDOS3 SPR  9C00  2E00

59K TPA
```

Dies sind Angaben über die einzelnen BIOS- und BDOS-Teile, an welcher Adresse diese geladen werden und welche Länge sie haben. Dies geht aber eigentlich recht schnell. 59K TPA bedeutet, daß 59 KBytes für ladbare Programme und Daten frei sind. TPA bedeutet übrigens *Transient Program Area.* Sind diese Dinge geladen, so erscheinen noch folgende vier Mitteilungen, diesmal etwas höher auf dem Bildschirm:

```
DATA TABLES
COMMON CODE
BANKED CODE
BIOS8502 CODE
```

Erst wenn auch diese Teile von Diskette geladen werden konnten, erscheint oben erwähnte typische CP/M-Meldung auf dem Bildschirm. Diese sieht beim Commodore 128 wie folgt aus:

```
CP/M 3.0 On the Commodore 128   3 JUNE 85
        80 column display (bzw. 40 column)
```

Sicherlich haben Sie bemerkt, daß während des Ladens in der rechten unteren Ecke immer irgendwelche Zahlen zu lesen waren. Auch wenn die Startmeldung erschienen ist, werden diese Zahlen nicht verschwinden. Die letzte Zeile des Bildschirmes ist die *Statuszeile*, die nicht erreicht oder überschrieben werden kann. Die Zahlen in der Ecke geben an, welcher Block gerade gelesen oder geschrieben wird. Ein R steht für Lesen, ein W für Schreiben eines Blockes. Ferner steht ein A oder ein B als Angabe, auf welchem Laufwerk gearbeitet wird.

Diese Informationszeile können Sie jederzeit mittels der Tastenkombination <CTRL><RUN/STOP> ein und auch wieder ausschalten, so, wie es Ihnen am besten gefällt.

A pros pos gefallen: Sollten Ihnen die Bildschirmfarben nicht gefallen? Kein Problem. Durch Betätigen der <CTRL>-Taste und einer der Tasten 1-8 können Sie die auf der Taste beschriftete obere Farbe als Zeichenfarbe auswählen. Auch die Hintergrundfarbe können Sie verändern, indem Sie die <CTRL>-Taste und eine der Zifferntasten 1-8 der Zehnertastatur betätigen. Wollen Sie beispielsweise schwarze Schrift auf weißen Grund, so müssen Sie <CTRL>-1 und <CTRL>-2 (Zehnertastatur) drücken. Das ist schon alles. Das CP/M auf dem Commodore hält noch weitere Überraschungen bereit.

Doch zurück zu unserer Anfangsmeldung: Diese Anfangsmeldung bedeutet für Sie, daß das CP/M-Betriebssystem richtig in den Arbeitsspeicher des Computers geladen wurde. Direkt anschließend meldet sich CP/M mit seiner Bereitschaftsmeldung, dem sogenannten Betriebssystem-Prompt. Dieser Prompt sieht so aus:

```
A>
```

Das große "A" sagt dem Benutzer, daß er mit dem Laufwerk "A" oder dem Laufwerk "1" seines Systems arbeitet, das ">" - Zeichen ist die eigentliche Bereitschaftsmeldung von CP/M. In der Zeile hinter der Bereitschaftsmeldung wartet CP/M auf die Eingabe eines Befehls. Na, dann wollen wir CP/M mal nicht so lange warten lassen und schreiben:

```
A>abcdefgh
```

Nun steht der Cursor hinter dem letzten Buchstaben, und nichts rührt sich, still ruht der See. Grund: CP/M weiß nicht, ob wir mit der Befehlseingabe bereits fertig sind oder vielleicht noch was dazuschreiben wollen. Um die Eingabe abzuschließen, müssen wir CP/M noch mitteilen, daß wir mit der Eingabe fertig sind.

Die Eingabe einer Zeile wird beim Computer wie bei der Schreibmaschine mit einem Wagenrücklauf beendet. Diese Taste heißt bei Computern häufig RETURN oder ENTER, was eine Abkürzung von CARRIAGE RETURN oder für "Eingabe" ist. Doch das hatten wir glaube ich ja bereits. Sobald Sie diese Taste drücken, weiß der Computer, daß Sie mit Ihrer Befehlseingabe fertig sind und beginnt den eingegebenen Befehl auszuführen. Sie drücken jetzt auf die RETURN-Taste und sehen

```
A>abcdefgh
ABCDEFGH?
A>
```

Tja, und was ist passiert? CP/M hat die Zeile gelesen, den Befehl nicht verstanden, und sagt uns das. Das Betriebssystem meldet uns solche unverständlichen Befehle, indem es die Eingabe wiederholt und mit einem Fragezeichen versieht. Das Fragezeichen bedeutet dabei etwa soviel wie: "Was soll der Quatsch? Ich kann damit nichts anfangen."

Das war also nichts. Versuchen wir doch mal, ob es vielleicht an der Kleinschreibweise gelegen hat. Nach dem Booten können Sie sowohl Klein- als auch Großbuchstaben eingeben (mittels SHIFT). Durch Betätigen der SHIFT-LOCK-Taste können Sie aber jederzeit ausschließlich Großbuchstaben eingeben. Übrigens tut die Tastenkombination <C=><SHIFT> genau dasselbe. Betätigen Sie nun die <SHIFT-LOCK>-Taste.

```
A>ABCDEFGH
```

Als Antwort darauf erhalten Sie:

```
A>ABCDEFGH
ABCDEFGH?
A>
```

Das Betriebssystem hat den Befehl also auch in Großbuchstaben nicht begriffen. Ist Ihnen zuvor aufgefallen, daß die Befehlszeile in Großbuchstaben wiederholt wurde, obwohl Sie sie in Kleinbuchstaben eingegeben haben? Das ist eine Eigen-

schaft von CP/M, die wir in manchen Fällen sogar ganz bewußt beachten müssen. CP/M wandelt alle Buchstaben die Sie eingeben, erst einmal in Großbuchstaben um und bearbeitet sie dann.

Damit Sie sehen, daß CP/M wirklich auch arbeiten kann, wollen wir jetzt einmal einen Befehl eingeben, der auch etwas bewirkt. Weiter vorne haben Sie gelesen, daß es zu den Aufgaben des Betriebssystems gehört, die Einträge auf der Diskette in einem Inhaltsverzeichnis zu verwalten. CP/M legt zu diesem Zweck auf jeder Diskette ein Inhaltsverzeichnis (in englisch Directory) an und zeigt das auf Verlangen vor.

Wenn wir dieses Inhaltsverzeichnis betrachten wollen, müssen wir dem Betriebssystem irgendwie mitteilen, daß wir es sehen wollen und daß es dieses Inhaltsverzeichnis auf dem Bildschirm darstellen soll. Der Befehl besteht aus den drei Buchstaben DIR, was eine Abkürzung für Directory ist. Geben Sie diesen Befehl jetzt ein und anschließend ENTER:

```
A>DIR
```

Kleine Anmerkung: Wie Sie es vielleicht von BASIC her gewohnt sind, konnte in BASIC durch Betätigen der F3-Taste das Inhaltsverzeichnis auf dem Bildschirm angezeigt werden. Damit die Umstellung nicht so schwierig ist, ist dies auch unter CP/M möglich. Betätigen Sie doch einfach mal die F3-Taste. Durch Betätigen der F4-Taste wird lediglich der Text DIR auf dem Bildschirm angezeigt, ohne automatischen Wagenrücklauf. So können Sie noch Optionen eingeben, doch dazu später mehr.

Sie sehen das Inhaltsverzeichnis Ihrer Betriebssystem-Diskette auf dem Bildschirm. Schauen Sie es sich jetzt

einmal genauer an. Ganz links wird das Laufwerk angegeben, in diesem Fall "A:". Daneben stehen die Namen von Dateien, und durch einen Leerraum getrennt, wird noch die Art der Datei angegeben. Sehr häufig sehen Sie den Zusatz COM, zum Beispiel bei CCP.COM, PIP.COM, HELP.COM. Diese Dateibezeichnung gibt Aufschluß über den Inhalt der Datei. Für uns sind im Moment die COM-Dateien besonders interessant, weil sie sofort ausführbare Programme enthalten.

COM ist die Abkürzung für das englische Wort "Command" oder Befehl. Die so bezeichneten Dateien enthalten Befehle, die sofort ausgeführt werden können. Wenn Sie den Namen eines solchen Programms ohne den Anhang COM hinter den Betriebssystem-Prompt eintippen, wird das Programm direkt in den Arbeitsspeicher des Computers geladen und ausgeführt. Es genügt also, einfach PIP einzugeben, ein RETURN dazu, und schon startet das Programm.

II.6 Sicher ist sicher

Falls Sie jetzt noch immer mit Ihrer Original-Diskette arbeiten, müssen wir uns schleunigst darum kümmern, eine Sicherheitskopie zu erstellen. Sie sollten sich angewöhnen, nie mit den Originalen eines Programms zu arbeiten, sondern immer eine Kopie des Programms zu erstellen und das Original gut und sicher aufzubewahren. Wie sich das für ein Betriebssystem gehört, das dem Benutzer ja die Arbeit erleichtern soll, ermöglicht CP/M die Herstellung von Sicherheitskopien.

II.7 Was Sie jetzt schon wissen

* *Sie kennen eine einfache Erklärung, was ein Betriebssystem für Aufgaben zu bewältigen hat*

* *Ihnen ist klar, was ein Programm ist und wofür Unterprogramme da sind*

* *Sie kennen auch die Aufgaben von CP/M*

* *Sie wissen, was der CP/M-Prompt ist und wie Sie das Inhaltsverzeichnis Ihrer Disketten anzeigen können*

* *Sie haben sich fest vorgenommen, von jeder wichtigen Diskette mindestens eine Sicherheitskopie anzulegen*

III. Arbeiten mit CP/M

III.1 Die System-Diskette

Ich habe Sie bereits darauf hingewiesen, daß Sie von Ihren wertvollen Originaldisketten Kopien anfertigen sollten, damit im Falle eines Falles Ihr Original nicht beschädigt wird. Wie eine komplette Diskette kopiert wird, will ich Ihnen jetzt zeigen, und wir benutzen dazu CP/M-Befehle, die erst später genauer erklärt werden. Sie brauchen zwei Programme von der CP/M-Diskette, um eine komplette Diskette kopieren zu können. Wenn Sie noch einmal DIR eingeben und sich das Inhaltsverzeichnis genau ansehen, werden Sie unter anderem zwei Programme finden, die heißen:

FORMAT.COM und PIP.COM

Normalerweise kopiert man das System CP/M mittels des Kommandos COPYSYS. Doch dieses Kommando ist beim Commodore CP/M nicht implementiert - also nicht vorhanden. Sie können zwar, sollten Sie das Kommando COPYSYS gewohnt sein, dieses Kommando eingeben, doch auf dem Bildschirm erscheint dann lediglich die Meldung, daß das Kopieren des Systems auf dem Commodore 128 anders zu bewerkstelligen sei.

Legen Sie also die Seite der Systemdiskette ein, die mit "System Disk" beschriftet ist, und dann kann es losgehen. Achten Sie darauf, daß der Schreibschutz aus Sicherheitsgründen nicht entfernt wird.

Die Aufgabe, die sonst das Kommando COPYSYS übernommen hat, nämlich das Kopieren des Systems auf die oberen beiden Spuren 0 und 1, wird nun sinnvollerweise direkt vom Kommando

FORMAT ausgeführt. So sind diese beiden Spuren zwar **immer** belegt, doch hat dies auf die Speicherfähigkeit der Diskette nur wenig Einfluß. Ferner präpariert die FORMAT-Routine auch die Diskette direkt mit einem BOOT-Sektor, so daß nach einem RESET oder der Eingabe des Kommandos BOOT das CP/M auch gebootet werden kann. Für den unbedarften Anwender wird dadurch die Arbeit sehr vereinfacht.

Die im *normalen* CP/M unsichtbaren Systemspuren erhalten beim Commodore 128 einen Namen: Es handelt sich hierbei um die Dateien **CPM+.SYS** und **CCP.COM**.

In der Datei CCP.COM verstecken sich die sogenannten residenten Kommandos, also die Kommandos, die con CP/M auch verstanden werden, ohne daß eine COM-Datei von Diskette geladen werden muß. Diese Kommandos sind permament im Speicher Ihres Commodore 128.

Bevor Sie sich nun an die Kopierarbeit machen, sollten Sie dafür sorgen, daß Sie eine frisch formatierte und leere Diskette zur Verfügung haben (Eine Diskette wird mit dem CP/M-Kommando FORMAT formatiert, indem Sie diesen Befehl hinter den CP/M-Prompt eingeben und den auf dem Bildschirm erscheinenden Anweisungen folgen).

III.2 Kopieren mit einem Laufwerk

Haben Sie alles parat, legen Sie Ihre CP/M-Originaldiskette in das Laufwerk A ein und schließen die Laufwerkstür. Arbeiten Sie mit nur einem Laufwerk, bekommen Sie ein wenig Mehrarbeit, als wenn Sie mit zwei Laufwerken operieren könnten. Aber es geht auch so, wenn Sie die folgenden Schritte befolgen. Um Ihr Betriebssystem zu kopieren, schieben Sie die Originaldiskette in das Laufwerk und geben ein:

A>FORMAT (RETURN)

Daraufhin erscheint diese Meldung auf dem Bildschirm:

```
C128 FORMAT PROGRAM
   15 May 1985
Drive A is a 1571 (1541)

Please select disk type to format
C128 double sided
C128 single sided
C64 single sided
```

Sie können jetzt mit den Cursor-Tasten in der ersten Tastenreihe eines der drei Formate auswählen, in dem Sie Ihre Diskette formatiert sehen wollen. Zu beachten ist für 1570-Besitzer, daß diese natürlich *nicht* doppelseitig formatieren können. Dies ist schließlich der kleine aber feine Unterschied der 1570 und der 1571: Die 1570 kann die Disketten nicht doppelseitig lesen und beschreiben.

Lassen Sie uns deswegen unsere erste Diskette zunächst einmal einseitig (single sided) formatieren. Dazu müssen Sie einmal die Cursor-Runter-Taste auf der oberen Tastenreihe betätigen, dann können Sie die RETURN-Taste betätigen. Auf dem Bildschirm erscheint dann:

```
Formatting C128 single sided (invers)

Insert diskette TO BE FORMATTED
in drive A.  Type $ when ready,
any other key to abort
```

Dies heißt nichts anderes, als daß Sie die zu formatierende Diskette ins Laufwerk einlegen möchten. Sollten Sie sich bei der Wahl des zu formatierenden Formates geirrt haben, dann kommen Sie aus dieser Misere durch Betätigen einer Taste, die nicht die $-Taste ist, wieder heraus. Wollen Sie aber die nun eingelegte Diskette formatieren, so betätigen Sie einfach die $-Taste, dann geht's los:

```
Formatting C128 single sided (invers)
```

Sollte beim Formatieren ein Fehler auftreten, beispielsweise weil die eingelegte Diskette nicht ganz in Ordnung ist, so wird der Bildschirm gelöscht und es erscheint die Meldung:

und die rote LED-Anzeige des Diskettenlaufwerkes blinkt. Man sollte dann seine Eingabe kontrollieren (hat man beispielsweise bei der 1570 versucht doppelseitig zu formatieren) und evtl. auch die Diskette, um es dann erneut zu versuchen. Bei uns hat es natürlich keinen Fehler gegeben, so daß das FORMAT-Programm folgenden Text ausgibt:

```
Do you want to format another disk?
```

Diese Frage können Sie ganz getrost mit N für Nein beantworten, da uns momentan mit einer formatierten Diskette gedient ist. Wollen Sie aber eine weitere Diskette formatieren, so betätigen Sie bitte die Y-Taste, da der Dialog unter CP/M, wie Ihnen sicherlich nicht verborgen geblieben ist, in Englisch durchgeführt wird - leider muß man sagen, für alle diejenigen unter Ihnen, die der englischen Sprache nicht mächtig sind.

Nachdem wir nun eine Diskette formatiert haben, der Vorgang sieht bis hierhin übrigens für Besitzer von zwei Laufwerken vollkommen identisch aus, müssen wir noch die zwei angesprochenen Dateien kopieren. Dies geschieht mittels des PIP-Kommandos, dem wir übrigens ein extra Kapitel gewidmet haben, da es so umfangreich und gleichermaßen wichtig für den CP/M-Anwender ist. Auch wenn Sie jetzt noch nicht wissen, was Sie tun, so sollten Sie sich davon nicht beirren lassen, schon bald werden die Eingaben für Sie keine bömischen Dörfer mehr sein. Momentan müssen wir aber unbedingt eine Sicherheitskopie von CP/M erstellen - sicher ist sicher.

Also lassen wir den Worten Taten folgen, und geben Sie folgende Zeile ein:

PIP E:=A:CPM+.SYS

Erschrecken Sie nicht - nach einiger Zeit des Ladens erscheint in der untersten Zeile (richtig, die Statuszeile) die Aufforderung

```
Insert Disk E in Drive A
```

Sie werden hier aufgefordert, die Systemdiskette dem Laufwerk zu entnehmen um dafür die eben formatierte Diskette (die hier mit E betitelt wird) ins Laufwerk einzulegen. Haben Sie dies getan, zieren Sie sich nicht und betätigen Sie die RETURN-Taste. Die Meldung verschwindet und das Diskettenlaufwerk beginnt wieder zu arbeiten. Es kopiert momentan die eingelesene Datei CPM+.SYS auf die neue Diskette. Ist die Arbeit beendet, erscheint das CP/M-Prompt wieder auf dem Bildschirm. Auf genau dieselbe Art und Weise müssen Sie jetzt auch noch die Datei CCP.COM kopieren. Dies erreichen Sie, indem Sie folgendes ähnliches Kommando eingeben:

PIP E:=A:CCP.COM

Das CP/M ist aber nicht dumm und weiß, daß Sie die Zieldiskette mit dem Namen E noch im Laufwerk haben. Es fordert Sie also in der Statuszeile auf, die Diskette A wieder einzulegen - was Sie natürlich machen und mit der RETURN-Taste quittieren. Erneut legen Sie die E-Diskette ein

und auch diese Datei CCP.COM wird kopiert. Wollen Sie sich nun das Inhaltsverzeichnis der "neuen" CP/M-fähigen Diskette ansehen, so geben Sie das Ihnen bekannte Kommando DIR ein. In der Statuszeile werden Sie aufgefordert, die A-Diskette einzulegen. Lassen Sie nun Ihre Diskette ruhig im Laufwerk, durch Betätigen der RETURN-Taste machen Sie aus der E-Diskette die A-Diskette. Dies ist alles. Auf dem Bildschirm erscheint:

```
A: CPM+      SYS : CCP      COM
```

Damit ist der erste Schritt getan. Da wir aber zunächst die *gesamte* Diskette aus Sicherheitsgründen kopieren wollen, müssen wir auch *alle* Dateien kopieren. Dies erreichen Sie, indem Sie folgendes Kommando eingeben:

```
PIP E:=A:*.*
```

Auf dem Bildschirm werden Sie informiert, welche Datei im Augenblick kopiert wird. In der Statuszeile werden Sie immer wieder zum Wechsel der Disketten aufgefordert, das geht nun mal nicht anders mit einem Laufwerk. Folgen Sie den Anweisungen, wechseln Sie immer die Disketten und betätigen Sie anschließend die RETURN-Taste. Wenn Sie fertig sind, haben Sie schonmal die erste Seite der CP/M-Systemdiskette kopiert.

```
COPYING -
CPM+.SYS
CCP.COM
HELP.COM
```

```
HELP.HLP
KEYFIG.COM
KEYFIG.HLP
FORMAT.COM
PIP.COM
DIR.COM
COPYSYS.COM
```

Das war dann die erste Seite - wollen Sie auch noch (und das solten Sie) die zweite Seite kopieren, dann holen Sie sich eine zweite Diskette, formatieren Sie diese ebenfalls, und geben Sie auch das Kommando PIP E:=A:*.* ein. Alles andere wird Ihnen auf dem Bildschirm angezeigt. Haben Sie beide Seiten kopiert, dann sollten Sie am besten nur noch mit den Kopien des CP/M arbeiten, und die Originale an einem sicheren Ort aufbewahren - man weiß ja nie was alles passieren kann.

III.3 Kopieren mit zwei Laufwerken

Sollten Sie glücklicher Besitzer von zwei Diskettenlaufwerken sein, so geht das Kopieren von Dateien für Sie etwas schneller und bequemer von sich. An der Bedienung ändert sich aber praktisch garnichts, bis auf die Tatsache, daß Sie anstatt des virtuellen Laufwerkes E: das tatsächlich vorhandene B: angeben müssen. Dies bedeutet, daß der Formatiervorgang exakt genauso abläuft, wie unter III.2 beschrieben, ein Formatieren ist nämlich nur auf dem A-Laufwerk möglich - leider.

Beim PIP-Kommando müssen Sie nun anstatt PIP E:=a:*.*

PIP B:=A:*.*

eingeben. Es erscheint in der Statuszeile keine Aufforderung zum Wechseln der Disketten. Bevor Sie diese Zeile eingeben, sollten Sie allerdings die frisch formatierte Diskette ins B-Laufwerk eingelegt haben. Verfügen Sie über kein B-Laufwerk oder ist dieses nicht ordentlich angeschlossen oder gar ausgeschaltet, so erhalten Sie folgende Fehlermeldung:

```
ERROR: OPEN FILE NONRECOVERABLE - B:CPM+.SYS
```

Überprüfen Sie dann Ihre Eingabe bzw. die Anschlüsse der Diskettenlaufwerke untereinander.

III.4 Inhaltsverzeichnis ansehen

Sie haben etwas von A nach B kopiert und sind neugierig, was auf der B-Diskette alles steht. Damit Sie nicht vor Neugier platzen, geben Sie direkt hinter den CP/M-Prompt ein:

DIR B:

Wenn Sie die Leerstelle nach DIR nicht vergessen haben, werden Sie sehen, wie das Laufwerk B zu arbeiten beginnt, und Sie eine Meldung auf dem Bildschirm bekommen. Mit DIR B: haben Sie CP/M angewiesen, den Inhalt der Diskette im Laufwerk B anzuzeigen. Geben Sie dieses Kommando bei einer Diskette ein, die Sie frisch formatiert haben, ohne etwas mittels PIP kopiert zu haben, so erscheint jetzt die Meldung NO FILE. FILE wird im Deutschen mit Datei übersetzt und NO

FILE bedeutet demnach nichts anderes, als daß nichts auf der Diskette steht. Wie Sie sich erinnern, werden die Programme der ersten beiden Systemspuren jeder Diskette versteckt. Sie haben darauf keinen direkten Zugriff und können sie auch nicht im Inhaltsverzeichnis sehen.

III.5 Kopieren mit PIP

Normalerweise dient PIP beim Kopieren von Dateien lediglich, wenn man im Besitze von zwei Diskettenlaufwerken ist. Dies ist beim Commodore 128 anders: Man kann durch Angabe des Diskettenlaufwerkes E:, das physikalisch nicht mehr möglich ist (lediglich A:-D: sind möglich), ein sogenanntes *virtuelles* Laufwerk ansprechen, das wir und der Rechner uns nur *ausdenken*, das aber in Wirklichkeit gar nicht existiert. Es wird immer das A-Laufwerk angesprochen, der Rechner unterrichtet uns aber immer darüber, welche Diskette man einzulegen hat. Man sollte sich vorher merken, welche Diskette die A- und welche die E-Diskette ist. Der Trick mit dem virtuellen Laufwerk funktioniert übrigens nicht nur beim PIP, doch dazu später mehr.

Das zweite Programm, das wir zu, Kopieren des Systems benötigen, hat den unendlich langen, deutschen Namen: Prozessor für Datenaustausch zwischen Peripherie-Einheiten, was im Englischen zu Peripheral Interchange Processor wird, und abgekürzt ein lustiges PIP ergibt. Was nun dieses Programm macht, ist überhaupt nicht zum Piepen, sondern unerhört nützlich.

Sie haben die Systemdiskette im Laufwerk und tippen nun PIP und anschließend das RETURN ein. Der Computer lädt PIP nun in den Arbeitsspeicher, worauf sich dieses mit einem eigenen PROMPT meldet. Das sieht so aus:

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

Der Stern zeigt Ihnen an, daß PIP zur Aufnahme von Befehlen bereit ist. Dieses Programm ist ausgesprochen nützlich und leistungsfähig. Sie werden die vielen Möglichkeiten von PIP im Kapitel 6 genauer kennenlernen.

Bevor wir PIP nun einen entsprechenden Befehl geben, überlegen wir uns, was es eigentlich machen soll. Die Dateien von der Diskette in Laufwerk A sollen auf die Diskette in Laufwerk B (oder nach "Laufwerk E:, verfügt man lediglich über ein Diskettenlaufwerk) übertragen werden. Man könnte auch sagen, "B:" bekommt alle Dateien, die auf "A:" gespeichert sind.

Den letzten Satz sollten Sie sich genau ansehen und vor allem gut einprägen. Er ist entscheident bei der Arbeit mit PIP. Wenn alle Dateien von A nach B übertragen sind, ist der Inhalt beider Disketten gleich. Deshalb wird in PIP-Befehlen das Gleichheitszeichen benutzt. Bleibt nur noch die Schwierigkeit, wie wir PIP sagen, daß es alle Dateien nehmen soll und nicht nur irgendeine einzelne.

Für diesen Fall ist vorgesorgt. Das Sternchen anstelle eines Dateinamens bedeutet in PIP soviel wie "alle Dateien". Sie können das Sternchen wie einen Joker beim Kartenspiel verstehen, der für jede beliebige Karte eingesetzt werden kann. Weil sich Dateien aber einmal durch den Dateinamen, zweitens aber durch die Dateikennung, das sind die drei kleinen Buchstaben hinter dem Punkt, unterscheiden, müssen Sie fürs Kopieren aller Dateien ein Sternchen/Punkt/Sternchen einge-

ben. Das sieht auf dem Bildschirm so aus

```
B:=A:*.*   bzw.   E:=A:*.*
```

Weil wir sichergehen wollen, daß alle Dateien auch wirklich korrekt übertragen werden, lassen wir gleich jede Datei überprüfen, ob sie auch richtig rübergekommen ist. Wir können PIP diese Arbeit übertragen, indem wir ein "V" in eckigen Klammern hinter unseren Befehl schreiben. "V" steht für "Verify", was soviel wie nachprüfen, verifizieren heißt. Damit PIP diesen Befehl richtig versteht, muß er allerdings in eckigen Klammern stehen. Allerdings führt der Commodore bereits automatisch ein *Verify* aus, so daß die Angabe der Option [V] praktisch entfallen kann. Haben Sie den kompletten Befehl eingegeben,

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

dann geht's auch schon los. Wie Sie während des Kopiervorgangs sehen, teilt Ihnen PIP mit, welche Datei gerade kopiert wird. Ist die Arbeit getan, meldet sich PIP wieder mit seinem Bereitschaftszeichen, dem Stern. Wenn Sie für PIP keine weitere Arbeit haben, und ich glaube, wir haben im Moment nichts mehr zu tun für PIP, geben Sie RETURN ein, und das Betriebssystem meldet sich wieder mit seinem A-Prompt.

Sollten Sie der Technik nicht so ganz trauen, können Sie mit DIR (B:) nachsehen, ob wirklich alle Dateien auf dem B:-Laufwerk (bzw. der zweiten Diskette) angekommen sind. Ist alles zufriedenstellend, nehmen Sie die Diskette aus dem A-Laufwerk, also Ihre Original-Diskette, verfrachten Sie in die Disketten-Hülle und legen diese Diskette weit und gut

gesichert weg. Sie arbeiten in Zukunft nur noch mit der Kopie, die Sie eben erstellt haben und ersparen sich viel Ärger, wenn mal was schiefgeht. Denn: Eine Diskette, um ein neues System zu kopieren, kostet Sie etwa fünf Mark, ein neues CP/M-Betriebssystem dagegen kostet gleich einige hundert Märker.

III.6 Regeln für Dateinamen

Bis jetzt haben Sie schon einige Male etwas über Dateien gehört, haben auch Datei-Namen im Inhaltsverzeichnis auf Ihrem Bildschirm gesehen, aber was eine Datei ist und was es damit auf sich hat, haben Sie noch nicht erfahren.

Wie Sie wissen, besitzt jede Datei in CP/M einen Namen. Das ist einmal wichtig für Sie, damit Sie wissen, was drinsteht, zweitens für CP/M wichtig, damit es diese Datei auf Ihren Aufruf hin auch finden kann. Leider haben die CP/M-Konstrukteure eine Beschränkung eingebaut, die zum Beispiel mir überhaupt nicht paßt.

Ein Datei-Name in CP/M darf aus maximal acht Zeichen bestehen, einem Punkt und wiederum drei Zeichen für die sogenannte Kennung. Das bedeutet, Sie müssen für Ihre Dateien Namen finden, die maximal acht Zeichen lang sind und trotzdem einen Sinn und einen Rückschluß auf deren Inhalt ergeben.
Die zulässigen Zeichen für einen Datei-Namen sind alle 26 Buchstaben des Alphabeths sowie das Plus-, Minus-, Schrägstrich-, Prozent- und Dollarzeichen in jeder Position des ersten Namens oder innerhalb der Kennung. Die Zeichen "kleiner als", "größer als", Punkt, Komma, Doppelpunkt, Gleichheitszeichen, Strichpunkt, Stern, Fragezeichen, eckige Klammer auf, eckige Klammer zu, Ausrufezeichen sollten Sie

in Datei-Namen oder der Kennung nicht benutzen, da diese Zeichen eine bestimmte Bedeutung für CP/M haben und zu Fehlfunktionen führen können.

Aus meiner langjährigen Erfahrung mit CP/M und anderen Programmen kann ich Ihnen nur raten, Ihren Dateien immer einen sinnvollen Namen zu geben, der Rückschlüsse auf den Inhalt der Datei zuläßt. Besonders wenn Sie viele Dateien auf Ihrer Diskette stehen haben, kommen Sie mit einem Namen wie XK2512AC.ABC überhaupt nicht mehr zurecht. Bei der Wahl Ihres Datei-Namens brauchen Sie ansonsten keine besondere Vorsicht walten zu lassen, weil zwei Dateien mit dem gleichen Namen und der gleichen Kennung nicht gleichzeitig auf einer Diskette stehen können. Allerdings können Dateien mit gleichem Namen und unterschiedlicher Kennung gleichzeitig auf einer Diskette stehen.

III.7 Datei-Kennung

Die Kennung eines Datei-Namens können Sie frei gestalten, d.h. Sie können jede Datei benennen, wie Sie wollen, sollten dabei aber auf CP/M-typische Abkürzungen achten. Eine Tabelle der in CP/M üblichen Abkürzungen und deren Bedeutung finden Sie am Ende dieses Kapitels. Dort finden Sie zum Beispiel Programme mit der Kennung COM. Das ist die Abkürzung für COMMAND und bedeutet, daß diese Programme sofort ausführbaren Befehlscode enthalten.
Sie erinnern sich, daß PIP und SYSGEN oder COPYSYS solche COM-Programme sind. Außerdem sollten Sie sich davor hüten, Dateien mit der Kennung BAK oder $$$ zu benennen. BAK steht für "back up", was soviel wie Sicherheitskopie heißt und wird vom CP/M-Editor benutzt, um Sicherheitskopien von Dateien zu bezeichnen. Die Datei mit den drei Dollarzeichen am Ende versteht sich als Zwischendatei, die von einem Programm

angelegt und am Ende des Programmlaufs rigoros gelöscht wird. PIP zum Beispiel legt sich solche Zwischendateien an und löscht diese am Ende total von der Diskette. Also lieber Vorsicht.

Ungeschickt ist es auch, gleiche Datei-Namen zu verwenden, und nur die Kennung zum Zählen der Dateien zu verwenden. Beispiel: Sie haben einen Text und benennen die erste Datei TEXT.1, die zweite Datei TEXT.2 und die dritte Datei TEXT.3. Was im Moment recht logisch aussieht, wird Sie nachher ärgern. Sobald Sie nämlich die erste Datei bearbeitet haben, wird aus Ihrer Original-Datei die neue Datei TEXT.BAK und die überarbeitete Version heißt jetzt TEXT.1. Nun überarbeiten Sie die Datei TEXT.2, und auch hier wird die Original-Datei zu TEXT.BAK und die überarbeitete Version zu TEXT.2.

Da bei CP/M keine zwei Dateien mit genau dem gleichen Namen und der gleichen Kennung auf einer Diskette stehen dürfen, hat das Betriebssystem kurzerhand Ihre erste BAK-Datei gelöscht und durch die zweite überschrieben. Somit haben Sie keine Möglichkeit mehr, den Text Ihrer ersten Original-TEXT.1-Datei nochmal anzusehen.

Um dieses Problem zu umgehen, benennen Sie Ihre erste Datei mit TEXT1.TXT, Ihre zweite Datei mit TEXT2.TXT und Ihre dritte Datei mit TEXT3.TXT.

Wenn Sie nun die Dateien überarbeiten, bekommen Sie für jede eine eigene Back-up-Datei und haben jederzeit die Möglichkeit, Ihren Originaltext wieder hervorzuholen.

III.8 Eine Datei wiederfinden

Wenn Sie Ihren Computer intensiv nutzen und viele verschiedene Dateien erstellen, stehen Sie möglicherweise bald vor einem neuen Problem. Sie finden die Dateien, die Sie suchen, nicht mehr auf Anhieb in Ihrem Inhaltsverzeichnis. Ihnen kann es beispielsweise aber recht nützlich sein herauszufinden, wieviel Dateien mit dem Namen TEXT.TXT auf der Diskette bereits stehen.

Sie haben zwei Möglichkeiten, nach diesen Dateien suchen zu lassen. Als Helfer dienen hier in beiden betriebssystem-Versionen der Stern und das Fragezeichen. Den Stern haben Sie schon als sogenannten Joker bei unserer Kopier-Operation mit PIP kennengelernt, das Fragezeichen arbeitet ähnlich. Während der Stern stellvertretend für alle Zeichen des Datei-Namens oder der Kennung steht, arbeitet das Fragezeichen als Ersatz für ein Zeichen an einer bestimmten Position.

Die Eingabe "Text?" zum Beispiel sucht und findet alle Dateien mit dem Namen "Text" oder mit dem Namen "Text" plus einem weiteren Zeichen. So würden auch alle drei Dateien "TEXT1", "TEXT2" und "TEXT3" gefunden. Sollte zusätzlich noch eine Datei "TEXT" auf unserer Diskette stehen, würde diese auch mitgefunden und angezeigt. Das liegt daran, daß CP/M für einen Datei-Namen immer acht Zeichen reserviert. Ist Ihr eingegebener Datei-Name kürzer, füllt CP/M die restlichen Plätze mit Leerzeichen auf. Dabei sind Leerzeichen genauso gültige Zeichen wie Buchstaben oder Ziffern.

III.9 Suchen mit Fragezeichen (?)

Da das Fragezeichen für jedes Zeichen gesetzt werden kann, werden nicht nur die Dateien "TEXT1", "TEXT2" und "TEXT3" gefunden, sondern auch die Datei "TEXT". Der Einsatz des Fragezeichens bleibt aber nicht auf ein Zeichen beschränkt. Sie können genauso gut bis zu acht Fragezeichen für den Datei-Namen eingeben und bis zu drei Fragezeichen für die Kennung. Das aber ist eine lästige Tipperei, die Sie sich mit der Eingabe des Sternchens für den Datei-Namen und eines Sternchens für die Kennung ersparen können. Dem Betriebssystem ist es völlig gleichgültig, ob Sie acht Fragezeichen für den Datei-Namen oder ein Sternchen eingeben. Intern wird sowieso jedes Sternchen in acht Fragezeichen umgewandelt, bevor die Suche beginnt.

Sie können auch mit dem Sternchen genauer suchen, wenn Sie vor den Stern den oder die Anfangsbuchstaben der gesuchten Dateien schreiben. Geben Sie allerdings ein Sternchen und dann noch ein paar Buchstaben ein, funktioniert die Suche nicht. Diese Methode, das Eingeben von Datei-Namen mit Fragezeichen oder Sternchen, können Sie zusammen mit den Befehlen DIR, TYPE, ERASE, SHOW und PIP verwenden.

III.10 Was Sie jetzt schon wissen

* *Sie können nun eine Systemdiskette kopieren und Ihr Original gut aufheben*

* *Mit dem Stern und den Fragezeichen als Ersatz für fehlende Buchstaben können Sie auch umgehen und sich das Leben leichter machen*
* *Sie wissen, daß man Dateien immer sinnvolle Namen geben soll, damit man sie auch wiederfindet*

IV. Eingebaute Befehle

Im letzten Kapitel haben wir ein paar Grundfunktionen des Betriebssystems CP/M kennengelernt. Das ist aber längst noch nicht alles. In diesem Kapitel werden wir uns genauer mit den eingebauten CP/M-Befehlen befassen, die da lauten: USER, DIR, DIRS(YS), ERA(SE), REN(AME) und TYPE.

Außerdem kümmern wir uns später noch um die sogenannten transienten Programme (das sind CP/M-Programme, die Sie im Inhaltsverzeichnis sehen können und die als COM-Dateien gespeichert sind). Zusätzlich werden Sie Ihre Kenntnisse über die Befehle DIR und PIP, die Sie in dem vorherigen Kapitel schon kennengelernt haben, erweitern.

IV.1 Befehle, Parameter und Optionen

Bevor ich Ihnen weitere CP/M-Befehle erkläre, sollen Sie noch den grundsätzlichen Unterschied zwischen einem Befehl, einem Parameter und einer Option kennenlernen. Ein Befehl ist ein Befehlswort, das CP/M anweist, eine bestimmte Aktion auszuführen. Ein Parameter ist in der Regel ein Dateiname, der angibt, auf welche Datei sich der Befehl bezieht. Eine Option steht bei CP/M in eckigen Klammern und kann, wie der Name schon sagt, auch weggelassen werden. Wird aber eine Option mit eingegeben, verändert sie auf eine bestimmte Weise die Funktion des vorne eingegebenen Kommandos.

Ein Beispiel für eine Option haben Sie bereits gesehen, als wir mit PIP Ihre Systemdiskette kopiert haben. Die Option lautete "V", und veranlaßte PIP, alle kopierten Dateien auch zu überprüfen.

Jenachdem, um welchen Befehl es sich handelt, werden Parameter mit eingegeben oder nicht. Sollte irgendwo ein Parameter notwendig sein, und Sie haben keinen eingegeben, verlangt CP/M ausdrücklich die Eingabe des Parameters. Sie brauchen also keine Angst zu haben, daß etwas nicht funktioniert.

Bei der Eingabe von Befehlen und Parametern müssen Sie darauf achten, daß mindestens ein Leerzeichen zwischen dem Befehl und dem Parameter steht. Und das ist auch der einzige Ort, wo eine Leerstelle auftauchen darf. Weder im Befehl, noch im Parameter, noch in der Option dürfen an irgendwelchen Plätzen Leerstellen auftauchen.

Normalerweise werden Sie nur einen oder zwei Dateinamen nach einem Kommando als Parameter eingeben. Sie sind allerdings nicht gezwungen, sich auf so wenige Angaben zu beschränken. Wenn Ihnen genug einfällt oder es Ihre Anwendung verlangt, können Sie auch die ganze Zeile vollschreiben, mit Control E (^E) eine neue Zeile anfangen und dort weitermachen. Das Commodore-CP/M beginnt allerdings auch selbständig eine neue Zeile - sobald dies notwendig werden sollte. CP/M wird diese beiden Zeilen als eine Befehlszeile lesen.

IV.2 Die eingebauten Befehle

Sobald CP/M von der Diskette in den Arbeitsspeicher im Computer geladen ist, stehen Ihnen zwei Sorten von Kommandos zur Verfügung. Dieses Einladen der residenten Kommandos erfolgt beim Commodore 128 durch Einlesen der Datei CCP.COM, wie bereits schon etwas vorher beschrieben wurde. Die eingebauten Kommandos sind im Arbeitsspeicher Ihres Computers und können sofort und sehr schnell von dort abgerufen werden, um irgendwelche Aufgaben zu erfüllen. Die anderen Kommandos stehen als Datei auf Ihrer Diskette und Sie können sie

auflisten, wenn Sie sich das Inhaltsverzeichnis ansehen. Diese Programme lassen sich wie ganz normale Dateien kopieren oder löschen oder, wenn Sie genügend Kenntnisse haben, auch verändern.

Diese Aufteilung in CP/M wurde gemacht, weil die beiden Systemspuren auf jeder Diskette, die extra für CP/M reserviert sind, längst nicht genug Platz für alle Hilfsprogramme bieten. Für Ihre Arbeit mit dem Computer ist es aber unerheblich, ob Sie ein eingebautes Kommando oder ein transientes Kommando aufrufen. Verlangen Sie zum Beispiel mit dem Kommando PIP ganz besondere Aktionen, lädt CP/M automatisch die Datei PIP.COM in den Arbeitsspeicher und führt Ihren Befehl aus.

Den einzigen Verlust, den Sie haben, ist etwas Zeitverlust durch das Lesen der Datei von der Diskette. Insgesamt besitzt das Betriebssystem CP/M 2.2 fünf und die Version 3.0 mittlerweile sechs eingebaute Kommandos, wobei hier vier eine Erweiterung durch eine gleichnamige COM-Datei erfahren. Wenn Sie in CP/M 3.0 eins dieser Kommandos aufrufen, müssen Sie nicht jedesmal den kompletten Namen eingeben. Sie können die Befehlseingabe so abkürzen, wie in der untenstehenden Tabelle:

Befehl	Abkürzung	Funktion
DIR	DIR	Zeigt das Inhaltsverzeichnis
DIRSYS	DIRS	Zeigt Verz. der SYSTEM-Dateien
ERASE	ERA	Löscht Dateien
RENAME	REN	Dateinamen ändern
TYPE	TYP	Zeigt Text-Datei
USER	USE	Ändert Benutzer-Bereich

Übrigens: In der 2er-Version kommen Sie mit den ausgeschriebenen Namen der Befehle nicht weiter, weil das Betriebssystem sie nicht versteht. Hier können Sie nur mit den Abkürzungen arbeiten.

Genaugenommen gibt es noch ein eingebautes Kommando in CP/M, das aber keinen eigenen Namen besitzt. Sie können dieses Kommando benutzen, um von Laufwerk A: auf Laufwerk B: oder auf irgendein anderes Laufwerk umzuschalten. Wenn Sie statt Laufwerk A lieber Laufwerk B als angemeldetes Laufwerk haben wollen, tippen Sie einfach ein:

B: (RETURN)

und CP/M macht Ihr B-Laufwerk zum angemeldeten Laufwerk. Es sind alle Laufwerksbezeichnungen von A: bis E: erlaubt. Alle anderen Aufforderungen führen bei CP/M zu der Fehlermeldung

```
CP/M Error On F: Invalid Drive
BDOS Function = 14
```

Wenn Sie nun auf dem B-Laufwerk arbeiten, während Ihre CP/M-Systemdiskette im Laufwerk A liegt, können Sie trotzdem auf die CP/M-Programme auf der A-Diskette zugreifen. Dazu geben Sie einfach die Laufwerks-Bezeichnung plus Doppelpunkt vor dem Befehl ein. Das kann so aussehen:

B>A:DIR

Genauso können Sie den Laufwerks-Buchstaben auch vor einen Parameter setzen

B:DIR A:DATEI.XXX

Doch geben Sie nun wieder A: ein, sollten Sie zwei Laufwerke besitzen und auf das zweite Laufwerk umgeschaltet haben. Sie werden sich wundern, denn das Disketteninhaltsverzeichnis auf *einer* Diskette ist schon fein unterteilbar:

IV.3 USER

Dieses eingebaute Programm ermöglicht es Ihnen, ein Speichermedium in 16 Bereiche zu unterteilen. Die Bezeichnung der Bereiche geht dabei von 0 bis 15. So eine Aufteilung kann recht nützlich sein, wenn sich mehrere Benutzer einen Computer teilen und ihre Dateien nicht durcheinander kommen sollen, beispielsweise in Schulen o.ä. Bei Festplattenlaufwerken macht sich diese Möglichkeit der Unterteilung als besonders positiv bemerkbar, da man sonst wegen der immensen Speicherkapazität leicht den Überblick verlieren kann. Eine andere Möglichkeit ist, verschiedene Programme und die dazugehörigen Dateien in jeweils einem anderen Benutzerbereich aufzubewahren.

Zum Beispiel könnten Sie einen Bereich für Ihre Textdateien, einen Bereich für Ihre BASIC-Dateien, einen Bereich für geschäftliche Briefe u.s.w. anlegen.

Durch die Unterteilung in Bereiche ist es auch möglich, daß zwei Dateien mit genau dem gleichen Namen und der gleichen Kennung auf einem Speichermedium stehen. Allerdings - jede Datei in ihrem eigenen Bereich.

IV.3.1 USER-Bereiche bei CP/M 2.2

Wir wollen Sie auch informieren, lieber Leser, wie die Vorgänger des CP/M 3.0, das ist speziell das CP/M 2.2, einige Dinge gehandhabt haben. Dies dient der Transparenz und Sie stehen nicht ganz auf verlorenen Posten, sollten Sie einmal bei einem Freund damit prahlen, CP/M zu beherrschen und dieser hat einen Rechner mit CP/M 2.2.

Aus dem Mehrplatzbetriebssystem MP/M wurde die Möglichkeit der verschiedenen Benutzerbereiche in die Version CP/M 2.2 übernommen. Jeder Bereich arbeitet wie eine selbstständige Diskette. Das heißt: Dateien werden in jedem Benutzerbereich einzeln erstellt und die benötigten Programme müssen alle in diesem Benutzerbereich zur Verfügung stehen. Zwischen den verschiedenen Benutzerbereichen wird mit dem Befehl:

USER n

umgeschaltet. Das "n" steht dabei für eine Ziffer zwischen 0 und 15. Sie können innerhalb eines Bereiches voll arbeiten, ohne die Dateien in anderen Bereichen zu berühren. Zum Beispiel löscht der Befehl:

ERA *.*

alle Dateien, aber nur innerhalb eines Benutzerbereiches. Arbeiten Sie mit zwei Laufwerken und kopieren Dateien von einem zum anderen Laufwerk, erscheinen diese Dateien normalerweise auf dem anderen Laufwerk im selben Benutzerbereich. Wenn Sie also von Bereich 3A mit PIP auf B: kopieren, landen die Dateien in dem Bereich 3B.

Nach jedem Kaltstart von CP/M beginnt das Betriebssystem im Benutzerbereich Null. Möchten Sie in einem anderen arbeiten, müssen Sie zuerst den neuen Benutzerbereich eingeben. Leider sehen Sie am CP/M-Prompt nicht, in welchem Benutzerbereich Sie sich gerade befinden; dies ist unter CP/M 3.0 allerdings erfreulicherweise anders.

IV.3.2 USER-Bereiche bei CP/M 3.0

Doch kommen wir nun zu dem für Sie als CP/M-3.0-Benutzer wichtigeren Kapitel. Eine besondere Stellung nimmt hier der Benutzer-Bereich Null(0) ein. Programme und Dateien, die in diesem Bereich stehen, und zu Systemdateien erklärt wurden, können von jedem Benutzerbereich aus aufgerufen und verwendet werden. Genaueres dazu erfahren Sie in einem späteren Abschnitt. Sobald Sie sich in einem Benutzerbereich befinden und dort eine neue Datei erstellen, merkt sich CP/M automatisch, in welchem Benutzerbereich diese Datei erstellt wurde.

Jedesmal, wenn Sie Ihren Computer einschalten und CP/M geladen haben, sind Sie im Benutzerbereich 0. Wenn Sie in einem anderen Benutzerbereich arbeiten wollen, können Sie den Befehl USER eingeben. Angenommen, Sie wollen in Benutzerbereich Nummer 3, dann geben Sie ein:

```
USER 3
```

Achten Sie darauf, ein Leerzeichen hinter **USER** und der entsprechenden Zahl einzugeben. Sobald CP/M auf den neuen Benutzerbereich umgeschaltet hat, sehen Sie, daß sich etwas auf dem Schirm verändert. Der CP/M-Prompt ist nun nicht mehr einfach ein "A", sondern Sie sehen die Angabe des Benutzerbereiches vor dem A, in unserem Fall also eine "3". Nun wollen Sie sicherlich auch mit dem zweiten Laufwerk arbeiten und dort die Benutzerbereiche umschalten können. Bei anderen CP/M-3.0-Versionen konnte man dies durch Eingabe von

```
USER 3B:
```

bewerkstelligen. Nicht so beim CP/M 3.0 Ihres Commodore 128. Hier müssen Sie diesen Vorgang schon in zwei Eingaben splitten. Beispielsweise zuerst Userauswahl, dann Laufwerksangabe.

```
USER 3
B:
```

Sollten Sie nur das Kommando **USER** eingeben, ohne eine Userkennung anzugeben, so schadet dies auch nichts. CP/M 3.0 erkundigt sich schon bei Ihnen, welchen Userbereich Sie denn nun wirklich wünschen:

```
A>USER
Enter User #: 4
4A>
```

Die Leute von Digital Research haben anscheinend an (fast) alles gedacht, nicht wahr?

Wenn Sie jetzt den Befehl DIR eingeben, um sich das Inhaltsverzeichnis Ihrer Diskette anzusehen, sehen Sie "NO FILE". Vorausgesetzt natürlich, daß Sie in diesem Benutzerbereich noch keine Datei erstellt haben. Sie können von jedem Benutzerbereich aus auf alle eingebauten CP/M-Befehle zugreifen. Um wieder in den Benutzerbereich 0 zu kommen, geben Sie ein:

USER 0

und landen wieder ganz unten im untersten Benutzerbereich. Programme, die hier stehen und für alle Benutzerbereiche verfügbar sein sollen, können Sie mit dem SET-Befehl darauf vorbereiten. Wie das funktioniert, zeige ich Ihnen in einem späteren Abschnitt. Den Wechsel zwischen den Benutzerbereichen können Sie aber noch einfacher gestalten, indem Sie Ihren Änderungswunsch so eingeben:

B1:

Auch die umgekehrte Eingabe, also "1B:" ist möglich. Bei den Benutzer-Bereichen sollte noch eins beachtet werden: Wenn Sie mit PIP eine Datei von "A" nach "B" oder umgekehrt kopieren, wird sie nur im selben Benutzerbereich kopiert.

Fazit: Benutzerbereiche können eine sehr hilfreiche Sache sein, besonders wenn man über eine Harddisk verfügt. Aber auch auf einer Diskette kann man beispielsweise Daten und Programme durch verschiedene Benutzerbereiche klar und sau-

ber voneinander trennen. Dennoch sollte man die Userbereiche zumindest anfangs mit Vorsicht genießen, da sie auch Verwirrung stiften können; beispielsweise findet man eine Datei nicht wieder, weil man den Gebrauch der Userbereiche noch nicht so im Griff hat.

IV.4 DIR

Den Befehl DIR haben Sie schon kennengelernt, als Sie sich im vorherigen Kapitel das Inhaltsverzeichnis Ihrer Disketten ansahen. Das DIR-Kommando gibt es praktisch in zwei "Ausfertigungen": einmal als residentes und ein weiteres mal als transientes Kommando. Wird das DIR-Kommando mit Optionen eingegeben, also mit eckigen Klammern, so wird die Datei DIR.COM geladen (dies ist das transiente DIR) andernfalls begnügt sich CP/M mit dem etwas einfacher gehaltenen residenten, also immer im Speicher befindlichen, DIR und dies reicht auch in den allermeisten Fällen vollkommen aus.

Wir beschäftigen uns jetzt erst einmal mit dem Befehl residenten DIR, also der eigentlich einfacheren Version des Kommandos:

Geben Sie das Kommando DIR ohne irgend etwas anderes dazu ein, erscheinen alle Dateien Ihrer Diskette in diesem speziellen Benutzerbereich. Wenn Sie also DIR im Benutzerbereich 3 befehlen, erscheinen auch nur die Dateien, die im Benutzerbereich 3 abgelegt sind. Manchmal stehen bei CP/M 3.0 in einem Benutzerbereich so viele Dateien, daß längst nicht alle auf einen Bildschirm passen.

In diesem Fall hält die Bildschirmausgabe des Inhaltsverzeichnisses an, bis Sie sich einen Überblick verschafft haben und durch Drücken irgendeiner Taste den Befehl zum

Weitermachen geben. Möchten Sie die Ausgabe des Inhaltsverzeichnisses stoppen, drücken Sie einfach Control C (^C).

Arbeiten Sie gerade auf Laufwerk A, möchten aber sehen, ob eine bestimmte Datei im Laufwerk B vorhanden ist, gibt es zwei Möglichkeiten, das Inhaltsverzeichnis von B zu sehen. Sie tippen ein:

```
B:
DIR
```

und sehen das Inhaltsverzeichnis von Laufwerk B. Die zweite Methode geht schneller. Sie befehlen:

```
DIR B:
```

und sehen ebenfalls das Inhaltsverzeichnis von Laufwerk B auf Ihrem Bildschirm. Zusätzlicher Vorteil der zweiten Version: Sie bleiben auf Laufwerk A. Möchten Sie das Inhaltsverzeichnis Ihrer Diskette lieber ausdrucken, geben Sie ein:

```
DIR B:
```

dann Control P (^P) und RETURN. Nun wird Ihr komplettes Inhaltsverzeichnis auf dem Drucker ausgedruckt. Nach Beendigung des Druckes drücken Sie nochmal Control-P, und der Drucker stellt die Arbeit ein.

IV.4.1 DIR mit Parametern

Die einfachen Funktionen von DIR sind eigentlich nicht besonders aufregend und man erwartet so etwas auch von einem normalen Betriebssystem. Richtig nützlich wird DIR aber, wenn Sie die vielen zusätzlichen Möglichkeiten nutzen, die in diesem leistungsstarken Kommando verborgen sind. Angenommen, Sie besitzen jede Menge Einträge in Ihrem Inhaltsverzeichnis und möchten eine bestimmte Datei heraussuchen. Statt nun alle Dateien einzeln durchzulesen und zu gucken, an welcher Stelle genau sich nun Ihre Datei befindet, geben Sie einfach ein

```
DIR B:TEXT.TXT
```

und bekommen diesen Namen angezeigt, wenn die Datei im Laufwerk B wirklich existiert. Ist die Datei nicht vorhanden, bekommen Sie von CP/M die Fehlermeldung:

```
No File
```

Wenn Sie die erweiterten Funktionen von DIR benutzen wollen, müssen Sie mit dem Programm DIR.COM arbeiten. Das bedeutet aber nichts anderes als: Wenn Sie auf Laufwerk B arbeiten, aber DIR.COM auf Laufwerk A zu finden ist, dürfen Sie nicht vergessen, die Laufwerksbezeichnung vor DIR mit einzugeben. Das sieht zum Beispiel so aus:

```
A:DIR [Optionen]
```

Sie erinnern sich sicher, daß das Sternchen uns eine Menge Tipparbeit beim Suchen abnehmen kann. Mit dem Befehl DIR und der Kombination mit dem Sternchen können Sie sich eine bestimmte Art von Dateien auflisten lassen. Das funktioniert so:

DIR *.COM

Wenn Sie den Befehl so eingeben, werden Sie auf Ihrem Bildschirm alle Dateien sehen, die als Kennung ein COM haben.

IV.4.2 Erweitertes DIR

Sie können in der neueren CP/M-Version genauso gut nach mehreren verschiedenen Dateitypen suchen, d.h. beispielsweise gleichzeitig nach Dateien mit der Kennung .COM und nach Dateien mit der Kennung .SYS. Der entsprechende Befehl sieht dann so aus

DIR *.COM *.SYS

Doch geben Sie diese Kommandozeile ein, so kann CP/M damit nicht allzuviel anfangen und sagt nur ganz verständnislos:

***.SYS?**

Soll heißen: Die Angabe *.SYS ist zuviel oder falsch! Hier hätte ich eigentlich eine Option in eckigen Klammern erwartet, da diese aber fehlt, benutze ich das residente DIR

und nicht das transiente. Und für das residente DIR sind zwei Angaben als Parameter einfach zuviel.

Damit das wirklich funktioniert, müssen Sie von CP/M ausdrücklich verlangen, daß DIR.COM benutzt wird.

Sie erreichen dies, indem Sie die Laufwerksbezeichnung wie A: oder B: vor dem DIR eingeben oder indem Sie eine Option in eckigen Klammern hinter die Datei-Namen schreiben. Die beiden, in diesem Fall, möglichen Optionen heißen [FULL] oder [DIR].

IV.4.3 DIR und seine Optionen

Insgesamt können Sie mit DIR 18 verschiedene Optionen angeben. Damit ist DIR ähnlich leistungsfähig wie PIP und gehört zu den besonders nützlichen CP/M-Programmen. Normalerweise werden Sie wohl selten mehr als ein oder zwei Optionen gleichzeitig eingeben. Die Optionen zu DIR stehen immer in eckigen Klammern.

Wenn mehr als eine Option angegeben wird, werden die beiden durch Kommata oder einen Leerraum getrennt. Ist die Bezeichnung der Option eindeutig, können Sie den Namen der Option auch auf zwei Buchstaben abkürzen. Zur Vereinfachung können Sie die geschlossene eckige Klammer, auch weglassen, wenn es das letzte Zeichen in der Befehlszeile ist.

Beispiele:

```
DIR *.COM *.SYS [FULL]
DIR *.* [NOSORT,SIZE]
DIR *.BAS [USER=5 NOSORT SIZE
```

Wird das transiente DIR verwendetet, so wird zunächst DIR.COM in den TPA geladen. Bedenken Sie, DIR.COM ist immerhin 15 KBytes groß und muß gesucht und geladen werden. Erst wenn dies geschehen ist, erscheint auf dem Bildschirm:

Scanning Directory...

Es werden dann gemäß den angegebenen Parametern und Optionen diejenigen Dateien ausgewählt, die ausgegeben werden sollen. Erst wenn dies geschehen ist und nicht die Option NOSORT angegeben wurde, kann mit dem Sortieren der Dateinamen in alphabetischer Reihenfolge begonnen werden. Auch der Sortiervorgang wird dem Benutzer gemeldet:

Sorting Directory...

ist hier die Meldung vom Betriebssystem. Dann erst erfolgt die Ausgabe der gewünschten Inhaltsverzeichniseinträge, jenachdem auf Drucker oder Bildschirm. Ein typischer Ausdruck sieht beispielsweise so aus:

```
Directory For Drive A:  User  0

    Name      Bytes   Recs   Attributes      Name      Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
CCP      COM     4k     25 SYS RW       COPYSYS COM     1k      3 Dir RW
CPM+     SYS    23k    182 Dir RW       DIR            15k    114 Dir RW
FORMAT   COM     5k     35 Dir RW       HELP            7k     56 Dir RW
HELP     HLP    83k    664 Dir RW       KEYFIG         10k     75 Dir RW
KEYFIG   HLP     9k     72 Dir RW       PIP             9k     68 Dir RW
```

IV.4.4 DIRSYS

Wenn Sie sich unter CP/M 3.0 ein Inhaltsverzeichnis ausgeben lassen, haben Sie vielleicht schon einmal bemerkt, daß am unteren Rand des Bildschirms eine Meldung erscheint, die so aussieht

SYSTEM FILE (S) EXIST

Damit weist Sie CP/M darauf hin, daß außer den aufgelisteten Dateien noch sogenannte Systemdateien auf der Diskette stehen. Allerdings ist auf der Systemdiskette keine Systemdatei. Systemdateien sind solche Dateien, die im Benutzerbereich 0 liegen und von jedem Benutzerbereich aus zugänglich und verwendbar sind. Möchten Sie sich Ihre Systemdateien ansehen, geben Sie das Kommando DIRSYS oder abgekürzt einfach DIRS ein. Unter den aufgelisteten Dateien bekommen Sie dann die Meldung, daß "Nicht-Systemdateien" existieren.

Selbstverständlich haben Sie mit dem Kommando DIRSYS die gleichen Möglichkeiten wie mit dem Befehl DIR, Sie können also auch das Sternchen und die Fragezeichen benutzen, genau wie vorher besprochen.

IV.5 ERASE

Der Platz auf Ihrer Diskette und auch die Zahl der möglichen Einträge in Ihrem Inhaltsverzeichnis pro Diskette sind limitiert. Um genau zu sein: Haben Sie Ihre Diskette einseitig formatiert, so haben Sie 64, bei doppelseitiger Formatierung 128 Eintragungsmöglichkeiten in das Inhaltsverzeichnis. Die

maximale Anzahl an Dateien und die tatsächlich vorhandene wird Ihnen aber auch ausgegeben, wenn Sie das transiente DIR in irgendeiner Form verwenden.

Irgendwann werden Sie keinen Platz mehr auf der Diskette haben, und sind gezwungen eine Datei zu löschen oder auch mehrere. Das Betriebssystem CP/M bietet die Möglichkeit, Dateien mit dem Befehl ERA zu löschen.

Den Befehl geben Sie so ein:

ERASE D:NAME

wobei D für die Laufwerksbezeichnung steht und NAME für den Namen Ihrer Datei. Arbeiten Sie zum Beispiel mit Laufwerk A und wollen dort eine Datei löschen, brauchen Sie die Laufwerksbezeichnung nicht mit einzugeben.

Um mehrere Dateien auf einmal zu löschen, können Sie natürlich auch wieder die beiden Zeichen Sternchen und Fragezeichen benutzen. Allerdings sollten Sie genau überlegen, was Sie tun, wenn Sie diese gewaltigen Befehle benutzen. Nur zu leicht kann es Ihnen nämlich passieren, daß Sie die falschen Dateien löschen und sich nachher vor Wut in den Bauch beißen.

Möchten Sie zum Beispiel alle Dateien löschen, die ein BAK als Kennung besitzen, geben Sie ein:

ERA *.BAK

und sind bei CP/M 2.2 alle Dateien mit der entsprechenden Kennung los. Bei Backup-Dateien, wie hier beschrieben ist ein Irrtum sicherlich nicht so schlimm. Stellen Sie sich aber bitte einmal vor, Sie hätten diesen Befehl gegeben:

```
ERA *.*
```

und dann, ohne nachzudenken RETURN hinterher. Tja, das war's dann auch schon. Ihre Daten sind futsch.

Weil dieser Befehl aber so mächtig ist und vieles auch zerstören kann, haben die CP/M-Erfinder eine klitzekleine Sicherheitsstufe eingebaut. Sobald Sie nämlich mit zwei Sternchen löschen (*.*), werden Sie sicherheitshalber nochmal gefragt, ob Sie das auch ernst meinen:

```
ALL (Y/N)
```

Soll alles gelöscht werden, geben Sie ein Y für Ja und dann ENTER ein. Andernfalls ein N für Nein und die Situation ist gerettet.

IV.5.1 Löschen mit ERA in CP/M 3.0

Wahrscheinlich hat der Chef von DIGITAL RESEARCH aus Versehen einmal selber alle seine Dateien mit seinen Geheimkonten gelöscht. Denn in der neuen CP/M-Version fragt das Programm auf die Eingabe mit einem Joker:

ERA *.BAS

schon nach, ob wirklich das alles gelöscht werden soll:

```
ERASE *.BAS (Y/N)?
```

Bevor Sie hier mit "Y" für Ja antworten, sollten Sie wirklich noch einmal ganz genau überlegen, was Sie tun. Sind Sie ganz sicher, daß der Befehl richtig formuliert ist und daß Sie genau diese Art von Dateien auch löschen wollen? Haben Sie erst mal Ihr "Y" eingetippt, gibt es keinen Weg zurück. Die Dateien sind weg. Achten Sie vor allem auch auf Schreibfehler. Zum Beispiel kann man leicht den Zusatz BAS durch einen Tippfehler als PAS schreiben. Wenn Sie diesen Befehl so losschicken, verschwinden alle PASCAL-Datein die auf Ihrer Diskette stehen...

Um sich vor solch unliebsamen Überraschungen zu schützen, gibt es zwei verschiedene Möglichkeiten. Einmal können Sie Ihre wichtigen Dateien "schreibschützen". Wie das geht, sehen Sie später. Zweitens können Sie an das Kommando ERASE die Option CONFIRM anhängen. CONFIRM heißt soviel wie bestätigen, und Sie können die Option abkürzen zu einem kurzen "C". Wenn Sie den Befehl nun so eingeben:

ERASE *.BAK[C

werden alle "Back-up"-Dateien gelöscht, aber CP/M fragt vor jeder Löschung nach, ob diese Datei auch gelöscht werden soll.

IV.6 Dateinamen ändern mit REN(AME)

Sie haben bisher gesehen, daß wir uns auf Dateien beziehen, indem wir den Dateinamen eingeben oder eintippen. Nun kann es ja sein, daß Ihnen ein Dateiname hinterher nicht mehr gefällt, oder Sie wollen eine Datei auf eine andere Diskette kopieren, auf der eine Datei gleichen Namens bereits besteht. In all diesen Fällen können Sie das eingebaute Kommando REN oder RENAME bei 3.0 (zu deutsch: Umbenennen) benutzen, um Ihre Dateien umzutaufen. Das allgemeine Format für den RENAME-Befehl lautet:

RENAME Neu=Alt

Zuerst kommt also der neue Name der Datei, dann der alte. Zwischen den beiden Namen steht ein Gleichheitszeichen. Wenn die "Umtaufe" auf dem angemeldeten Laufwerk gemacht werden soll, brauchen Sie keine Laufwerksbezeichnung vor die Dateinamen zu schreiben. Möchten Sie einer Datei einen Namen geben, der bereits existiert, werden Sie von CP/M 2.2 gewarnt:

File exists

Das neue CP/M ist da ein wenig hilfreicher und Sie werden gefragt, ob die alte Datei gelöscht werden soll. Diese Meldung sieht beim Commodore-CP/M wie folgt aus:

```
Error: Not renamed, NEU      .EXT file already exists, delete (Y/N)?
```

Beantworten Sie diese Frage mit einem N für Nein, dann ist die Sache für RENAME erledigt. Alles bleibt beim Alten. Geben Sie Y für Yes=Ja ein, dann wird die Datei NEU.EXT gelöscht und die Datei ALT.EXT erhält diesen Namen NEU.EXT. Interessant ist in diesem Zusammenhang, daß der residente Teil von RENAME ein einfaches Umbenennen schafft, tritt aber eine Komplikation aus, in diesem Fall existiert die neue Datei bereits, dann wird erst der transiente Teil von RENAME geladen, bevor es weitergehen kann.

Als Beispiel aber auch eine "einfache" Umtaufe:

```
REN neu.bas=alt.bas
```

Sie können auch bei RENAME 3.0 wieder das Sternchen und die Fragezeichen benutzen. Ein einfaches Beispiel könnte so aussehen

```
RENAME *.TXT=*.BAK
```

Auf diesen Befehl hin werden alle Backup-Dateien dieser Diskette in Dateien mit der Kennung TXT umgewandelt, ganz gleich, was in den Dateien drinsteht. Der Befehl RENAME kümmert sich nämlich überhaupt nicht um den Inhalt von Dateien, sondern nur um die Namensgebung.

IV.7 TYPE

Mit diesem Befehl lassen Sie sich Dateien auf dem Bildschirm anzeigen. Das funktioniert allerdings nur mit Text-Dateien, weil diese mit den Zeichen des ASCII-Zeichensatzes gebildet werden. Andere Dateien, mit der Kennung COM, REL oder ähnlich, enthalten Steuerzeichen, die jede Bildschirmausgabe durcheinander bringen und den Computer veranlassen, sich "aufzuhängen". Den TYPE-Befehl geben Sie so ein

```
TYPE A:DATEI.XXX
```

Wenn sie eine Datei auf einem anderen Laufwerk sehen wollen, stellen Sie eine andere Laufwerksbezeichnung davor.

Zu diesem Befehl gibt es bei CP/M 3.0 eine Option: NO PAGE. Damit lassen Sie den angezeigten Text kontinuierlich über den Bildschirm rollen. Normalerweise stoppt die Bildschirmausgabe nach jeweils 23 Zeilen und geht erst nach einem ENTER weiter.

Wenn Sie NO PAGE mit eingegeben haben, können Sie mit Control S (^S) die Bildschirmausgabe anhalten und mit Control Q (^Q) weiterlaufen lassen. Haben Sie vorher noch ein Control P (^P) eingetippt, wird die Bildschirmausgabe auf Ihrem Drucker aufgelistet.

Übrigens ist auf der zweiten Seite der CP/M-Systemdiskette, also die Seite, die mit "Utilities" betitelt ist, eine Assembler-Quelldatei, die Sie sich sehr gut als Beispiel einmal ausgeben lassen können. Es handelt sich hierbei um die Quelldatei des Kommandos DATEC. Um den Quellcode auf dem Bildschirm zu sehen, geben Sie folgendes Kommando ein:

TYPE DATEC.ASM

und Sie erhalten das Quellprogramm-Listing des DATEC-Kommandos auf den Bildschirm oder Drucker. Da es sich hierbei um eine ASCII-Datei handelt, ist das TYPE-Kommando auch in der Lage, die Datei "vernünftig", also ohne Steuerzeichen auszugeben.

IV.8 Was Sie jetzt schon wissen

* *CP/M besitzt sowohl eingebaute Befehle, wie auch solche, die erst durch ein Programm auf der Diskette ermöglicht werden*

* *Befehle können zusammen mit Parametern oder Optionen eingegeben werden*

* *Sie wissen, wie alle eingebauten Befehle heißen, wie sie wirken und welche Optionen nur mit den transienten Befehlen möglich sind*

V. Transiente Befehle

V.1 Allgemeines

Von den eingebauten residenten Befehlen haben Sie jetzt schon so einiges gelesen. Die wirkliche Macht des Betriebssystems liegt aber in den transienten Befehlen, die als COM-Dateien im Inhaltsverzeichnis zu finden sind. Schon allzuoft haben wir sie erwähnt, diese Wunderkommandos, die bei Eingabe von Diskette in den Arbeitsspeicher geladen werden. Zumeist sind diese Kommandos nicht ohne Grund auf Diskette ausgelagert: sie sind einfach so groß, daß sie zu viel Platz in Anspruch nähmen, würde man sie immer im Hauptspeicher belassen. Und groß sind sie meistens wegen ihrer vielen Möglichkeiten, die sie einem bieten.

Da transiente Kommandos erst im Inhaltsverzeichnis gesucht und dann in den Speicher geladen werden müssen, sind sie nicht ganz so schnell wie ihre residenten Kollegen. Ich bitte Sie, dies bei der Arbeit mit CP/M 3.0 immer zu bedenken.

V.2 Transiente Befehle unter CP/M 3.0

Jetzt kümmern wir uns um wichtige und häufig gebrauchte CP/M 3.0-Befehle. Es handelt sich um transiente Befehle, die Sie alle in Ihrem Inhaltsverzeichnis finden, mit einem COM als Kennung versehen. Sie wissen ja bereits, wie Sie Programme aufrufen: den Namen der Programme ohne die Kennung eintippen und ein RETURN danach geben. Sie haben drei Möglichkeiten die transienten Programme aufzurufen. Einmal, wenn Sie im USER-Bereich 0 sind und dort auch arbeiten wollen, zum Zweiten, wenn Sie die Programme mit dem SET-Befehl für alle

USER-Bereiche zugänglich gemacht haben, drittens können Sie noch den Befehl SETDEF benutzen. Über den letztgenannten Befehl erzähle ich Ihnen später noch mehr.

V.3 SET

Der SET-Befehl übernimmt verschiedene Aufgaben bei CP/M 3.0. Zur Hauptsache wird er wohl verwendet, um verschiedene Datei-Attribute zu setzen. Das sind Zusätze zu den Dateien, die bestimmte Wirkungen haben. Zum Beispiel benutzen Sie den SET-Befehl, um Ihre CP/M-Dateien im USER-Bereich 0 für alle USER-Bereiche zugänglich zu machen. Weiterhin können Sie aber auch Ihre Dateien schützen, indem Sie sagen: "Diese Datei darf nur gelesen werden" oder "Diese Datei ist durch ein Geheimwort geschützt".

Der SET-Befehl beinhaltet aber auch Optionen, die das Inhaltsverzeichnis beeinflußen. Zum Beispiel können Sie sich ein Inhaltsverzeichnis erstellen lassen, in dem jede Datei einen "Zeitstempel" enthält, der darüber Auskunft gibt, wann sie erstmals angelegt und wann zum letzten Mal damit gearbeitet wurde.

Bevor Sie ein Inhaltsverzeichnis mit den "Timestamps", den "Zeitstempeln" einrichten können, müssen Sie das Programm INITDIR.COM laufen lassen. Glauben Sie nicht, die Stempelei sei nur ein Spaß. Sie können diese Einrichtung später nutzen, um alle Dateien die verändert wurden, automatisch auf eine andere Diskette zu sichern.

Wollen Sie aber eine Diskette mit INITDIR bearbeiten, so wird das Disketteninhaltsverzeichnis total umgekrempelt und es wird Platz verbraucht. Es empfiehlt sich aus mehreren Gründen, das Kommando INITDIR auf eine Diskette anzuwenden,

bevor Sie auch nur eine einzige Datei auf derselben befindet. Es soll schon vorgekommen sein, daß bei Komplikationen während INITDIR alle Einträge auf der entsprechenden Diskette verloren gegangen sind.

Präparieren auch Sie eine Diskette mit INITDIR A:, um unsere Beispiele am Rechner ausprobieren zu können.

Damit Sie die vielfältigen Möglichkeiten von SET näher kennenlernen, hier erst einmal eine Übersicht über die verschiedenen Möglichkeiten

OPTION	BEDEUTUNG
DIR	Macht eine SYSTEM-Datei wieder im normalen Inhaltsverzeichnis sichtbar
SYS	Macht eine Datei zur SYSTEM-Datei
RO	Bestimmt, daß eine Datei nur gelesen werden kann
RW	Bestimmt, daß eine Datei gelesen und verändert werden kann
ARCHIV=OFF	Setzt das ARCHIV-Attribut auf "aus". Das bedeutet, daß diese Datei noch nicht gesichert (archiviert) wurde. Das Programm PIP mit der Option AAÜ kann Dateien mit dem Attribut ARCHIV=OFF kopieren. Den PIP-Befehl geben Sie mit den Sternchen für die Dateinamen ein und PIP kopiert alle Datei, die seit dem letzten Kopieren mit PIP und der ÄAÜ-

	Option verändert wurden. Nachdem PIP kopiert hat, setzt es die Datei-Attribute auf ARCHIV=ON.
ARCHIV=ON	Setzt das ARCHIV-Attribut auf "ein". Das bedeutet, daß diese Datei gesichert wurde. Normalerweise ändert PIP mit der Option [A] das Attribut nach dem Sichern von Dateien. Sie können das Attribut auch selber ändern, wenn Sie den SET-Befehl verwenden.
F1=ON/OFF	Schaltet das benutzerdefinierte Datei-Attribut F1 ein oder aus.
F2=ON/OFF	Schaltet das benutzerdefinierte Datei-Attribut F2 ein oder aus.
F3=ON/OFF	Schaltet das benutzerdefinierte Datei-Attribut F3 ein oder aus.
F4=ON/OFF	Schaltet das benutzerdefinierte Datei-Attribut F4 ein oder aus.

Nun haben Sie gesehen, was es alles gibt und sollen jetzt die Anwendung kennenlernen. Sie haben einige wertvolle Programme, zum Beispiel Ihr Textprogramm oder die Datenbank und wollen damit aus allen USER-Bereicher heraus arbeiten. Normalerweise stehen diese Programme, genau wie die CP/M-Programme, im Benutzerbereich 0. Damit Sie auf diese Programme oder auch andere Dateien immer zugreifen können, verwenden Sie den SET-Befehl mit einer oder mehreren Optionen und machen die entsprechende Datei zu einer SYSTEM-Datei. Das sieht so aus:

SET PIP.COM[SYS]

Wollen Sie sichergehen, daß in die Datei PIP.COM (oder jede andere) nichts hineingeschrieben wird bzw. daß diese Datei nicht versehentlich überschrieben wird, geben Sie eine zweite Option mit an:

SET PIP.COM[SYS RO]

Damit ist PIP.COM jetzt eine System-Datei und zudem schreibgeschützt. Wollen Sie eine so gesicherte Datei wieder frei zugänglich machen und das SYSTEM-Attribut aufheben, geben Sie ein:

SET PIP.COM[DIR RW]

Dabei ist es gleichgültig, in welcher Reihenfolge Sie die Optionen eingeben, hauptsache Sie trennen die Optionen logisch mit einem Leerezeichen oder einem Komma etc.

V.4 Laufwerk-Attribute

Sie haben auch die Möglichkeit, ganze Laufwerke so zu definieren, daß von ihnen nur gelesen, aber nicht darauf geschrieben werden kann. Haben Sie ein Laufwerk auf "RO", was für "read only" (nur lesen) steht gesetzt, kann keine Datei mit ERASE gelöscht werden, RENAME funktioniert nicht und PIP kann keine Dateien darauf kopieren.

Sobald Sie ein <Control>-C eingeben (Control-Taste drücken und festhalten und dann zusätzlich noch "C" drücken) ist der RO-Status des Laufwerks wieder aufgehoben. Wenn Sie:

```
SET A:[RO]
```

eingeben, ist das Laufwerk A: schreibgeschützt. Geben Sie RW, (heißt read/write = lesen/schreiben) statt RO ein, kann das Laufwerk wieder voll genutzt werden. Das Setzen des Read-Only-Attributes ist gleichbedeutend mit dem Aufkleben der Schreibschutzmarke; auch dann ist eine Diskette Read-Only. Dieses Attribut kann man freilich nicht durch Betätigen von <Control>-C aufheben.

V.5 Labels

Besonders wenn Sie viele Disketten haben oder mehrere Benutzer an einem Computer arbeiten, ist die Möglichkiet nützlich, den Disketten Namen geben zu können. Dazu benutzen Sie wieder den SET-Befehl, aber mit der Option "NAME=". Die Diskettennamen unterliegen den gleichen Beschränkungen, wie die Dateinamen. Das heißt, Sie können maximal acht Zeichen für den Namen verwenden und drei Zeichen für die Kennung.

Um eine Diskette mit einem Namen zu versehen geben Sie ein:

```
SET B:[NAME=FIBU]
```

Verfügen Sie lediglich über ein Diskettenlaufwerk, so funktioniert das SET-Kommando natürlich nur dann, wenn die Datei SET.COM sich auch auf der Diskette befindet, die Sie mittels SET bearbeiten wollen.

Die Lösung? Na klar: Sie verwenden das virtuelle Laufwerk E: Geben Sie also folgendes ein, es muß sich dabei die Diskettenseite mit der Aufschrift "Utilities" im Laufwerk befinden:

```
SET E: [NAME=FIBU]
```

CP/M holt sich die notwendige Datei SET.COM und fordert Sie dann, die Diskette E: ins Laufwerk A: einzulegen. Haben Sie dieser Aufforderung Folge geleistet, so erscheint folgender Text auf dem Bildschirm:

```
Label for drive E:

Directory      Passwds  Stamp    Stamp    Stamp
Label          Reqd     Create   Access   Update
-------------  -------  -------  -------  -------
E:FIBU    .      off      off      off      off
```

Zu den einzelnen Zuständen, die mit OFF gekennzeichnet sind, werde ich später noch näher eingehen.

Dies wäre beispielsweise eine Bezeichnung für eine Diskette mit Ihrer Finanzbuchhaltung. Wenn Sie schon auf dem Laufwerk B: arbeiten, können Sie die Laufwerksbezeichnung auch weglassen. Bei Benutzung des E-Laufwerkes ist die Angabe der

Laufwerksbezeichnung natürlich unabdingbar. Sie sehen den Namen einer Diskette nicht in Ihrem Inhaltsverzeichnis, wenn Sie es mit DIR aufrufen. Dazu benutzen Sie den Befehl SHOW, näheres wird später erklärt, und dazu die Option "LABEL". Die Option können Sie auf ein einfaches "L" abkürzen und der komplette Befehl sieht so aus:

```
SHOW E:[L]
```

CP/M 3.0 fordert Sie im Laufe der weiteren Beispiele immer wieder auf, Diskette A oder Diskette E, je nachdem, einzulegen. Sie folgen den Aufforderungen am besten und betätigen jeweils die RETURN-Taste, wenn Sie die entsprechende Diskette ins Laufwerk eingelegt haben.

Nun wird die Ausgabe noch etwas detaillierter, als wir sie eben bei dem SET-Kommando erhalten haben:

```
Label for drive E:

Directory     Passwds  Stamp    Stamp    Stamp
Label         Reqd     Create   Access   Update   Label Created   Label Updated
------------  -------  -------  -------  -------  --------------  --------------
E:FIBU    .     off      off      off      off    12/15/82 23:41  12/15/82 23:41
```

Wie bei einem profesionellen *großen* Computer erhält man Angaben über Erstellungsdatum und letztes Update einer Datei. Immer dann, wenn man eine Datei verändert, wird das Updatedatum geändert. Auf diese Weise kann man sich einige Vorteil verschaffen, beispielsweise durch das PIP-Kommando: Man kann mit der entsprechenden Option alle an einem Tag geänderten Dateien updaten lassen. Das Ganze hat natürlich

nur dann Sinn, wenn man mit dem DATE-Kommando bei jeder Sitzung auch Datum und Zeit eingibt, sonst stimmen die übernommenen Daten ja nicht. Wie in unserem Beispiel.

V.6 PASSWORD

Damit Ihnen niemand Ihre schöne Diskettenordnung durcheinander bringt, gibt es in CP/M die Möglichkeit ein Kennwort zu vereinbaren. Sie schützen also den Namen Ihrer Diskette mit einem Kennwort, das nur Sie kennen.

Sobald Sie ein Kennwort für eine Diskette oder eine einzelne Datei eingegeben haben, verlangt CP/M vor jeder Befehlsausführung genau dieses Kennwort. Deshalb legen Sie sich besser eine Datei mit allen Kennworten an, damit Sie selber auf Dauer zurechtkommen. Haben Sie erst einmal ein Kennwort vergessen, gibt es für Sie keine Möglichkeit, mehr an Ihre Daten heranzukommen. Also - Vorsicht.

Um also Ihr Label mit einem Kennwort zu versehen, benutzen Sie den SET-Befehl mit der Option "PASSWORD=Kennwort" oder im umgekehrten Fall "PASSOWRD=<cr>" (<cr> steht für RETURN). Das Kennwort geben Sie so ein:

SET [PASSOWRD=GEHEIM]

oder, um ein Kennwort aufzuheben:

SET [PASSWORD=<cr> (<cr> = RETURN)

Natürlich können Sie das Passwort nur dann aufheben, wenn Sie es auch eingeben können - das ist ja wohl klar. Lassen Sie uns einmal das DIR.COM auf der CP/M-Sicherheitsdiskette mit einem Passwort versehen.

```
SET E:DIR.COM[PASSWORD=DUDU]
```

Wollen Sie ein Passwort aufheben, so fragt CP/M:

```
Password?
```

Wundern Sie sich nicht - aber wenn Sie auf den Tasten "rumhacken", dann erscheint kein einziger Buchstabe auf dem Bildschirm. Aus Sicherheitsgründen versteht sich: Denn wenn jeder sehen könnte, was Sie eingeben, so wäre es nicht mehr so weit her mit der Sicherheit. Haben Sie ein falsches Passwort eingegeben, so wird das Kommando nicht ausgegeben und es erscheint die Fehlermeldung:

```
ERROR: Wrong Password
```

Nochmal: Denken Sie daran, Ihr Kennwort irgendwo aufzubewahren. Wenn Sie es nicht mehr wissen, haben Sie keinen Zugriff auf Ihre Daten auf dieser Diskette.

V.7 PROTECT

Sie können nicht nur das Label Ihrer Diskette mit einem Kennwort versehen, sondern auch einzelne Dateien oder sogar CP/M-Befehle. So können Sie sicherstellen, daß wirklich kein Unbefugter an Ihren Daten herumspielt oder Einblick in Dinge bekommt, die ihn nichts angehen.

Bevor Sie nun eine einzelne Datei oder einen Befehl schützen können, müssen Sie den PROTECT-Modus einschalten. Das geschieht einfach dadurch, daß Sie folgenden Befehl eingeben:

SET E:[PROTECT=ON]

oder entsprechend umgekehrt:

SET E:[PROTECT=OFF]

Wenn Sie **SET E:[PROTECT=ON]** eingegeben haben, ist das Betriebssystem darauf eingestellt, einzelne Dateien zu schützen. Wichtig ist, daß es verschiedene Formen des Schützens gibt, die Sie vor dem einfachen Ausprobieren unbedingt kennenlernen sollten. Dazu mehr im folgenden Kapitel V.8.

V.8 Datei-PASSWORD

Um eine Datei unter Ihren persönlichen Schutz zu stellen, gehen Sie im Prinzip genauso vor, wie vorher beim Schützen von Diskettennamen.

Sie können auch mehrere Dateien gleichzeitig unter den Schutz eines Kennwortes stellen, wenn Sie die Möglichkeit der Sternchen (*) benutzen. Dann gilt für alle Dateien, auf welche die Bedingung zutrifft, das gleiche Kennwort. Die Befehlseingabe sieht so aus:

```
SET E:DIR.COM[PASSWORD=DUDU]
```

oder mit "*" formuliert:

```
SET E:*.COM[PASSWORD=Kennwort]
```

In dem ersten hier gezeigten Beispiel ist die Datei DIR.COM durch das Kennwort "DUDU" geschützt, im zweiten Beispiel bekommen alle COM-Dateien das Kennwort "DUDU".

Zusätzlich läßt sich noch festlegen, *wie* die Dateien geschützt werden sollen; wir haben es eben schon kurz erwähnt, daß es hier Unterschiede gibt. Dazu gibt es folgende Möglichkeiten:

READ	Das Kennwort wird benötigt, um Dateien zu lesen, kopieren, beschreiben, löschen oder umzubenennen

WRITE	Das Kennwort ist nötig, um Dateien zu beschreiben, löschen oder um einen neuen Namen zu geben
DELETE	Das Kennwort ist nur zum Löschen notwendig
NONE	Es gibt kein Kennwort. Wenn bereits ein Kennwort vorhanden ist, kann es mit dieser Option gelöscht werden

Eingegeben werden diese Optionen wie folgt. Erfolgt keine Angabe über die Form des Schutzes, so wird READ angenommen.

```
SET E:TEXT.TXT[PROTECT=DELETE]
```

Damit haben Sie die Datei TEXT.TXT vor versehentlichem Löschen geschützt.

Um Mißverständnissen vorzubeugen: Man kann (leider) beispielsweise keine COM-Dateien auf diese Weise vor dem Ausführen schützen. Wird eine COM-Datei mit einem Passwort und mit Schutzart READ belegt, so erhalten Sie beim Versuch der Ausführung des Kommando die Meldung:

```
CP/M Error On A: Password Error
BDOS Function = 15  File = DIR      .COM
```

Es erfolgt also keine Abfrage des Passwortes *vor* der Ausführung des Kommandos, das Kommando ist einfach gesperrt. Erst wenn man das Passwort aufheben will, dann fragt CP/M nach dem eingegebenen Passwort, das dann natürlich korrekt

eingegeben werden muß, da sonst der Schutz nicht aufgehoben oder verändert werden kann. Auf diese Weise könnten Sie beispielsweise selbsterstellte COM- oder SUB-Dateien vor dem Ausführen unberechtigter Personen schützen, Sie müßten vor dem Ausführen lediglich mittels des SET-Kommandos den Schutz wieder aufheben.

V.9 TIME STAMP

Mit der Einrichtung des Zeit-Stempels, können Sie sich einen Überblick darüber verschaffen, wie oft Sie eine bestimmte Datei benutzen, wann Sie eine Datei erstellt und geändert haben und Ihre Sicherungsdateien managen. Ein Zeitstempel ist wie die Stechuhr in einer Firma, womit alle Arbeitszeit-Daten festgehalten werden.

Um Zeit und Datum zu jeder Datei festhalten zu können, müssen Sie zuerst das Programm INITDIR benutzt haben. Das Verwalten der Zeiten ist nur möglich, wenn das Inhaltsverzeichnis in einer besonderen Weise organisiert ist. Das erreichen Sie mit INITDIR. Bitte nehmen Sie zur Kenntnis, daß beim Einrichten einer Diskette mit INITDIR Platz im Inhaltsverzeichnis verloren geht, der für die zu speichernden Zeitangaben benötigt wird. Auf eine Diskette passen also bei weitem nicht mehr 64 Dateien, eher die Hälfte!

Sie haben auf einer Diskette, die Sie mit INITDIR bearbeitet haben, drei verschiedene Optionen zur Verfügung, die Sie mit dem SET-Kommando ein- und natürlich auch ausschalten können. Dabei ist zu beachten, daß nicht alle Optionen parallel eingeschaltet sein können.

CREATE=ON	Schaltet die Zeit-und Datummarkierung für die Dateien auf einem bestimmten Laufwerk ein.
ACCESS=ON	Schaltet die Markierung des letzten Zugriffs auf eine Datei ein. Sie können aber nur CREATE oder ACCESS einschalten. Wenn ACCESS auf einem Laufwerk gewünscht wird, für das bereits CREATE gewählt wurde, wird CREATE automatisch ausgeschaltet.
UPDATE=ON	Sorgt für eine Markierung der Dateien, jedesmal wenn sie neu bearbeitet wurden. Das Anzeigen in einem Inhaltsverzeichnis hat keine Wirkung auf diesen Zeit-Stempel.

Wenn Sie UPDATE und CREATE als Ihre Optionen wählen, sollten Sie beachten, daß bei einer Bearbeitung der Dateien beide Zeitstempel aktualisiert werden.

Das liegt daran, daß beim Bearbeiten einer Datei eine neue eingerichtet wird, während die alte zur BAK-Datei (Sicherheits-Kopie) wird.

Ihre Computer-Stechuhr bringen Sie so zum Laufen:

```
SET E:[ACCESS=ON]
```

Um sich das Ergebnis dieser Operation anzusehen, gibt man den Befehl:

DIR[FULL]

und bekommt dies:

```
Directory for Drive B:

Name      Bytes   Recs   Attributes   Prot   Update          Access
-------   -----   ----   ----------   ----   ------------    -------------
TEXT.TXT    5K      38   DIR    RW    NONE                   04/01/85 17:31
FIBU       20K     152   SYS    RO    NONE                   04/01/85 09:10
```

Unter der Überschrift ACCESS können Sie ablesen, wann Sie die einzelnen Dateien zuletzt benutzt haben. Der Befehl für zwei Eintragungen in das Inhaltsverzeichnis sieht so aus:

SET E:[CREATE=ON,UPDATE=ON]

Ein Inhaltsverzeichnis hätte dann (ungefähr) diesen Aufbau:

```
Directory for Drive B:

Name      Bytes   Recs   Attributes   Prot   Update          Create
-------   -----   ----   ----------   ----   ------------    -------------
TEXT.TXT    5K      38   DIR    RW    NONE   04/17/85 10:00  01/01/85 09:00
FIBU       20K     152   SYS    RO    NONE   04/17/85 16:43  01/01/82 19:21
```

V.10 SETDEF

Wie Sie wissen, sucht CP/M nach Eingabe eines Kommandos auf der CP/M-Diskette immer nach der entsprechenden Datei, die CP/M ausführen könnte. Dies können COM- aber beispielsweise auch SUB-Dateien sein. CP/M sucht aber immer auf dem angemeldeten Laufwerk, es sei denn, Sie haben ein A:, B: oder wie auch immer dem Kommando unmittelbar vorangestellt. Gehen wir einmal ein Beispiel durch, wann SETDEF interessant werden könnte: Sie arbeiten beispielsweise auf dem Laufwerk B:, haben aber alle CP/M-Dateien oder andere Programme auf dem Laufwerk A: gespeichert. Mit SETDEF teilen Sie CP/M nun einen Suchpfad mit, damit es die gewünschten Programme lädt, ohne daß Sie jedesmal die Laufwerksbezeichnung mit eingeben müssen.

Wenn Sie einfach:

SETDEF

eingeben, erhalten Sie Informationen darüber, wie Ihr Suchpfad derzeit aussieht, welches Laufwerk für temporäre Dateien verwendet wird und nach welchen Datei-Typem gesucht wird. Dies sieht auf dem Bildschirm etwa so aus:

```
Drive Search Path:
1st Drive             - Default

Search Order          - COM
Temporary Drive       - Default
Console Page Mode     - On
Progran Name Display  - Off
```

Was bedeuten aber nun die einzelnen Informationen? Darauf wollen wir nun gleich eingehen. Ein Beispiel, um den Suchpfad zu verändern, wäre folgendes Kommando:

SETDEF A:

Dann sucht CP/M immer auf dem A:-Laufwerk nach den gewünschten Dateien, selbst wenn Sie auf Laufwerk B: arbeiten. Wenn Sie den Befehl so erweitern:

SETDEF A:,*

sucht CP/M zuerst auf dem A:-Laufwerk und dann auf dem angemeldeten nach den Dateien. Der Stern steht für das gerade angemeldete Laufwerk. Sie erhalten nach der Eingabe eines SETDEF-Kommandos auch einen Auszug aus der Informationsliste.

```
Drive Search Path:
1st Drive          - A:
2nd Drive          - Default
```

werden Sie nun belehrt. Übrigens können Sie sich die Informationen selbstverständlich auch auf dem Drucker protokollieren lassen, indem Sie die <CTRL>-P-Tastenkombination betätigen.

Sollen temporäre Dateien, also Zwischendateien, wie PIP zum Beispiel welche erstellt, auf ein besonderes Laufwerk, geben Sie

SETDEF [TEMPORARY=C:]

ein und die Zwischendateien werden auf ein drittes Laufwerk oder auch in die RAM-Disk geschrieben. Die temporären Dateien erkennt CP/M daran, daß diese die Dateiextension $$$ erhalten, aber dies haben wir ja auch bereits schon einmal erwähnt.

Gesucht werden kann nur nach Dateien, die eine COM- oder SUB- Kennung haben. Normaleinstellung bei CP/M 3.0 ist die Suche nach COM-Dateien, was aber so:

SETDEF [ORDER=(SUB,COM)]

geändert werden kann. SUB-Dateien sind solche, die mit dem Befehl SUBMIT aufgerufen werden und hintereinander ausführbare Befehle enthalten. Dazu kommen wir später aber natürlich noch.

V.11 SHOW

Wir kommen jetzt zu einem Befehl, der Ihnen viele Informationen über den Platz auf Ihren Disketten, die Namen Ihrer Disketten und über die Anzahl der Dateien pro USER geben kann. Wenn Sie einfach:

SHOW

eintippen, sehen Sie die Attribute aller Laufwerke und den noch verbleibenden Platz auf jedem Laufwerk. Es muß sich dabei die Utilitiediskette im Laufwerk befinden.

A: RW, Space: 11k
E: RW, Space: 2k

Geben Sie SHOW mit einer Laufwerksbezeichnung dahinter an, bekommen Sie nur die statistischen Angaben zu diesem einen Laufwerk.

Mit SHOW können Sie sich aber auch, wie schon weiter oben erwähnt, die Labels Ihrer Disketten anzeigen lassen:

SHOW A:[LABEL]

wobei Sie LABEL auch auf ein schlichtes "L" abkürzen dürfen. Auf dem Bildschirm sehen Sie dann:

```
Label for drive A:

Directory     Passwds  Stamp   Stamp
Label         Reqd     Create  Update  Label Created    Label Updated
-----------   -------  ------  ------  --------------   --------------
FIBU     .COM    off      off     off  04/17/85 11:41   04/17/85 11:41
```

Mit einer weiteren Option zum SHOW-Befehl können Sie sich anzeigen lassen, welche USER-Bereiche auf Ihrer Diskette genutzt werden und wieviele Dateien zu jedem Bereich gehören. Diese Option heißt (leicht zu merken) auch **USER**.

Zusätzlich bekommen Sie noch die Anzahl der freien Inhaltsverzeichnis-Einträge angezeigt:

```
A: Active User :    0
A: Active Files:    0   2  11  12
A: # of files  :   22   6   1   1

A: Number of free directory entries:        24
```

Wenn Sie einfach:

SHOW A:[DIR]

eingeben, bekommen Sie lediglich die Zahl der noch freien Einträge angezeigt, d.h. Sie bekommen die letzte hier aufgeführte Zeile auf dem Bildschirm.

Für CP/M-2.2-Benutzer und -Umsteiger: Der SHOW-Befehl ist stark artverwandt mit dem CP/M-2.2-Kommando STAT, das noch einige andere CP/M-3.0-Kommandos beinhaltet. Von CP/M 2.2 sind also in der Entwicklung zu CP/M 3.0 die Funktionen vom Kommando STAT auf mehrere neue CP/M-3.0-Kommandos verteilt worden. Dabei sind diese neuen Befehle leistungsstärker und bedienerfreundlicher geworden - es ist also nicht zu Ihrem Nachteil geschehen.

V.12 SUBMIT

Bisher haben Sie alle Befehle schön brav über die Tastatur eingegeben. Das ist ja alles ganz logisch und auch gut so, aber wenn Sie immer wieder die gleichen Befehle eingeben sollen, um immer das Gleiche zu erreichen, wird's auf die Dauer doch lästig. CP/M kann Sie von dieser Last befreien.

Mit dem Befehl SUBMIT können Sie eine ganze Reihe von Befehlen abarbeiten lassen, die in einer Datei stehen und wie Eingaben über die Tastatur verarbeitet werden. Eine Datei, in der solche Befehlsfolgen stehen, wird mit der Kennung SUB versehen und kann dann von dem Programm SUBMIT gelesen und ausgeführt werden.

Wenn Ihr Computer zum Beispiel über keine eingebaute und durch eine Batterie am Laufen gehaltene Uhr verfügt, und das ist beim Commodore 128 wohl zu 99% der Fall, so müssen Sie die Zeit und das Datum jedesmal nach dem Start des Computers neu eingeben. Wollen Sie sich selber zwingen, diesen Eintrag jedesmal vorzunehmen, schon damit die Zeitstempel richtig und kontinuierlich gesetzt werden, verwenden Sie die Datei PROFILE.SUB. Diese Datei PROFILE.SUB wird von CP/M nach jeden Booten oder Reset gesucht und wird diese Datei gefunden, so werden die in der Datei vorhandenen Kommandos als SUBMIT-Datei abgearbeitet. Man kann dieses PROFILE.SUB mit den AUTOEXEC.BAT-Dateien auf den IBM-PC und -kompatiblen vergleichen.

Wenn Sie hier also vorgeben, daß Zeit und Datum eingetragen werden sollen, kommen Sie um die Eingabe nicht herum. Die Datei schreiben Sie sich so:

```
B:DATE SET
```

mit Ihrem Textprozessor und taufen diese Datei PROFILE.SUB. Natürlich können Sie auch ein anderes Laufwerk angeben. Das hängt alleine davon ab, auf welchem Laufwerk die Datei DATE.COM zu finden ist. Vergessen Sie die Kennung SUB, führt der SUBMIT-Befehl gar nichts aus.

Übrigens: wollen Sie in einer Datei über Tastatur etwas eingeben, in unserem Beispiel wollen wir eine Zeile eingeben, so ist dies beispielsweise mittels dem Editor **ED** möglich. Allerdings macht das Arbeiten mit diesem Editor überhaupt keinen Spaß, es ist eher eine Qual, damit zu arbeiten. Es gibt aber noch eine andere Möglichkeit, Daten über die Tastatur in eine Datei zu schreiben, Sie können diese Datei dann allerdings nicht editieren. Unter der Benutzung des PIP-Kommandos ist dies möglich:

```
PIP PROFILE.SUB=CON:
```

Warten Sie, bis sich das Diskettenlaufwerk nicht mehr bewegt. Dann können Sie die Zeile eingeben und dann RETURN drücken. Wollen Sie mehrere Zeilen eingeben, so schreiben Sie fleißig weiter. Als letzte Zeile müssen Sie <CTRL>-Z eingeben.

Eine SUB-Datei kann CP/M-Befehle, verschachtelte SUBMIT-Befehle und Eingabedaten für ein Programm oder einen CP/M-Befehl enthalten. Wenn Sie etwas darüber nachdenken werden Sie feststellen, daß damit eine ganze Menge zu machen ist.

Sie können in einer SUB-Datei auch mit allgemeinen Anweisungen arbeiten, die entsprechend Ihren Eingaben dann verwendet werden. Diese allgemeinen Anweisungen heißen Parameter und werden durch ein Dollarzeichen ($) dargestellt. Verwenden können Sie Parameter von $1 bis $9.

Nehmen wir an, Sie schreiben sich eine Datei, die so aussieht:

```
ERA $1.BAK
DIR *.$2
```

und von mir aus DIR.SUB heißen soll. Mit dieser Befehlsfolge in der Datei löschen Sie zuerst alle Datei eines bestimmten Namens und mit der Kennung .BAK. Dann lassen Sie sich alle Dateien mit einer bestimmten Kennung zeigen. Um dorthin zu gelangen geben Sie ein:

```
SUBMIT DIR TEXT COM
```

DIR ist hier der Name der Submitdatei, TEXT und COM sind die zu übergebenden Parameter für $1 und $2. Zuerst werden alle Dateien mit der Extension BAK und der Dateikennung $1 gelöscht. Anschließend bekommen Sie eine Übersicht über alle Dateien mit der Kennung COM. Natürlich muß sich die Datei SUBMIT.COM sowie die SUB-Datei auf derselben Diskettenseite befinden. Andernfalls muß man wieder mit dem virtuellen Laufwerk E: arbeiten. Die "übersetzte" SUBMIT-Datei, das ist also praktisch die Submitdatei mit eingesetzten Parametern, sieht so aus:

```
ERA TEXT.BAK
DIR *.COM
```

und CP/M führt diese Befehle genauso aus, als wären sie über die Tastatur eingegeben worden.

Geben Sie weniger Parameter ein, als in der SUBMIT-Datei vorgesehen sind, werden die übrigen Parameter nicht beachtet. Geben Sie mehr Parameter ein, als in der SUBMIT-Datei stehen, werden die überzähligen ebenfalls nicht beachtet. Möchten Sie ein Dollarzeichen in einem Befehl innerhalb einer SUBMIT-Datei stehen haben, geben Sie einfach zwei Dollarzeichen hintereinander ein ($$) und Sie bekommen, was Sie wünschen.

Eine SUBMIT-Datei kann aber nicht nur einzeilige Befehle enthalten, sondern auch Befehlseingaben für Programme. Sie können also mit einer Befehlszeile ein Programm aufrufen und mit der nächsten Befehle an das aufgerufene Programm weitergeben. Ein Beispiel:

```
PIP
<B:=A:*.COM
<
DIR *.COM
```

Hier sehen Sie eine kleine SUBMIT-Datei, die etwas Neues enthält. Mit der ersten Zeile rufen Sie PIP.COM auf, geben in der zweiten Zeile den Befehl, alle COM-Dateien auf Laufwerk B: zu kopieren, steigen dann aus PIP aus und sehen sich anschließend das Inhaltsverzeichnis an. Alle Befehle, die direkt an ein Programm gehen, werden mit dem "Kleiner als"- Zeichen (<) kenntlich gemacht. Geben Sie es ohne irgendeinen Zusatz ein, wie in der dritten Zeile, bedeutet es ein RETURN, was wiederum, in diesem Fall, PIP beendet. Übrigens haben es hier die Benutzer von CP/M 2.2 nicht ganz so einfach: Will man Parameter an ein Programm übergeben, so reicht das Kleinerzeichen nicht aus, hier muß man sich zusätzlich des Kommandos XSUB (oder ähnlichen Namens) bedienen, um denselben Erfolg zu erzielen. CP/M 3.0 auf dem Commodore 128 nimmt uns diese Arbeit ab.

Der SUBMIT-Befehl ist aber nicht nur Spielerei, sondern kann durchaus nützlich angewendet werden. Sie können damit zum Beispiel beliebig viele Datein hintereinander drucken lassen und in der Zwischenzeit etwas essen gehen. Die Kommando-Datei schreiben Sie einfach so:

```
PIP LST:=DATEI1
PIP LST:=DATEI2
PIP LST:=DATEI3
PIP LST:=DATEI4
usw.
```

oder, mit noch weniger Aufwand kommen Sie so hin:

```
PIP LST:=DATEI?
```

Sie nennen die Datei dann SAMMELDR.SUB und geben, bevor Sie weggehen, ein:

```
SUBMIT SAMMELDR (RETURN)
```

Dabei können die Dateien natürlich auch auf verschieden Laufwerken stehen. Sie müssen nur den Laufwerksbuchstaben vor den Dateinamen schreiben und SUBMIT sucht sich den Rest zusammen.

Sie könnten sich auch eine Kommandodatei schreiben, die alle Dateien einer Festplatte über alle USER-Bereiche hinweg mit DIR auflistet, die Bildschirmausgabe dann aber in eine neue Datei mit dem Namen Inhalt schreibt. In dieser Datei suchen

Sie eine bestimmte Datei dann mit dem Suchbefehl Ihres Textprogramms oder drucken die Dateienliste aus.

Eigentlich sind dem Anwender mit dem SUBMIT-Befehl keinerlei Grenzen gesetzt, lediglich die Phantasie und einige kleine Programmierkniffe sind erforderlich.

V.13 Der HELP-Befehl

CP/M 3.0 ist mit seinen vielen Befehlen nicht mehr so leicht zu erlernen und beherrschen wie seine Vorgänger. Besonders die Optionen werden Sie sicherlich nicht so schnell alle im Kopf behalten. Glücklicherweise bietet das Betriebssystem seine Hilfe sozusagen "auf Knopfdruck" an. Sie müssen nur auf englisch "Hilfe" schreien und schon kommt, was Sie brauchen. Mit dem Schreien war natürlich das Eintippen in den Computer gemeint. Auch viele neue komfortable Programme bieten Hilfen im Sinne von sogenannten *Helpscreens* an. Man braucht dem Rechner (dem Programm) lediglich mitzuteilen, daß man Hilfe benötigt. Ferner muß man noch mitteilen, wo diese Hilfe notwendig geworden ist.

Wir wollen nun diese freundliche Hilfe von CP/M 3.0 einmal in Anspruch nehmen. Weil wir noch nicht genau wissen, worüber wir nicht so genau bescheid wissen, geben wir einfach das Wort HELP=Hilfe ein:

```
HELP
```

Und wir bekommen ein Menü mit den möglichen Hilfestellungen auf dem Bildschirm angeboten. Da nun suchen Sie sich einen Unterpunkt aus und werden ausführlicher informiert.

Übrigens: Die Tastatur des Commodore 128 verfügt über eine HELP-Taste, die im BASIC-Betrieb im Falle eines Fehlers im Programm diesen Fehler auf dem Bildschirm unterstreicht. Unter CP/M 3.0 erscheint nach Betätigen dieser Taste das Wort HELP auf dem Bildschirm, allerdimgs ohne automatischen Wagenrücklauf, was den Vorteil hat, daß Sie nötigenfalls noch ein Stichwort angeben können, wenn Sie es für notwendig halten. Auf dem Bildschirm erscheint folgender Text, wenn Sie die Hilfe ohne Stichwort in Anspruch nehmen:

```
HELP UTILITY V1.1

At "HELP>" enter topic [,subtopic]...

EXAMPLE:  HELP> DIR EXAMPLES

Topics available:

C128_MODE   COMMANDS    CNTRLCHARS  COPYSYS     DATE        DEVICE
DIR         DUMP        ED          ERASE       FILESPEC    GENCOM
GET         HELP        HEXCOM      INITDIR     KEYFIG      LIB
LINK        MAC         PATCH       PIP (COPY)  PUT         RENAME
RMAC        SAVE        SET         SETDEF      SHOW        SID
SUBMIT      TYPE        USER        XREF

HELP>
```

Sie können die Unterpunkte auch direkt aufrufen, indem Sie zum Beispiel eingeben:

HELP SETDEF

damit kommen Sie direkt an die Hilfsinformationen heran. Doch leider ist nicht alles Gold was glänzt und englisch nicht jedermannes Muttersprache. Selbstverständlich gehen unsere Freunde die amerikanischen Programmierer davon aus, daß die Welt englisch spricht. Nach langem Hin und Her und nach vielen durchkauten Kaugummis haben sich die CP/M-Erfinder aber durchgerungen, eine Möglichkeit für Nicht-Amerikaner zu schaffen, die Hilfsinformationen auch in anderen Sprachen auf den Schirm zu holen.

Das Hilfe-Programm besteht aus den Dateien HELP.COM und HELP.HLP. Um hier etwas zu ändern, rufen Sie die Datei HELP.COM auf und geben Sie als Option EXTRACT ein:

HELP [EXTRACT]

Sie können das EXTRACT auch auf ein kurzes "E" abkürzen. Das HELP-Programm erstellt dann aus der Datei HELP.HLP eine neue Datei mit dem Namen HELP.DAT, die Sie mit Ihrem Textverarbeitungssystem nach Ihren Vorstellungen und in Ihrer Sprache umändern können. Durch die Option EXTRACT sind die Helptexte nicht deutsch geworden, aber es sind die Grundlagen zum Editieren der Topics geschaffen worden.

Um neue Hilfstexte einzugeben, müssen Sie folgendes beachten:

Jedes Stichwort muß mit drei Schrägstrichen (///) und einer Nummer beginnen. Die Nummer gibt die Hilfsstufe des Stichwortes an. Zum Beispiel:

///1DIR (RETURN)

///2OPTIONEN (RETURN)

///3PARAMETER (RETURN)

///4BEISPIELE (RETURN)

Haben Sie alles geändert oder vielleicht eine Hilfe für ein wenig benutztes Programm installiert, speichern Sie die Datei ab und rufen wieder HELP.COM auf, allerdings diesmal mit der Option CREATE, abgekürzt "C". Daraufhin wird eine neue HELP.HLP-Datei erstellt, die Ihre Änderungen enthält.

Wer weiß, vielleicht gibt es schon bald einen Anbieter für ins Deutsche übersetzte Hilfstexte. Ein Bedarf ist bestimmt da.

V.14 Was Sie jetzt schon wissen

* *Sie haben den gewichtigen Befehl STAT von CP/M 2.2 kennegelernt und können sich damit die verschiedenen Informationen über Ihr System holen oder diese ändern*

* *Sie haben die transienten Befehle von CP/M 3.0 kennengelernt*

* *Sie kennen mittlerweile eine große Anzahl von Optionen, mit denen Sie die Wirkung der transienten Befehle verändern oder erweitern können*

* *Sie wissen, daß Sie Dateien mit einem Label versehen können*

* *Sie haben gesehen, wie man eine ganze Diskette, eine Datei oder bestimmte CP/M-Befehle vor unbefugtem Zugriff schützt*

* *Ihnen ist auch klar, wie Sie Ihre Dateien mit einem Stempel für die Zeit und das Datum präparieren können*

* *Sie haben die Möglichkeiten des Suchpfades für Programme kennengelernt*

* *Sie wissen nun, wie Sie die System-Daten einer Diskette mit SHOW anzeigen lassen können*

* *Die Datei PROFILE.SUB können Sie verwenden, um die Startroutine Ihres Computers zu automatisieren*

VI. Alles über PIP

Von dem Wunderding namens PIP haben Sie jetzt schon einige Male gehört. Damit Sie nicht glauben, ich übertreibe hier schamlos, zähle ich Ihnen schnell auf, was PIP alles kann

* *Eine einzelne Datei von einer Diskette auf eine andere übertragen*
* *Eine Gruppe von Dateien von einer Diskette auf eine andere übertragen*
* *Eine Datei kopieren und mit einem anderen Namen versehen*
* *Text zum Ausdrucken formatieren*
* *Zu lange Zeile kürzen*
* *Eine Sammlung von Datei auf einen Befehl hin drucken*
* *Mehrere Dateien zu einer zusammenkopieren*
* *Ein Stück aus einer Textdatei herausholen*
* *Kleinbuchstaben in große verwandeln und umgekehrt*
* *Das achte oder Parity-Bit wieder auf Null setzen*
* *Eine Datei mit Zeilennummern versehen*
* *Eine Datei während des Übertragens auf dem Bildschirm anzeigen*
* *Systemdateien übertragen*

* *Dateien von einem USER-Bereich zu einem anderen kopieren*

* *Neu erstellte oder geänderte Dateien automatisch sichern*

Na, ist das genug? Auf jeden Fall Grund genug, sich mit diesem Programm ausführlich zu beschäftigen. Lesen Sie dieses Kapitel und das Handbuch lieber drei als zweimal und machen Sie sich klar, was PIP alles für Sie erledigen kann.

VI.1 Diskette kopieren

Nahezu jedes Handbuch zu irgendeinem Programm beginnt mit dem Rat, die Originaldiskette erst einmal mit PIP auf eine neue, formatierte Diskette zu überspielen und dann nur noch diese Kopie zu benutzen. In einem früheren Kapitel hab' ich Sie auch darauf hingewiesen, und gezeigt wie es geht. Weil die Denkweise von PIP etwas verschieden von der normalen Denkweise einfacher Menschen ist, hier noch mal das einfache Beispiel. Sie kopieren von einer Diskette auf eine andere so:

```
PIP
*B:TEXT.TXT=A:TEXT.TXT
```

Damit erreichen Sie, daß die Datei TEXT.TXT von der Diskette im Laufwerk A: unter dem gleichen Namen auf dem Laufwerk B: nochmal erzeugt wird. Ihr Original bleibt wie es ist, Sie haben nur eine Kopie auf einem anderen Laufwerk.

Um eine komplette Diskette zu kopieren, geben Sie ein

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

damit schaufeln Sie alle Dateien von A: nach B:. Sollten Sie kein B:-Laufwerk haben, so erreichen Sie dieses Kopieren aller Dateien auch durch Eingabe des Kommandos

```
PIP E:=A:*.*
```

allerdings bei einigen Mehraufwand, da Sie die Disketten immer wieder austauschen müssen. Anders ist ein Kopieren von Diskette zu Diskette auch garnicht möglich. Anders als bei anderen CP/M-System brauchen Sie beim Commodore 128 CP/M das System nicht speziell mit COPYSYS oder SYSGEN zu kopieren; wie man das System kopiert, haben wir ja bereits besprochen.

Haben Sie beide Seiten Ihrer Originaldiskette kopiert, besitzen Sie eine voll verwendbare, das heißt auch "bootbare", Kopie Ihres Originals. Sie können die Kopie in's Laufwerk schieben und den Computer starten. Um die oben vorgestellte Prozedur zu vereinfachen und zu beschleunigen gibt's zwei Möglichkeiten.

Der Befehl PIP kann auch mit einer ganzen Reihe von Optionen versehen werden, die die unterschiedlichsten Auswirkungen haben. Eine Option heißt "V" und steht für "verify", was übersetzt "nachprüfen" bedeutet. Sobald PIP mit dieser Option arbeitet, prüft es nach jedem Kopiervorgang genau nach, ob die neue Datei auch wirklich genau der Ursprungsdatei entspricht. Die Optionen müssen wieder mit eckigen Klammern eingegeben werden. Um die Datei TEXT.TXT zu übertragen und gleichzeitig zu überprüfen geben Sie ein:

```
PIP
*B:=A:TEXT.TXT[V]
```

Das war der erste Schritt. Allerdings finde ich es lästig, jedesmal PIP aufzurufen und dann erst in der zweiten Zeile den Befehl zu geben. Aus CP/M-Sicht ist das auch gar nicht erforderlich. Sie können den oben beschriebenen Befehl auch so abkürzen

```
PIP B:=A:TEXT.TXT[V]
```

Dann beginnt PIP sofort mit der Arbeit und kehrt danach direkt zum Prompt (A>) zurück. Sicher haben Sie gemerkt, daß ich für's Laufwerk B: keinen Dateinamen eingegeben habe. Wenn der Name der Datei beim Kopieren nicht verändert werden soll, können Sie genauso vorgehen. Möchten Sie den Namen aber ändern, geben Sie ein:

```
PIP B:TEXT1.TXT=A:TEXT.TXT
```

Damit können Sie Dateien gleichzeitig kopieren und den Namen ändern.

Bevor Sie aber mit PIP auf eine andere Diskette kopieren, sollten Sie sich vergewissern, ob dort auch genug Platz für die neue Datei ist. PIP überträgt nämlich die Datei auf die andere Diskette, errichtet dort eine Zwischendatei, die Sie am gleichen Dateinamen, aber mit einer $$$-Kennung, erkennen können. Erst wenn feststeht, daß die Datenübertragung erfolgreich war, benennt PIP die Zwischendatei auf den richtigen Namen um.

Nehmen wir an, Sie kopieren die Datei TEXT.TXT auf die Diskette B:, aber dort steht noch, mit gleichem Namen, die alte Version dieser Datei. Was macht PIP? Wenn die Diskette nicht mehr genug Platz bietet, bekommen Sie eine Fehlermeldung. Das liegt an der Arbeitsweise von PIP. Es versucht zuerst, die zu kopierende Datei auf die Diskette zu schreiben, obwohl die alte Version dort auch noch steht. Daher der Platzbedarf. Ist genügend Platz vorhanden, wird die neue Datei übertragen, mit $$$ gekennzeichnet, dann erst die alte Datei gelöscht und die neue umbenannt. Klappt also Ihre Kopieroperation einmal nicht, haben Sie möglicherweise nicht genug Platz für die neue Datei und müssen erst mit ERASE die alte Datei löschen.

Bei jeder Übertragung durch PIP ohne spezielle Option, werden die Datei-Attribute wie SYS,DIR,RO und RW mit übertragen. Wenn Sie also eine SYSTEM-Datei mit PIP kopieren, ist auch die Kopie eine SYSTEM-Datei.

Verlangen Sie von PIP, eine Datei zu übertragen und auf der Zieldiskette existiert bereits eine Datei gleichen Namens, ist aber schreibgeschützt (RO=Read Only), fragt PIP bei Ihnen nach, ob es diese Datei löschen darf.

Antworten Sie auf die Frage mit "Y" für Ja und mit "N" für nein.

VI.2 Zwischen Benutzerbereichen kopieren

Was Sie bis jetzt gelesen haben, ermöglichte nur die Übertragung von Dateien innerhalb eines USER-Bereiches. Befinden Sie sich im USER-Bereich 3 und verwenden PIP zum Kopieren einer oder mehrere Dateien, landen die Kopien auf der anderen Diskette auch im USER-Bereich 3. Denken Sie

daran, wenn Sie Ihre übertragenen Dateien einmal vermissen sollten.

Um nun von einer Diskette und gleichzeitig von einem USER-Bereich in einen anderen zu kopieren, benutzen Sie die Option "Gn", wobei das "n" für die Nummer des USER-Bereiches steht. Um vom Benutzerbereich 0 die Datei TEXT.TXT auf Laufwerk B: in den Benutzerbereich 2 zu kopieren, geben Sie ein:

```
PIP B:[G2]=A:TEXT.TXT
```

Denken Sie daran, daß andere CP/M-Befehle wie DIR,ERASE,TYPE usw. nur in dem jeweils angewählten USER-Bereich etwas ausführen, hier existieren keine Userbereichübergreifende Optionen.

VI.3 Text- und Nicht-Textdateien

Bis jetzt haben Sie PIP immer nur benutzt, um Dateien von einer Diskette auf die andere zu bringen. Wie ich oben schon erwähnte, haben Sie auch die Möglichkeit, aus mehreren Dateien eine Gesamtdatei zu erstellen, und zwar in einem Arbeitsgang.

Bevor Sie über dieses interessante Feature mehr erfahren, muß ich Ihnen noch schnell etwas zu den unterschiedlichen Dateiformen sagen. Grundsätzlich unterscheidet CP/M zwei Dateiformen: Die Textdatei und die Nicht-Textdatei. Eine Textdatei erstellen Sie mit Ihrem Textverarbeitungs-Programm oder dem in CP/M eingebauten Editor ED (lieber nicht) und verwenden dabei alle Zeichen, die Ihre Tastatur so hergibt.

Eine Nicht-Textdatei besteht aus Binärcode und trägt in der Regel die Kennung COM.

Diese beiden Dateiarten müssen unterschieden werden, weil CP/M das Ende von Textdateien an einem Control Z (^Z) erkennt. Jeder Text, der nach dem ^Z geschrieben steht, wird nicht ausgedruckt oder sonstwie beachtet. In Nicht-Textdateien dagegen, wie die COM-Dateien beispielsweise, ist ein ^Z ein ganz normales Zeichen, wie viele andere auch und CP/M schert sich nicht darum.

Normalerweise geht PIP immer davon aus, daß Nicht-Textdteien zu übertragen sind und liegt damit häufig richtig. Wenn Sie aber mit PIP mehrere Dateien zu einer zusammenfassen, geht PIP davon aus, daß es sich um Textdateien handelt. (Weil es doch zumeist recht unsinnig ist, beispielsweise COM-Dateien zusammenzufügen).

VI.4 Dateien zusammenkopieren

Als Erkennungszeichen für PIP, daß es mehrere Dateien zusammenlegen soll, dient das Komma. Aus diesem kühlen Grunde sollen Sie auch kein Komma in einem Dateinamen benutzen. Sie bringen sonst PIP völlig durcheinander.

Eine einfache Zusammenlegung verschiedener Dateien veranlassen Sie so:

```
PIP ALLES.TXT=TEIL1.TXT,TEIL2.TXT,TEIL3.TXT
```

wenn alle Dateien auf dem gleichen Laufwerk stehen. Soll die Sammeldatei in einen anderen USER-Bereich geschrieben wer-

den, geben Sie bei CP/M 3.0 ein:

PIP ALLES.TXT[G3]=TEIL1.TXT,TEIL2.TXT,TEIL3.TXT

Bestehen Sie zusätzlich noch auf der Überprüfung nach dem Kopieren, müssen Sie hinter jede Datei, die Teil der neuen werden soll ein [V] eingeben. Damit sieht die Befehlszeile dann schon etwas komplizierter aus:

PIPALLES.TXT[G3]=TEIL1.TXT[V],TEIL2.TXT[V],TEIL3.TXT[V]

Auf jeden Fall ist das immer noch weniger Arbeit, als alle drei Datei "von Hand" zusammenzutragen.

Wollen Sie Nicht-Textdateien mit PIP zusammenfassen, kann es zu Problemen kommen. Wie Sie gelesen haben, denkt PIP jedesmal wenn ein ^Z auftaucht, die Datei sei zu Ende. In Nicht-Textdateien kann das ^Z häufiger vorkommen und würde so eine Übertragung mit PIP unmöglich machen oder doch sehr erschweren.

Weil die CP/M-Erfinder dieses Problem kennen und davon ausgehen, daß meistens COM-Dateien als Nicht-Textdateien übertragen werden, übersieht PIP gnädig jedes ^Z in einer COM-Datei und kopiert bis zum bitteren Ende derselben.

Wenn Sie Dateien kopieren möchten, die weder Textdateien noch COM-Dateien sind, müssen Sie den Parameter "O" mit angeben. Wie gehabt, mit den eckigen Klammern direkt hinter dem Dateinamen.

Sie haben sogar die Möglichkeit, während der Zusammenstellung mehrerer Dateien in jede Datei noch etwas hineinzuschreiben. Der Kopierbefehl muß dann so aussehen:

PIP ALLES.TXT=TEIL1.TXT,CON:,TEIL2.TXT,CON:,TEIL3.TXT

So beginnt PIP die erste Datei zu übertragen, stoppt dann und erwartet Ihre Eingaben über die Tastatur, angeregt durch den Befehl CON:.

Allerdings müssen Sie die Befehl für RETURN und "neue Zeile" (^J) von Hand eingeben, damit PIP keine Zeile überschreibt. Mit der Eingabe von ^Z geht's dann weiter. Bedenken Sie aber, daß Schreibfehler nicht korrigiert werden können.

VI.5 Zeilen durchnummerieren

Eine nette Einrichtung für Programmierer, Journalisten und alle, die ihre Texte oder Programme zeilenweise numeriert haben möchten, ist der "N"-Parameter von PIP. Wenn Sie eine Datei mit diesem Parameter oder seinem Bruder "N2" kopieren, bekommt jede Zeile Ihres Textes eine eigene Nummer.

Kopieren Sie so:

PIP TEXTNR.TXT=TEXT.TXT[N]

erscheinen Ihre Textzeilen hinterher so:

1: Dies ist Ihr Text
2: Zeilenweise numeriert
3: praktisch, nicht wahr?

Nehmen Sie dagegen den Parameter "N2", erscheint Ihr Text hinterher so:

000001 Die ist Ihr Text
000002 Zeilenweise numeriert durch N2 als Option
000003 Auch nicht ohne, oder?

Die Zeilen werden einfach hochgezählt, was sich nicht ändern läßt. Also in der Reihenfolge: 1,2,3,4,5...

VI.6 Buchstaben umwandeln

Möchten Sie statt Zeilenummern lieber Ihren gesamten Text in Großbuchstaben erscheinen lassen, sind Sie ein Kandidat für den Parameter "U". Das "U" steht für "Uppercase", was Großbuchstaben bedeutet. Das Gegenteil bekommen Sie, wenn Sie den Parameter "L" für "Lowercase" eingeben. Dann besteht Ihr gesamter Text nur noch aus kleinen Buchstaben. Anwendungsgebiete? Beispielsweise beim Listendruck u.ä.

VI.7 String suchen

Manchmal passiert es, daß der Drucker streikt oder einfach mitten im Druck das Papier zu Ende geht. Dann haben Sie unter Umständen das Problem, daß ein Teil Ihres Ausdruckes zwar noch auf der Diskette steht, aber nicht auf dem Papier. Bevor Sie nun die ganzen 200 Seiten nochmal ausdrucken, benutzen Sie lieber einen weiteren PIP-Parameter, um den restlichen Text auf's Papier zu bekommen.

Sie können mit PIP einen Textteil kopieren, indem Sie PIP mitteilen, ab welchem String, das ist eine Zeichenkette, es suchen und bei welchem String es wieder aufhören soll. Den Anfangs-String markieren Sie durch ein "S" für "Start" und den End-String mit einem "Q" für "Quit", was aufhören bedeutet. Schreiben Sie hinter jede Zeichenkette ein ^Z und PIP macht den Rest. Es durchsucht Ihren Text bis es auf die vorgegebene Zeichenkette stößt, kopiert und stoppt bei der zweiten, von Ihnen vorgegebenen Zeichenkette. Ein solcher Befehl sieht so aus:

```
PIP
*TEXTTEIL=TEXT.TXT[Qsuchwort^Z]
```

Wenn Ihre Eingabezeile so aussieht, beginnt PIP am Anfang der Datei mit dem Kopieren und stoppt, sobald das Wort "suchwort" auftaucht. Beachten Sie bitte, daß PIP ohne Option aufgerufen und erst in der zweiten Zeile alles andere eingegeben wurde. So sucht PIP nach dem "suchwort", wie Sie es eingetippt haben, nämlich in Kleinbuchstaben.

Schreiben Sie den obigen Befehl in einer Zeile, wandelt PIP das "suchwort" erst in Großbuchstaben um und beginnt dann

mit der Suche. Steht Ihr "suchwort" nicht als "SUCHWORT" in Ihrer Datei, findet PIP nichts und gibt Ihnen eine Fehlermeldung aus. Diese Fehlermeldung lautet, für alle, die es nicht selbst ausprobieren wollen:

```
      ERROR: START NOT FOUND
oder: ERROR: QUIT NOT FOUND
```

Einen Textabschnitt kopieren Sie mit diesem Befehl:

```
PIP
*TEXTTEIL=TEXT.TXT[Sanfang^ZQende^Z]
```

Das Programm beginnt mit dem Kopieren beim Wörtchen "anfang" und hört beim ersten Auftauchen des Wortes "ende" auf. Auf diese Weise können Sie beispielsweise Teile aus Programmlistings kopieren; aus einem PASCAL-Quellprogramm eine Prozedur zum Beispiel.

VI.8 Mehrere Dateien hintereinader drucken

Sie kennen bisher die Methode eine Datei auszudrucken, indem Sie ^P eingeben und dann mit dem Befehl TYPE die Bildschirmausgabe auf den Drucker umleiten. Das ist etwas umständlich und kann mit Hilfe von PIP besser gelöst werden. Jedes Peripherie-Gerät hat für PIP einen eigenen Namen. Das Ausgabegerät wird mit LST: bezeichnet. Sie erinnern sich, daß wir das Eingabegerät (die Tastatur) bereits einmal durch CON: angesprochen haben. Die Doppelpunkte sind wichtig, damit PIP diesen Gerätenamen von einem Dateinamen unter-

scheiden kann. Wollen Sie Ihre Textdatei auf dem Drucker erscheinen lassen, geben Sie ein:

```
PIP LST:=TEXT.TXT
```

Sie können selbstverständlich alle Optionen, die PIP beeinflussen, auch bei dieser Operation verwenden. Um zum Beispiel mehrere Dateien hintereinander zu drucken, geben Sie ein:

```
PIP LST:=TEXT1.TXT,TEXT2.TXT,TEXT3.TXT
```

und der Text kommt hintereinander aus Ihrem Drucker gequollen. Wenn Sie die ausgedruckte Form Ihres Textes beeinflussen wollen, benutzen Sie den Namen PRN:. Sie bekommen dann durchnumerierte Textzeilen, acht Zeichen breite Tabulator-Schritte und alle 60 Zeilen wird ein neues Blatt begonnen.

Am Ende dieses Kapitels gebe ich Ihnen einige Beispiele, was man mit PIP Sinnvolles machen kann.

VI.9 Dateien automatisch sichern

Nun ist es aber nicht nur sehr sinnvoll, Daten auszudrucken, sondern genauso wichtig, Daten zu sichern. Ich habe Sie bereits darauf hingewiesen und spreche aus leidvoller Erfahrung. Bei meinen Wanderungen durch das PIP-Handbuch habe ich dann eine PIP-Funktion entdeckt, die das Sichern wesentlich erleichtert. Es ist die "Archiv"-Option, abgekürzt "A".

Wie so oft bekommt man auch hier nur durch häufiges "Probieren" heraus, wie man eine solche Option sinnvoll einsetzen kann.

Besonders wenn Sie mit einer Harddisk arbeiten, werden Sie die Vorteile dieses Parameters schnell schätzen lernen. Jede Datei in CP/M 3.0 besitzt im Dateikopf einen Platz für ein Bit, das mit "archive flag" bezeichnet wird. Ein Flag ist soetwas wie ein Anzeiger, der einen bestimmten Zustand anzeigt. Jedesmal, wenn Sie eine Datei eröffnen oder ändern, wird dieses Flag verändert. Also beim Ändern einer Datei zum Beispiel auf "1" gesetzt und nach erfolgter Datensicherung wieder auf "0".

So kann PIP erkennen, welche Dateien seit der letzten Sicherung verändert oder neu erstellt wurden und nur diese auf eine Diskette sichern. Sie ersparen sich damit, jedesmal die gesamte Harddisk zu sichern, was eine Menge Zeit und Geld für zusätzliche Disketten kosten würde.

Nach einem langen Arbeitstag sichern Sie Ihre bearbeiteten Dateien, indem Sie eingeben:

```
PIP B:=A:*.TXT[A]
```

Wollen Sie sicher gehen, setzen Sie noch die "V"-Option dazu und jede Datei wird auf vollständige Übertragung überprüft

```
PIP B:=A:*.TXT[AV]
```

Wenn Sie die Originale der gesicherten Dateien auf der Festplatte jetzt nicht mehr anrühren, werden diese beim nächsten Sichern nicht beachtet.

Sie können sich im Inhaltsverzeichnis ansehen, welche Dateien gesichert sind und welche nicht. Dazu benutzen Sie den Befehl DIR mit den Optionen FULL oder RW. Im Inhaltsverzeichnis sehen Sie die gesicherten Datein mit "Arcv" gekennzeichnet.

VI.10 Überschreiben ohne Rückfrage

Es gibt noch ein paar Feinheiten zu beachten, wenn Sie mit PIP arbeiten. Normalerweise überschreibt PIP beim Kopieren ohne zu zögern eine Datei auf der Zieldiskette, wenn sie den gleichen Namen besitzt. Handelt es sich dabei allerdings um eine Datei, die durch das Attribut R/O (Read Only) geschützt ist, fragt PIP extra nochmal nach, ob diese Datei überschrieben werden darf.

DESTINATION FILE IST R/O, DELETE (Y/N)?

Geben Sie hier "Y" für Ja ein, wird die Datei auf der Zieldiskette überschrieben und damit gelöscht. Bestehen Sie auf Ihrem Nein, indem Sie "N" drücken, bricht PIP die Kopieroperation ab. Nach "Y" oder "N" brauchen Sie übrigens kein RETURN zu drücken.

Möchten Sie, daß auf jeden Fall alle Dateien auf die Zieldiskette geschrieben werden und die Frage nach der Löscherlaubnis unterbleibt, verwenden Sie den "W"-Parameter.

Der Befehl sieht dann so aus:

```
PIP B:=A:TEXT1.TXT,TEXT2.TXT,TEXT3.TXT[W]
```

Damit übernehmen Sie aber auch die volle Verantwortung für's Kopieren und können PIP keinen Vorwurf machen. (Was eigentlich meistens ziemlich sinnlos ist).

VI.11 Systemdateien kopieren

Versuchen Sie eine Datei zu kopieren, die das SYSTEM-Attribut besitzt, bringen Sie PIP in Verlegenheit. Es kann sie nämlich nicht finden. Sie müssen ihm schon ein wenig helfen und die "R"-Option mit auf den Weg geben. Natürlich können Sie die beiden Optionen auch kombinieren. Sollen Dateien und SYSTEM-Dateien auf die Zieldiskette geschrieben werden, ohne Rücksicht darauf, ob sie schreibgeschützt sind oder nicht, geben Sie ein:

```
PIP B:=A:*.COM[RW]
```

Damit kopieren Sie alle COM-Dateien und überschreiben gleichnamige auf der Zieldiskette. Sie erinnern sich: Systemdateien sind Dateien, die nicht im "normalen" Inhaltsverzeichnis auftauchen. Sie werden nur beim Kommando DIRSYS angezeigt. Systemdateien sind in jedem Benutzerbereich zugänglich.

VI.12 Das 8. Bit "säubern"

Der amerikanische Zeichensatz (ASCII) wird mit Hilfe von sieben Bits dargestellt. Das achte Bit, das ist die im Computer verwendete Wortlänge, bleibt für Sonderaufgaben frei. WordStar zum Beispiel verwendet es, um die Zwischenräume beim Schreiben in Blocksatz zu markieren. Andererseits verlangt MBASIC, daß das achte Bit frei sein muß. Wenn Sie mit WordStar ein BASIC-Programm bearbeiten und fälschlicherweise den Dokumenten-Modus (D) gewählt haben, kann Ihr BASIC-Programm nicht laufen, weil das 8. Bit nicht frei ist. Dieses Problem bekommen Sie schnell in den Griff, wenn Sie die "Z"-Option von PIP benutzen. Sie kopieren einfach Ihre Datei mit PIP und alles ist wieder in Butter. Das entsprechende Kommando sieht dann so aus:

```
PIP SPIEL.BAS=SPIEL.BAS[Z]
```

Wenn Sie wollen, können Sie auch noch die "V"-Option dazu setzen. Aber - benutzen Sie den Befehl in dieser Form nicht, um WordStar-Dateien zu kopieren. Sie verlieren sonst die Formatierung des Textes, da ja die Wortzwischenräume bzw. dessen Markierungen zerstört werden.

VI.13 Praktische Beispiele

Sie sollen jetzt noch ein paar Beispiele sehen, wie Sie PIP sinnvoll einsetzten können. Die verschiedenen Parameter haben Sie in diesem Kapitel kennengelernt und wissen auch, daß man sie zusammen anwenden kann.

Einen Textausdruck in besserer Form als nur mit ^P oder einfachem LST: ergibt diese Befehlsform:

```
PIP LST:TEXT.TXT[NT8P60]
```

Damit wird die Datei TEXT.TXT zum Drucker geschickt, alle Zeilen werden durchnummeriert, Tabulatoren sitzen in jeder achten Spalte und die Seitenlänge beträgt 60 Zeilen. Wenn Sie einmal zurückblättern werden Sie feststellen, daß dies genau die voreingestellten Werte von PRN: sind. Jetzt wissen Sie, was Sie an PRN: haben: weniger Arbeit.

Möchten Sie beim obigen Beispiel noch zusätzlich dafür sorgen, daß alle Buchstaben als Kleinbuchstaben erscheinen, geben Sie ein:

```
PIP LST:TEXT.TXT[NT8P60L]
```

Sie fügen also nur die "L"-Option dazu und schon klappt's.

Um die Archiv-Arbeit zu rationalisieren, schreiben Sie sich in CP/M 3.0 eine SUBMIT-Datei folgenden Inhalts:

```
PIP A:=B:*.*[WAR]
```

und nennen diese ARCHIV.SUB. Die Optionen "WAR" bewirken, daß SYSTEM-Dateien mit kopiert, schreibgeschützte Dateien ohne Nachfrage überschrieben und nur solche Dateien kopiert werden, die noch nicht archiviert sind. Wenden Sie diesen

Befehl regelmäßig an, kann Ihren Daten eigentlich nichts passieren. Der Aufruf erfogt einfach mit:

SUBMIT ARCHIV

und der Rest geht automatisch. Sie können den Befehl noch erweitern, wenn Sie möchten. Mit der Option "V" werden die Dateien auf richtige Übertragung überprüft und durch den Parameter "E" bekommen Sie alles Kopierte auf dem Bildschirm angezeigt. Das "E" sollten Sie aber nur bei Textdateien verwenden, weil Sie sonst Schwierigkeiten bekommen.

Weitere Möglichkeiten wären, den DIR und den SHOW-Befehl in die SUBMIT-Datei mitaufzunehmen. Dann sehen Sie welche und wieviele Dateien kopiert werden sollen und wieviel Platz noch auf der Zieldiskette vorhanden ist.

Beim Commodore 128 hat man - alleine schon wegen des 40- und des 80-Zeichen-Bildschirmes - mehr Gerätekanalbezeichnungen zur Verfügung. Dies sind die folgenden:

KEYS = Tastatur des Commodore 128
40COL = 40-Zeichen-Monitor
80COL = 80-Zeichen-Monitor
PRT1 = Serieller Drucker am IEC-Bus (Gerätenummer 4)
PRT2 = Serieller Drucker am IEC-Bus (Gerätenummer 5)

Folgende Einstellungen für Ein-/Ausgabe sind Standard:

CONIN: = KEYS
CONOUT: = 80COL (40COL)

AUXIN: = Null Device
AUXOUT: = Null Device
LST: = PRT1

Sie sehen, daß die Druckerausgabe standardmäßig auf den seriellen Drucker am IEC-Bus, Geräteadresse 4 erfolgt.

VI.14 Was Sie jetzt schon wissen

* *PIP ist eines der leistungsfähigsten Unter-Programme von CP/M*
* *Sie können einzelne Dateien auf eine andere Diskette übertragen*
* *Sie können ganze Disketten auf einmal kopieren*
* *Mit PIP machen Sie aus mehreren Dateien eine*
* *Sie können ein Stück aus einer Datei herausholen*
* *PIP nummeriert für Sie automatisch die Zeilen einer Text-Datei*
* *Mit PIP können Sie zuletzt veränderte Dateien automatisch sichern*
* *Sie können Großbuchstaben in Kleinbuchstaben verwandeln und umgekehrt*
* *Sie können zwischen den verschiedenen Benutzer-Bereichen kopieren*

VII. CP/M intern C-128

VII.1 Gernerelles zum CP/M 3.0 auf dem C-128

CP/M 3.0 ist nicht gleich CP/M 3.0 - das stellt man immer wieder fest. Jedoch muß man sagen, daß sich die Hersteller neuer Rechner *weitestgehend* an die CP/M-Norm halten, die ihnen von Digital Research vorgegeben wurde. Die verschiedenen hardwaremäßigen Gegebenheiten machen es notwendig, das eine oder andere von CP/M zu ändern oder einiges hinzuzufügen. Wir wollen zunächst auf einige Besonderheiten des CP/M 3.0 auf Ihrem Commodore 128 eingehen, damit Sie das Betriebssystem auch optimal nutzen können.

Erste wichtige Eigenschaft ist, daß sich das System CP/M 3.0 auf einer Diskette befindet, die von der 1541, der 1570 und der 1571 gleichermaßen gelesen werden kann. Bei der 1571 hat man allerdings den Nachteil, daß man den eigentlichen Vorteilt der 1571 - das beidseitige Lesen und Schreiben einer Diskette - überhaupt nicht ausnutzen kann. Andauernd muß man die Diskette wechseln, weil die COM-Dateien auf beiden Diskettenseiten verstreut sind.

VII.2 Systemdiskette für 1571-Besitzer

Als glücklicher Besitzer der 1571 sollten Sie die Eigenschaften der 1571 auch unbedingt nutzen. Sie ersparen sich nicht nur Arbeit, sondern auch Zeit und Nerven, wenn Sie eine 1571-gerechte Systemdiskette erstellen. Hierzu müssen Sie eine neue Diskette mit FORMAT **beidseitig** formatieren. Dies ist der wichtigste Schritt für eine 1571-Systemdiskette. Es genügt nicht, beide Seiten einzeln zu formatieren.

Haben Sie dies getan, so müssen beide Diskettenseiten der Originaldiskette auf die neue Systemdiskette kopiert werden. Legen Sie hierzu die Seite A mit der Beschriftung "CP/M SYSTEM DISK" ins Laufwerk ein und geben Sie folgendes PIP-Kommando ein:

PIP E:=A:*.* (bzw. PIP B:=A:*.*)

CP/M 3.0 fordert Sie auf, die Disketten regelmäßig zu wechseln. Folgen Sie diesen Aufforderungen und bedenken Sie: Diskette E ist die **Zieldiskette**, Diskette A ist die **Quelldiskette**. Ist diese Diskettenseite kopiert, legen Sie wieder die Originaldiskette mit der Seite A ein. Nun muß noch die zweite Seite der Diskette kopiert werden. Hierzu geben Sie folgendes Kommando ein:

PIP A:=E:*.* (bzw. PIP B:=E:*.*)

Es ist also unbedingt erforderlich - auch wenn Sie zwei Laufwerke haben -, daß die Diskette vor dem eigentlichen Kopiervorgang im Diskettenlaufwerk A umgedreht wird. Jetzt ist die Diskette A die **Zieldiskette**, mit Namen E wird die **Quelldiskette** bezeichnet. Befolgen Sie auch hier die Aufforderungen zum Wechseln der Disketten.

Ist diese Prozedur abgeschlossen, so verfügen Sie über eine beidseitig beschriebene Systemdiskette, die Sie ab diesem Augenblick verwenden sollten. Es ist einfacher, mit einer solchen Diskette zu arbeiten, auf der alle CP/M-Kommandos jederzeit zur Verfügung stehen.

Am besten, Sie ziehen sich sofort noch eine Sicherheitskopie von dieser 1571-Systemdiskette und verwahren diese sicher im Schrank.

VII.3 Das virtuelle Laufwerk E:

Schon oft haben wir darüber gelesen - in diesem Buch und noch öfter als Laufwerksangabe E: angewandt. Unter CP/M 3.0 auf dem Commodore 128 ist es theoretisch möglich, bis zu vier Laufwerke anzuschließen. Auch unter BASIC ist dies vorgesehen. Die Floppy-Laufwerke hätten dann die Geräteadressen 8 bis 11, die Sie ja im bzw. am Laufwerk (1571) einstellen können. Unter CP/M spricht man diese Laufwerke ja nicht unter der Geräteadresse an, sondern unter der Laufwerksangabe, die aus einem Buchstaben und einen Doppelpunkt besteht. Bei vier möglichen Laufwerken bedeutet dies jedoch nichts anderes als daß wir Laufwerke mit den Bezeichnungen A:, B:, C: und D: haben können.

Nun gut, was ist aber nun bitte das *virtuelle fünfte Laufwerk* mit Namen E:? Der Duden sagt zu dem Wort "virtuell":

"Der Kraft oder Möglichkeit nach vorhanden, scheinbar"

Dem muß man eigentlich kaum mehr etwas anfügen! Das fünfte Laufwerk kann man zwar ansprechen, ist jedoch in Wirklichkeit gar nicht vorhanden. Es wird durch das Laufwerk A ersetzt. Jedoch unterscheidet CP/M immer, ob es nun auf Laufwerk A oder auf Laufwerk E zugreift. Dabei weiß CP/M genau, ob das Laufwerk A nun wirklich Laufwerk A ist, oder ob es gerade Laufwerk E spielt. Wird das anderes Laufwerk

verlangt als dasjenige, das gerade durch das Laufwerk A repräsentiert wird, so fordert CP/M zum Wechsel der Disketten und zum Betätigen der RETURN-Taste auf.

Dank dieser - möchte sagen genialen - Lösung, ist es unter CP/M auch mit einem Laufwerk möglich, Dateien mit PIP zu kopieren. Sie können Kommandos ausführen, die sich auf einer anderen Diskette befinden oder mit Dateien agieren, die sich auf einer anderen als der eingelegten Diskette befinden. All diese Dinge währen mit nur einem Laufwerk sonst gar nicht oder nur mit Spezialbefehlen möglich gewesen. Sie sollten sich deshalb schnell an das "virtuelle Laufwerk" gewöhnen und es akzeptieren - auch wenn es nur "virtuell" ist.

Für alle diejenigen Leser, die über zwei Laufwerke verfügen, muß man sagen, daß sie natürlich einen immensen Vorteil gegenüber den anderen CP/M-Benutzern haben. Das Kopieren beispielsweise geht viel schneller; man hat einen Zugriff auf viel mehr Daten zur gleichen Zeit etc. Machen Sie sich dieses zweite (oder gar dritte und vierte) Laufwerk beim Arbeiten auch zunutze. Dennoch kann es manchmal sinnvoll sein, vom virtuellen Laufwerk Gebrauch zu machen.

VII.4 COPYSYS auf dem Commodore 128

Bei jedem "normalen" CP/M-3.0-Rechner wird das System, das aus den beiden oberen Spuren 0 und 1 besteht, mit der Eingabe des Kommandos **COPYSYS** kopiert. Dies ist beim Commodore 128 nicht der Fall: Das Kopieren des Systemes ist viel einfacher, da man es in den beiden Dateien CPM+.SYS und CCP.COM untergebracht hat, die man ganz einfach mittels PIP kopieren kann.

Sollten Sie aber das Kommando COPYSYS gewohnt sein, so ist dies nicht ganz so schlimm. Der Commodore 128 zeigt Ihnen dann ganz "nett" auf dem Bildschirm an, daß Sie Ihr gestecktes Ziel mit dem Kommando COPYSYS nicht erreichen; es existiert nämlich eine Datei COPYSYS.COM, so daß Sie keine Fehlermeldung wie etwa

COPYSYS?

auf dem Bildschirm erhalten. Wie bereits in einem anderen Kapitel erwähnt, wird das System unter CP/M 3.0 auf dem Commodore 128 mit folgenden drei Kommandos kopiert:

- **FORMAT**
- **PIP E:=A:CPM+.SYS**
- **PIP E:=A:CCP.COM**

Es werden dann alle residenten Kommandos, aber auch die Bootroutinen etc. kopiert. Ebenfalls wird der Bootsektor an Spur 1, Sektor 0 eingerichtet. Nach Eingabe dieser drei Kommandos ist die Systemdiskette bereits fertig - d.h. sie ist bootbar, da sich aber kein einziges transientes Kommando auf der Diskette befindet, ist sie natürlich noch relativ nutzlos. Sie können sich aber alle von Ihnen zumeist benötigten COM-Dateien kopieren - auf diese Weise können Sie einigen Platz auf der Diskette sparen, wenn Sie nicht immer alle COM-Dateien kopieren.

VII.5 Die Statuszeile

Die letzte Zeile auf Ihrem Bildschirm ist die sogenannte *Statuszeile.* Sie werden in dieser Zeile immer die wichtigsten Informationen erhalten, die ausschließlich vom System kommen. So werden Sie in genau dieser Zeile beispielsweise aufgefordert, die Diskette A oder die Diskette E einzulegen und die Taste RETURN zu betätigen.

Keinem, der bereits mit dem CP/M 3.0 auf dem Commodore 128 gearbeitet hat, wird es verborgen geblieben sein, daß in der rechten unteren Ecke alle möglichen Zahlen und Buchstaben erscheinen, die sich ab und zu abwechseln.

Hinter diesem Chaos steckt aber System: Es wird an dieser Stelle der sogenannte *Disk Status* angezeigt. Im groben kann man sagen, daß angezeigt wird, welcher Block gerade gelesen oder beschrieben wird. Das Format dieser Disketten-Statusanzeige sieht so aus:

O Ltt ss

Dabei haben die Kennzeichnungen im einzelnen folgende Bedeutungen:

- O = Operation, R=Read (Lesen) und W=Schreiben (W)
- L = Physikalisches Laufwerk (A,B,C oder D)
- tt = Zweistellige Tracknummer (Spur) des gerade zu lesenden oder zu schreibenden Blockes
- ss = Zweistellige Sektornummer des gerade zu lesenden oder zu schreibenden Blockes

Spur- und Sektorangabe wird normalerweise durch ein Leerzeichen getrennt sein. Ist die eingelegte Diskette allerdings im sogenannten MFM-Format also beidseitig formatiert, so werden Spur- und Sektorangabe bei Zugriffen auf die zweite Seite durch einen Bindestrich (-) getrennt.

Sollte Sie diese Diskettenstatuszeile stören oder Sie empfinden sie als unnötig, so können Sie diese Anzeige durch die Tastenkombination <CTRL>-<RUN/STOP> abschalten. Die Control-Taste müssen Sie dabei zuerst niederhalten. Mit dieser Tastenkombination können Sie auch jederzeit die Statusanzeige wieder einschalten.

Sie werden es übrigens nie erleben, daß etwas in die Statuszeile hineingeschrieben wird, beispielsweise bei dem Kommando TYPE oder ähnlichen. Die letzte Zeile ist davor geschützt.

Wollen Sie die letzte Befehlszeile wiederholen, so reicht es aus, wenn Sie die Cursor-Down-Taste rechts unten auf der Commodore-Tastatur betätigen. Der seperate Cursor-Block bleibt hierbei ausgeschlossen, dieser hat eine andere wichtige Funktion. Haben Sie beispielweise als letztes Kommando DIR *.* eingegeben, so erscheint auf dem Bildschirm dieser Text auch nach Betätigen der Cursor-Down-Taste. Allerdings wird keine automatisches RETURN ausgeführt - Sie können also die letzte Eingabe editieren um Fehler o.ä. zu korrigieren.

VII.6 Die Diskettenformate

Die 1570 und die 1571 sind ja in der Lage, verschiedene Formate bezüglich CP/M zu lesen. *Alles nun folgende gilt mit keinem Wort für die 1541!*

Der Controller - der sich bei der 1570 und der 1571 programmieren läßt - macht diese verschiedenen Formate möglich. Folgende Formate werden von den beiden Laufwerken unterstützt (die zweiseitigen Formate natürlich nur von der 1571):

```
Osborne DD          (1024 Bytes/Sektor, einseitig,   5 Sektoren/Spur)
Epson QX10          (512  Bytes/Sektor, zweiseitig, 10 Sektoren/Spur)
IBM-8 SS (CP/M 86)  (512  Bytes/Sektor, einseitig,   8 Sektoren/Spur)
IBM-8 DS (CP/M 86)  (512  Bytes/Sektor, zweiseitig,  8 Sektoren/Spur)
KayPro II           (512  Bytes/Sektor, einseitig,  10 Sektoren/Spur)
KayPro IV           (512  Bytes/Sektor, zweiseitig, 10 Sektoren/Spur)
```

Es werden also sechs Formate direkt unterstützt. Theoretisch können auch alle anderen Formate durch Programmierung des Controllers realisiert werden. Wie man aber diesen Controller programmiert, kann im CP/M-Buch nicht erläutert werden. Andere Lektüre wie beispielsweise das Floppybuch wird hier mehr Auskünfte geben können.

Wenn Sie eine Diskette einlegen - das haben Sie bestimmt bereits einmal bemerkt - und das Türchen schließen, beginnt der Motor kurz zu laufen. Die intelligente Floppy versucht das Format zu erkennen. Dabei dienen die Angaben über Bytes pro Sektor und Sektoren pro Spur zur Lokalisierung des Formates. Sollte durch diese Angaben allein keine Bestimmung des Formates möglich sein, so erscheint in der linken unteren Ecke ein Kästchen mit den möglichen Formaten. Sie können durch Scrollen im Fenster nun dem System mitteilen, um welches Format es sich bei der neu eingelegten Diskette handelt. Dieses Scrollen wird durch die Tasten Cursor-Links und Cursor-Rechts ermöglicht. Haben Sie das eingelegte Format im Fenster erreicht, so können Sie das Format durch Betätigen der RETURN-Taste bestätigen.

Wollen Sie zukünftig nur noch dieses Format verwenden, so betätigen Sie die Control- und die RETURN-Taste zur gleichen Zeit. So merkt sich CP/M das erkannte Format und fragt nicht jedesmal nach, wenn Sie die Diskette wieder einlegen.

VII.7 Die Tastatur

Die Tastatur des Commodore 128 unter CP/M unterscheidet sich ein wenig von der unter BASIC bekannten Tastatur. Die Cursortasten beispielsweise tun nicht alle das, was sie sollen. Betätigt man die Cursor-Runter-Taste rechts unten auf der Tastatur, so wird das zuletzt eingegebene Kommando wiederholt. Wir haben dies bereits erwähnt. Betätigt man aber die entsprechende Taste im oberen separaten Cursorblock, so passiert diesbezüglich gar nichts.

Eine Besonderheit direkt vorweg: Wollen Sie das System neu booten, so brauchen Sie nicht die RESET-Taste zu suchen, Sie können denselben Effekt auch über die Tastenkombination <CTRL>-<ENTER> erreichen, gemeint ist die ENTER-Taste rechts in der Zehnertastatur.

Jede Taste auf der großen 128er-Tastatur hat bis zu vier verschiedene Werte. Diese vier Werte ist der ungeshiftete Wert, der geshiftete Wert, der Control-Wert und last not least der CAPS-LOCK-Wert. Sie sehen, daß der CAPS-LOCK-WERT nicht immer mit dem geshifteten Wert übereinstimmen muß. Den ungeshifteten Wert erhalten Sie durch einfaches Betätigen der Taste, die CAPS-LOCK-Taste ist ausgerastet. Es erscheint dann das mit schwarzer Schrift markierte Zeichen auf dem Bildschirm. Die ASCII/DIN-Taste hat keine Wirkung im CP/M-Modus auf die Tastaturdekodierung.

Der geshiftete Wert ergibt sich selbstverständlich durch Betätigen einer Taste bei gleichzeitigem Niederhalten der

SHIFT-Taste. Der Control-Wert ergibt sich dementsprechend bei gleichzeitigem Betätigen der Control- und einer anderen beliebigen Taste (wie im Beispiel bereits erwähnt mit der ENTER-Taste). Der CAPS-LOCK-Modus wird durch Betätigen der Commodore-Taste erreicht und auch durch nochmaliges Betätigen wieder verlassen. Anders als bei Betätigen der CAPS-LOCK-TastewerdenimCommodore-ModusnurdieBuchstaben geshiftet, nicht aber die Ziffern und sonstigen Tasten. Das ist ein simulierter CAPS-LOCK-Modus, so wie er sein sollte.

Die Zifferntastenreihe oberhalb der Buchstabentasten dienen bei gleichzeitigem Betätigen der Control-Taste der Auswahl der Zeichenfarben; die Cursorfarbe ändert sich dann entsprechend (zunächst). Sie kennen diese Farben aus dem BASIC-Modus. Sie können lediglich acht - also nicht alle immerhin theoretisch denkbaren 16 Farben auswählen. Wenn Sie die Zifferntasten der Zehnertastatur in Verbindung mit der Control-Taste betätigen, so ändert sich die Hintergrundfarbe des 80-Zeichen-Bildschirmes entsprechend.

Wer übrigens nicht über das Zeichen "#" verfügt, weil er beim Booten von CP/M auf deutschen Zeichensatz geschaltet hat, der erhält dieses Zeichen, indem er die Pfund-Taste betätigt. Die Pfeil-Hoch-Taste generiert ^ im ungeshifteten Modus und den Querbalken I im Control-Modus, den Sie bei manchen Kommandos benötigen. Die Pfeil-Links-Taste oberhalb der Control-Taste erzeugt den Underlinestrich _.

VII.8 Sonderfunktionen

Wie sich noch zeigen wird, ist es kein Problem, auf dem Commodore 128 unter CP/M 3.0 die verschiedenen Tasten mit den verschiedensten Werten zu belegen. Hierfür gibt es dann sogar zwei Wege. Wie man beispielsweise den Inhalt der HELP-

Taste leicht abändert und andere Dinge mehr, erfahren Sie in dem nun folgenden Beschreibungen.

Alle nun im folgenden beschriebenen Sonderfunktionen - es sind drei an der Zahl - erreicht man, indem man die Control- und die rechte SHIFT-Taste gleichzeitig betätigt. Ferner kommt dann noch die dritte Taste hinzu, die man zur Funktionsselektierung benötigt. Es ist wichtig, daß Sie die *rechte* SHIFT-Taste betätigen, die linke erzielt nicht denselben Effekt. Folgende drei Tasten dienen nun dieser Funktionsauswahl:

* *ALT-Taste*
* *Cursor-Rechts-Taste im Cursor-Block*
* *Cursor-Links-Taste im Cursor-Block*

Grob gesagt kann man die ASCII-Werte der verschiedenen Tasten ändern. Hat eine Taste einen Wert größer als #80 (dezimal 128), so besitzt sie einen Sonderstatus. Ein ASCII-Wert größer als 127 ändert die Vorder- oder Hintergrundfarbe des 40- oder 80-Zeichenbildschirmes oder ruft einen festprogrammierten String hervor; etwa so, wie die HELP-Taste. Hier zunächst eine Tabelle der ASCII-Codes, die größer als 127 sind:

```
80-9F: Funktionsstring 0-15
A0-AF: 80-Zeichen-Vordergrundfarben
B0-BF: 80-Zeichen-Hintergrundfarben
C0-CF: 40-Zeichen-Vordergrundfarben
D0-DF: 40-Zeichen-Hintergrundfarben
E0-EF: 40-Zeichen-Rahmenfarbe
F0-FF: Spezialfunktionen
```

VII.8.1 Das Ein/Ausschalten des Sondermodus

Daß man die Vorder- und Hintergrundfarbe des 80-Zeichen-Bildschirmes ändern kann, haben wir bereits kennengelern - Sie erinnern sich, durch die Zifferntasten 0-9 in Verbindung mit der CONTROL-Taste konnte man dies erreichen. Das bedeutet nichts anderes, als daß diese Tasten mit den Werten #A0-#A7 bzw. #B0-#B7 belegt sein müssen. Und genauso ist das auch. Wie wir diese Tastenwerte ändern können, darauf kommen wir gleich noch zu sprechen. Erst einmal wollen wir Ihnen die erste Dreier-Tastenkombination nicht mehr länger vorenthalten. Es handelt sich hierbei um die Tastenkombination:

<CTRL> <RSHFT> <ALT>

<RSHFT> steht selbstverständlich für **rechte SHIFT-Taste**. Bei den ersten paar mal ist diese Dreierkombination sicherlich noch eine Fingerturnübung, aber diese Funktionen sollen ja absichtlich nicht "aus Versehen" aufgerufen werden können. Die Tastenkombination stellt eine Ein/Aus-Schaltung für unsere mit Sonderfunktionen belegten Tasten dar. Haben Sie diese Tastenkombination einmal betätigt, so wird sich die Cursorfarbe nicht mehr ändern, wenn Sie die <CTRL>-Taste mit einer der Zifferntasten der oberen Tastenreihe betätigen. Ferner wird es Ihnen auch nicht mehr möglich sein, die Hintergrundfarbe des 80-Zeichen-Bildschirmes zu verändern. Auch die HELP-Taste (Sie könnes es ausprobieren) ist nun außer Funktion. Nach nochmaligen Betätigen dieser Tastenkombination sind die Sondercodes wieder zugelassen.

VII.8.2 Der Hexcode-Editor

Wir wollen zunächst einmal die ASCII-Werte einiger Tasten betrachten und natürlich auch verändern. Fangen wir doch direkt einmal mit der Leertaste an: Es liegt nahe, da wir alle den ASCII-Code der Leertaste kennen. Also, betätigen Sie nun folgende Tastenkombination, die der Editierung des ASCII-Codes einer Taste ermöglicht. Sie können auf diese Weise also auch eine deutsche Tastatur simulieren!

<CTRL> <RSHFT> <CURSOR-LINKS-TASTE>

Ich habe bereits erwähnt, daß es sich hierbei ausschließlich um die CURSOR-LINKS-Taste im speziellen Cursor-Tastenblock handelt und brauche dies bestimmt nicht noch einmal zu wiederholen, oder? Nachdem Sie auch die Fingerübung ohne gebrochene Finger überstanden haben, erscheint in der Statuszeile ein kleines, hell unterlegtes Fenster. Sie werden nun aufgefordert, die Taste zu betätigen, deren ASCII-Wert Sie einsehen und gegebenenfalls ändern wollen. Natürlich sind auch hier die Kombinationen mit der <CTRL>- und der SHIFT-Taste erlaubt, anders können Sie ja nicht die drei anderen Tastenwerte erreichen. Lassen Sie uns der Aufforderung nachkommen und betätigen Sie einfach die Leertaste.

Im Fenster erscheint der *hexadezimale* ASCII-Wert der betätigten Taste, im unseren Beispiel die Leertaste mit dem ASCII-Wert 32 gleich **20** in hexadezimaler Schreibweise. Sie sehen, daß ein Cursor im Fenster blinkt. Das ist das Zeichen dafür, daß CP/M nun eine Eingabe von Ihnen erwartet. Es sind alle Ziffern und die Buchstaben A-F erlaubt. Jede andere Taste bricht den Eingabemodus sofort ab - der angezeigte Wert wird nicht verändert und das Fenster verschwindet.

Ansonsten können Sie den Wert der Taste ändern. Haben Sie beide Stellen des zweistelligen Codes geändert, wird der Code automatisch abgespeichert, es ist also nicht notwendig, die RETURN-Taste zu betätigen.

Sie können sich dies ganz einfach verdeutlichen, indem Sie aus dem Code 20 ganz einfach den Code 21 machen. Wählen Sie also mit <CTRL><RSHFT><LINKS-CURSOR> und <Leertaste> den ASCII-Wert der Leertaste an. Es erscheint der aktuelle Wert, die 20. Danach betätigen Sie einfach nacheinander die Tasten 2 und 1. Warten Sie einen ganz kleinen Augenblick, denn es dauert ein wenig, bis das System den neuen ASCII-Wert entschlüsselt und in der Tastaturmatrix abgelegt hat. Dann verschwindet das Fenster auch schon und Sie können daran erkennen, daß die Eingabe abgeschlossen ist. Zur Kontrolle erscheint ja auch jede eingegebene Taste rechts vom alten ASCII-Wert auf dem Bildschirm. Haben Sie sich verschrieben, einfach auf die RETURN-Taste oder eine andere beliebige Taste drücken, Sie verlassen dann ohne jede Veränderung den Editmodus.

Kommen wir aber zu unserer Änderung zurück: Haben Sie schon einmal die <LEERTASTE> gedrückt? Sie haben sich bestimmt gewundert, daß nun kein Leerzeichen mehr, sondern ein Ausrufungszeichen auf dem Bildschirm erscheint. Auf diese Weise können Sie alle Tastenwerte ändern, auch die der RETURN-Taste beispielsweise. Da ein Ausrufungszeichen aber auf der Leertaste ziemlich unpassend liegt, wollen wir doch den Code wieder in 20 umändern. Sie wissen nun ja, wie es geht, nicht wahr?

Eine sinnvolle Anwendung fiel mir sofort ein, als ich den Nutzen dieser Möglichkeit begriff. Wenn ich die SHIFT-Werte der Zehnertastatur so abändere, daß ich nicht nur acht, sondern alle sechzehn Hintergrundfarben für den 80-Zeichen-Bildschirm habe, so ist dies soch sicherlich schon sinnvoll.

Dazu muß man also alle 8 SHIFT-Werte der Zehnertastatur mit den Werten #B8 bis #BF belegen, da die Tastenwerte <CTRL>-1 bis <CTRL>-8 mit den ASCII-Werten #B0 bis #B7 belegt sind. Haben Sie dies getan, so stehen Ihnen alle möglichen sechzehn Farben als Hintergrundfarben für den 80-Zeichen-Bildschirm zur Verfügung. Natürlich können Sie sich so als 40-Zeichen-Bildschirm-Benutzer diese Tasten für die 40-Zeichen-Farbcodes programmieren. Sie brauchen die entsprechenden Codes lediglich der Tabelle zu entnehmen. Was für uns nun noch interessant ist, sind die Codes #80 bis #8F sowie #F0 bis #FF. Dieser Codes wollen wir uns nun annehmen.

VII.8.3 Der String-Editor

Ich habe es bereits angekündigt: Man kann einzelne Tasten auch mit Strings - das sind Zeichenketten - belegen. Es bleibt noch eine der drei Sonderfunktionstasten übrig. Dies ist die Tastenkombination:

<CTRL> <RSHFT> <CURSOR-RECHTS-TASTE>

Haben Sie diese Tastenkombination vollbracht, so erscheint ein deutlich größeres Fenster in der Statuszeile. Kleine Anmerkung des Autors: Sie haben es vielleicht schon gemerkt: CP/M wählt für Fensterhintergrund- und Zeichenfarbe immer zwei Farben aus, die sich vom Hintergrund abheben. So ist jederzeit eine korrekte Eingabe und ein Erkennen des Fensters gewährleistet.

Ähnlich wie beim Editor der Hexcodes erscheint also auch beim String-Editor ein Fenster in der Statuszeile. Sie können es sich schon denken: Sie müssen irgendeine Taste

betätigen. Sollten Sie sich eine aussuchen, die einen ASCII-Wert kleiner als #80 oder größer #9F besitzt, so verschwindet das Fenster wieder und der Modus ist beendet. Erwischen Sie aber eine Taste, die einen Code zwischen #80 und #9F aufzuweisen hat, so wird der aktuelle Stringinhalt des Strings auf dem Bildschirm angezeigt. Wir kennen eine Taste, die mit einem Text versehen ist - richtig, es ist die HELP-Taste. Also drücken Sie einmal sanft auf diese HELP-Taste, und schon füllt sich das Fenster mit dem Textinhalt:

```
>Help <
```

erscheint es auf dem Bildschirm. Das Größerzeichen (>) stellt die linke Begrenzung der Zeichenkette dar. Das Kleinerzeichen (<) entsprechend die rechte Grenze. Der Cursor steht auf dem ersten Zeichen und blinkt. Ich kann Ihnen nur empfehlen, die Tastatur zunächst in Ruhe zu lassen. Wir wollen nämlich zuerst einmal klären, wie wir den Cursor bewegen, Stellen löschen und einfügen können. Dazu soll folgende Tabelle dienen:

<CTRL> <RSHFT> & eine der folgenden Tasten:

```
RETURN - Abschluß der Eingabe
-->    - Cursor rechts im Fenster (Taste im Cursorblock)
<--    - Cursor links im Fenster (Taste im Cursorblock)
 +     - An Cursorposition eine Stelle einfügen
 -     - Zeichen unter Cursor löschen
```

Nun kann es ja losgehen. Man gewöhnt sich an diese Form des Editierens. Sollten Sie beispielsweise die RETURN-Taste ohne das vorgeschriebene <CTRL> und <RSHFT> betätigen, so wird

dieses RETURN gespeichert und auch ausgeführt, wenn diese Tastenfunktion einmal zur Ausführung kommen sollte - also Vorsicht! Sollen Sie einfach drauf losschreiben, so werden die Zeichen unter dem Cursor *überschrieben.*

Haben Sie die rechte Grenze des Fensters erreicht und schreiben dennoch weiter, so fügt das System automatisch eine Stelle ein und verschiebt somit die rechte Grenze um eine Stelle nach rechts.

Ihnen stehen auch mehr Zeichen zur Verfügung, als das Fenster in der Statuszeile vermuten läßt. Das Fenster verfügt über 22 Zeichen, sollten Sie oft genug eine Leerstelle einfügen, so können Sie ja einmal mit dem Cursor an die rechte Grenze wandern. Sie werden sehen, daß der Fensterinhalt gescrollt wird. Sollten Sie dann die wirkliche rechte Grenze des Strings erreichen, dann springt der Editor wieder an die erste Stelle des Strings.

Probieren Sie die einzelnen Funktionen der Cursorbewegungsfunktionen und der Lösch- sowie Einfügemöglichkeiten aus. Auf diese Weise lernen Sie am besten, mit den Tasten umzugehen. Bedenken Sie auch, daß es im CP/M-Modus keine Repeat-Funktion gibt, das Festhalten einer Taste führt also nicht automatisch zum Wiederholen dieser. Damit Sie Bescheid wissen, auf welchen Tasten breits Strings definiert wurden:

```
F1:    "F1"              F5: "F5"
F2:    "F2"              F6: "F6"
F3:    "DIR<CR>"         F7: "3 June 85<CR><LF>"
F4:    "DIR "            F8: "F8"
HELP:  "Help "
```

Sie sehen, daß hauptsächlich die Funktionstasten vorbelegt sind. Bedenken Sie, daß *32* Strings auf die Tasten verteilt werden können - es ist beispielsweise durchaus sinnvoll, den Funktionstasten, die man mit der <CTRL>-Taste zusammen anspricht, andere Strings zuzuordnen, so daß man schon eine Dreifachbelegung bei den Funktionstasten hat. Bedenken Sie auch unbedingt, daß der erzeugte Code bei eingeschaltetem CAPS-LOCK-Modus, den Sie nur durch Betätigen der Commodore-Taste einschalten können, wiederum ein zusätzlicher Code generiert wird. Doch hier sollte man keine Umdefinierung vornehmen, da es wohl nur zu Mißverständnissen führen dürfte. Hier noch eine Liste der Vorbelegungen der Strings, so wie sie das System beim Booten erstellt:

```
80: "F1"                          90: "F17"
81: "F2"                          91: "F18"
82: "dir<CR>"                     92: "F19"
83: "dir "                        93: "F20"
84: "F5"                          94: "F21"
85: "F6"                          95: "F22"
86: "F7"                          96: "F23"
87: "3 June 85<CR><LF>"           97: "F24"
88: "F9"                          98: "F25"
89: "F10"                         99: "F26"
8A: "F11"                         9A: "F27"
8B: "<SCREEN LEFT>"               9B: "F28"
8C: "<SCREEN RIGHT>"              9C: "F29"
80: "<SCREEN LEFT>"               90: "F30"
8E: "<SCREEN RIGHT>"              9E: "F31"
8F: "F16"                         9F: "Help "
```

Die Codes 8B bis 8E sind scheinbar leer geblieben, jedoch sind sie mit Steuercodes für den 40-Zeichen-Schirm belegt. Sie dienen dem horizontalen Scrolling des 40-Zeichen-Bildschirmes.

Wie Sie der ersten Tabelle entnehmen können, steht neben dem ASCII-Bereich F0 bis FF das Wort "Spezialfunktionen". Diese Spezialfunktionen sind beispielsweise das Rebooten - also das Neustarten - des Systemes, wie es durch Betätigen der <CTRL>- und der <ENTER>-Taste gemeinsam erreicht wird.

Gehen Sie in den Hexcode-Editor, und Sie werden sehen, daß diese Taste den ASCII-Code FF hat. Hier eine Aufstellung der Sonderfunktionen, soweit diese laut CP/M definiert sind:

F0: Ein/Ausschalten des Diskstatus
F1: Anzeige anhalten/fortführen (wie <NO SCROLL>)
F2: 40-Zeichen
F3: Bildschirm links (40 Zeichen)
F4: Bildschirm rechts (40 Zeichen)
F5: MFM Unlock
FF: Rebooten des Systemes

Drei dieser sieben Funktionen liegen auch bereits auf verschiedenen Tasten. Die Funktion F0 befindet sich in der Tastenkombination <CTRL> <RUN/STOP> und bewirkt, daß die Diskettenstatusanzeige in der Statuszeile entweder zugelassen oder unterbunden wird. Die zweite Funktion F1 wird durch die <NO SCROLL>-Taste hervorgerufen. Beim ersten Mal drücken erscheint *Pause* in der Statuszeile und die Anzeige wird angehalten. Erst wenn Sie die Taste erneut betätigen, wird mit der Ausgabe bzw. Ausführung fortgefahren. Die letzte Funktion FF befindet sich, wie bereits erwähnt, auf der Taste <ENTER> in Verbindung mit der <CTRL>-Taste.

VII.9 KEYFIG und wie man es nutzt

All diese eben beschriebenen Möglichkeiten sind resident, d.h. Sie können Tastatur und Strings nach Belieben jederzeit ändern - allerdings nicht sehr komfortabel, wie man zugeben muß. Doch hierzu gibt es das transiente Kommando KEYFIG, das alle die Leiden in komfortabler Menütechnik dem Benutzer vereinfacht. Geben Sie KEYFIG einmal ein, das Programm meldet sich direkt:

```
C128 SOFT KEYBOARD PROGRAM
      3 June 1985


Welcome to the Commodore C128 Keyboard
Definition program.  Do you want help?
```

Ein solches Angebot sollte man wirklich *nie* ausschlagen, wenn man ein Programm zum ersten Mal aufruft. Deswegen antworten wir auch mit einem "Y" für Ja. Auf dem Bildschirm erscheint eine ganze Liste an Hilfsangeboten:

```
Help is available on the following
topics:
--> done help <--
--> General Usage <--
--> Setting up your work file <--
--> What to do with your work file <--
--> Key values <--
--> Selecting a key to edit <--
--> Logical/Physical colors <--
--> Editing keys <--
--> Assigning/Editing Strings <--
```

```
--> Assigning Colors <--
--> Assigning Special Functions <--
--> Assigning HEX values <--
--> Finishing up <--
--> For experts only <--
Use the up and down arrow keys to scroll
through the menu; type the return key to
select the topic on which you want help.
```

Die Erklärungen ergeben sich fast schon - dennoch sollten Sie sich die verschiedenen Hilfstexte einmal ansehen, meistens dienen sie dem allgemeinen Verständnis. Sie können mit der Cursor-Rauf- und der Cursor-Runter-Taste das unterlegte Feld herauf und herunter bewegen; dies geht aber ausschließlich mit den Tasten im Cursorblock. Benutzen Sie diese Tasten bitte auch bei späteren Auswahlen. Die RETURN-Taste dient zur Bestätigung des untermalten Feldes. Der letzte Menüpunkt sagt Ihnen als "Experte", was Sie eben schon gelernt haben: das alles auch viel schneller mit der <CTRL> <RSHFT>-Kombination funktioniert. Haben Sie sich genug helfen lassen, dann sollten Sie den ersten Menüpunkt "done help" anwählen, um in das ursprüngliche KEYFIG-Programm zu gelangen.

Sollten Sie Ihre Tastatur umbelegen oder Strings umdefinieren oder ganz neue kreieren, die deutsche Tastatur durch Austauschen der Z- und Y-Taste simulieren wollen etc., so soll die Arbeit ja nicht immer wieder neu gemacht werden. Es wäre also wünschenswert, die geänderte Tastaturbelegung abzuspeichern. Genau dies können Sie mit dem KEYFIG-Programm unter anderem tun. Nachdem Sie den HELP-Modus verlassen haben, erscheint folgendes auf dem Bildschirm Ihres C 128:

```
From which of the following  sources of
key definitions would you like to work:

Default definitions
Definitions on the CP/M boot disk
Current definitions
(Your previous work file)
```

Sie können nun auswählen, welche Tastaturbelegung als Grundlage für die zukünftige Arbeit dienen soll. Die Angabe in Klammern erscheint dabei nur, wenn Sie KEYFIG nicht zum ersten Male starten. Sie können also auswählen, ob Sie:

```
Die Standardbelegung der Tastatur
Die Tastaturbelegung von CP/M-Diskette holen
Die momentane Tastaturbelegung
Die letzte Tastaturbelegung
```

bearbeiten wollen. Je nachdem holt sich das System die Tastaturbelegung von der Diskette oder auch nicht. In der dritten Zeile wird angezeigt, welche Tastatur bearbeitet wird.

Wenn Sie einen Einblick gewinnen wollen, wie komfortabel dieses KEYFIG-Utility ist, dann brauchen Sie nach der Anzeige der folgenden Menüpunkte

```
Edit a key definition
Set up logical<-->physical colora
Exit and save your work file
```

lediglich den zweiten Menüpunkt auszuwählen. Sie erhalten sehr eindrucksvoll (nachdem Sie den 80-Zeichen-Bildschirm quittiert haben) eine Farbpalette auf dem Bildschirm, die logische und physikalische Farben darstellt. Sie können durch einfache Buchstabeneingabe diese Zuordnung leicht verändern. Die Korrekturen werden umgehend angezeigt.

Um aber noch kurz zu erläutern, wie Sie die Tastenbelegungen verändern können: Wählen Sie den Menüpunkt "Edit a key definition" an. Sie sollten übrigens nie die Tastaturbelegung abspeichern, wenn Sie noch nicht ganz fertig sind, das hat Zeit. Auf dem Bildschirm erscheint:

```
Editing:        no key

This key has the 4 values shown below.

normal --->
CMDR SHFT->
SHIFTED -->
CONTROL -->
(done editing-exit and save work file)
```

Auch hier können Sie mittels den Cursortasten im Cursorsonderblock mit dem Markierungsfeld umherfahren. Wenn Sie eine Taste betätigen, so werden die vier Belegungen der Taste auf dem Bildschirm angezeigt. CMDR SHFT bedeutet übrigens den CAPS-LOCK-Modus, den Sie durch einfaches Betätigen der Commodore-Taste einschalten.

Besonders eindrucksvoll finde ich, daß KEYFIG beispielsweise für die Links-Taste ganz oben links auf der Tastatur den Text "LEFT ARROW NEXT TO 1" ausgibt oder für das Pfundzeichen "BRITISH POUND" etc. Eine Taste, die kein

druckbares ASCII-Zeichen darstellt, wird also mit Text umschrieben. Auch werden für nichtdruckbare Codes wie etwa Delete Texte ausgegeben, im Beispiel Delete ist es "RUBOUT". Wollen Sie einen dieser vier Tastenbelegungen ändern, so bewegen Sie das Markierungsfeld dorthin und betätigen die RETURN-Taste.

```
ASSIGN a STRING (more than 1 character)
ASSIGN  new (single) character
ASSIGN a COLOR
ASSIGN hex value
ASSIGN a SPECIAL FUNCTION
don't modify this key
```

Mußten wir bei der Drei-Tastenkombination in unserer Tabelle nachschauen, welchen Code wir zuteilen wollten, so nimmt uns KEYFIG diese Arbeit ab. KEYFIG ist also wesentlich komfortabler, für den versierten Anwender könnte die Krähenfußtaktik (wegen der gespreizten Hand) aber als wesentlich schneller bezeichnet werden, da man sich nicht mit den Cursortasten jedesmal die gewünschte Funktion heraussuchen muß. Entscheiden müssen Sie! Das Abspeichern allerdings auf Diskette kann Ihnen nur KEYFIG abnehmen. Hier können Sie sich sogar aussuchen, ob die neue Tastaturbelegung auf einer separaten Diskette oder auf der CP/M-Systemdiskette abgespeichert werden soll. Speichern Sie die Belegung auf der CP/M-Systemdiskette ab, so wird diese nach dem Booten automatisch geladen. Um abzuspeichern, müssen Sie ab dem Hauptmenü folgende Menüpunkte auswählen:

Exit and save your work file
on CP/M boot disk

Nach dem Abspeichern fragt Sie KEYFIG ganz lieb:

```
Do you want to do anything else [ ]
```

Diese Frage können Sie mit Y für Ja und N für Nein beantworten, je nach Bedarf. Sollten Sie mittels <CTRL>-C das Programm verlassen wollen, so wird hier auch eine Sicherheitsabfrage gemacht, damit Sie KEYFIG ncht unversehentlich verlassen.

VIII. Die "Additional Utilities"

Erschrecken Sie bitte nicht, aber beim Kauf des Commodore 128 hat Ihnen die Firma Commodore noch etwas vorenthalten. Dies sind recht wichtige Hilfsprogramme (Utilities), die Sie zusätzlich bei Commodore bestellen können, allerdings gegen ein Entgeld. Zusätzlich erhalten Sie eine Beschreibung zu CP/M, das im Handbuch ja vollkommen zu kurz kommt.

Ob sich der Kauf dieses Paketes lohnt, können Sie nicht zuletzt auch diesem Kapitel entnehmen, da wir hier auf die verschiedenen Utility-Programme eingehen wollen. Für einen "normalen" Anwender, so kann man schon kurz aber deutlich sagen, lohnt sich ein Kauf mit an Sicherheit grenzender Wahrscheinlichkeit nicht, denn: Eine "Bedienungsanleitung" zu CP/M haben Sie ja jetzt (in den Händen); die Programme, die sich auf der "Additional Utility"-Disk befinden, sind Assembler, Disassembler, Monitor und andere Hilfsprogramme, die für den engagierten Programmierer unter CP/M bestimmt sind; man könnte dieses Paket also auch recht gut *Entwicklungspaket* nennen.

Zunächst also einmal die ganz "normale" Anwendung der im "Additional Utility"-Paket mitgelieferten Programme; man könnte dieses Kapitel also auch mit

CP/M für Fortgeschrittene

bezeichnen. Später gehen wir dann noch auf einige Spezialitäten des Paketes ein.

VIII.1 Der Assembler MAC und RMAC

MAC und RMAC sind beides Assembler und sind Nachfolger des DIGITAL RESEARCH Assemblers ASM. MAC unterscheidet sich von ASM dahingehend, daß MAC ein Makroassembler ist. Assembler-Freunde wissen dies sicherlich zu schätzen. RMAC ist nun eine Spezialität von MAC; RMAC erzeugt einen relokablen Code. Dieser Zungenbrecher bedeutet auf deutsch, daß er einen frei verschieblichen Code erzeugt, den man mit dem Programm LINK nachbehandeln muß. Mit RMAC werden die dubiosen REL-Dateien erstellt. Später mehr. Alle drei Assembler assemblieren ausschließlich 8080-Mnemocode. Allerdings hat sich Commodore hier scheinbar etwas einfallen lassen: Man hat zwei LIB-Dateien erstellt, die eine Menge Makros enthalten. Die eine LIB-Datei heißt **Z80.LIB**, die andere hat den verheißungsvollen Namen **X6503.LIB**. Ihre jetzt entfachten Erwartungen werden zumindest teilweise befriedigt: Durch diese beiden optionalen LIB-Dateien können Sie mit MAC und RMAC auch Z-80-Mnemocodes assemblieren, allerdings sehen die meisten Mnemocodes eher aus wie 8080-Mnemocodes. Ein LD HL,0 gibt es hier nicht, das heißt in 8080-Stil LXI H <nn>. Es sind also "lediglich" die zusätzlichen Z-80-Codes hinzugefügt worden, wie beispielsweise die relativen Sprünge, die Register IX und IY oder die Austauschregister. Wie diese Mnemocodes auszusehen haben, können Sie der Datei Z80.LIB durch ein

```
TYPE Z80.LIB
```

schnell und problemlos entlocken. Dasselbe gilt für 6502-Programmierer. Auch hier hat man sich - mit einigen kleinen MAC-spezifischen Eigenheiten - eine Makro-Datei erdacht, die einem das Programmieren in 6502 ermöglichen soll. Man sollte den guten Willen und die gute Idee honorieren.

Es gibt also zwei Assembler, darüber haben Sie gerade gelesen. Wir wollen uns zunächst mit dem einem beschäftigen, dem MAC-Assembler.

Für diejenigen unter Ihnen, die es noch nicht so genau wissen: Der 8080 ist älter als der Z-80 und gewissermaßen sein Vater. Der Z-80 versteht zwar die Programme, die auch sein Vater versteht, er hat sogar einen weit größeren Sprachschatz als dieser, aber er hat einen anderen Mnemocode. Nein, wir haben uns nicht verschrieben. Mnemocode nennt man die Schreibweise, in der Maschinenprogrammierer Ihre Maschinenprogramme kodieren. Sie wissen sicherlich, daß Maschinenprogramme ausschließlich aus Zahlen bestehen. Beispielsweise versteht der Z-80-Prozessor das Kommando &41 und kann es ausführen. Für einen Menschen ist es allerdings sehr schwer, sich hierunter etwas vorzustellen. Dazu hat man die Mnemocodes entwickelt, die diese Maschinencodes zeitweise ersetzen sollen. Beim Z-80 sieht dieser Mnemocode für das Kommando &41 so aus:

```
LD B,C
```

Hierunter kann sich der Maschinenprogrammierer sehr wohl etwas vorstellen, und es erleichtert die Arbeit doch sehr, wenn man nicht immer in irgendwelchen Tabellen nachschlagen muß, was der Code &41 beispielsweise für Konsequenzen hat.

Der Assembler hat nun die Aufgabe, diese Mnemocodes, die der Mensch "so gut" versteht, in maschinenverständliche Codes zu übersetzen. Der Prozessor beispielsweise kann mit dem Mnemocode LD B,C nun überhaupt nichts anfangen. Hier tritt der Assembler als Dolmetscher oder Schnittstelle hervor, der die Menschensprache in Maschinensprache übersetzt. Der Assembler unterstützt den Menschen bei seiner Arbeit aber

noch durch einige weitere Hilfsmittel, auf die ich später kurz eingehen will.

Sie wissen jetzt also, was ein Mnemocode ist, und daß der ältere 8080-Prozessor der Vater des Z-80 ist. Ferner ist Ihnen bekannt, daß die Z-80 mehr Befehle beherrscht als die 8080. Alle CP/M-Programme müssen aber in 8080-Code programmiert sein, weil es auch CP/M-Rechner gibt, die mit einem 8080 bestückt sind. Das bedeutet nichts anderes, als daß sich ein Z-80-Programmierer lediglich auf die Kommandos beschränken muß, die der 8080 auch verstehen kann (und dabei auf einige wirklich bemerkenswerte Kommandos des Z-80 zu verzichten hat). Wollen Sie ausschließlich für die heute geläufigeren Z-80-CP/M-Rechner programmieren, so können Sie sich der oben erwähnten Z80-LIB bedienen. Das ist aber leider nicht alles, denn man hat auch die Mnemocodes bei der Weiterentwicklung des 8080 zum Z-80 geändert. Erinnern wir uns an unser Kommando $41. Es passiert in beiden Prozessoren bei der Erkennung und Ausführung dieses Kommandos dasselbe: Es wird der Inhalt des C-Registers in das B-Register kopiert. Der Mnemocode sieht beim Z-80 wie folgt aus:

```
LD B,C
```

Beim 8080 sieht dasselbe (!!) Kommando so aus:

```
MOV B,C
```

Sie erkennen sicherlich schon die Problematik: Ihnen nutzen Ihre Z-80-Kenntnisse wenig, wenn Sie einen 8080 programmieren wollen, selbst wenn Sie sich auf die Kommandos beschränken, die der 8080 auch verstehen kann. Also: Den ASM-

Assembler können Sie nur mit Mnemocodes füttern, die ein 8080-Assembler auch verstehen kann.

Doch wollen wir mal diesen Assembler näher betrachten, der es uns ermöglicht, eigene COM-Files zu programmieren.

VII.2 Die Bedienung des Assemblers MAC

In der Programmierung der Maschinensprache tritt häufig das Problem auf, daß man sich auf bestimmte Speicherstellen beziehen muß, beispielsweise um Register zu laden oder abzulegen. Weiterhin kommt es auch sehr häufig vor, daß man an eine Speicheradresse springt etc. Stellen Sie sich einmal vor, Sie hätten ein Programm geschrieben, daß an der Speicheradresse $5000 beginnt, und wollen es nun an die Adresse $4000 verschieben. Sie müßten alle Sprungadressen außer den relativen Sprüngen ändern. Oder wenn Sie ein Kommando ins Programm einfügen, verschieben sich alle relativen Sprünge und alle folgenden direkten Sprünge. Spätestens dann tritt der Fall ein, daß der Programmierer durchdreht. Um diese Arbeiten zu erleichtern, hat man bei Assemblern die Möglichkeit eingeräumt, Marken (Labels) zu definieren, die während der Assemblierung (das ist die Übersetzung der Mnemocodes in Maschinensprache) in das Programm eingesetzt werden. Fortan ist es kein Problem mehr, ein Kommando an einer beliebigen Stelle einzufügen oder eine Routine oder gar das gesamte Programm zu verschieben - der Assembler übernimmt die unangenehmen Arbeiten schon für Sie. Wie ein solches Assembler-Programm aussieht, wollen Sie wissen? Für den Additional Utility Paket-Besitzer ist das gar kein Problem. Auf dieser CP/M-Diskette wurden die Assemblerprogramme MAC und RMAC aufkopiert, mit denen wir auch die Handhabung des Assemblers üben wollen. Sehen Sie

sich das Inhaltsverzeichnis Ihrer Additional-Utility-Diskette einmal genau an.

```
A: LIB      COM : LINK     COM : MAC      COM : RMAC     COM : SID      COM
A: CBDOS3   SPR : BNKBDOS3 SPR : CRESBDOS3 SPR : HEXCOM   COM : XREF     COM
A: CALLVERS ASM : DUMP     COM : ZECHOVERS ASM : RANDOM   ASM : HIST     UTL
A: TRACE    UTL : READ     ME
```

Wir haben ein File mit Namen DUMP.COM; dies ist, wie wir bereits wissen, ein COM-File, das wir unter CP/M starten können, indem wir lediglich den Namen eintippen:

DUMP <Filename>

wäre die Syntax in diesem Fall. Dump liefert Ihnen einen Ausdruck der angegebenen Datei in HEX-Format (Hex-Dump). Probieren Sie es ruhig einmal aus:

DUMP HEXCOM.COM

Auf dem Bildschirm erscheint dann der Hex-Dump der COM-Datei HEXCOM, das Programm gewissermaßen. Sie können die Ausgabe mit <Ctrl>/S anhalten und mit <Ctrl>/Q wieder mit der Ausgabe fortfahren. Mittels <Ctrl>/C können Sie die Ausgabe abbrechen. Ferner können Sie sich natürlich der NO-SCROLL-Taste bedienen, die weitaus praktischer als die <CTRL>-Sequenzen ist.

DUMP.ASM ist nun das Maschinenprogramm von DUMP, so wie es vom Programmierer kodiert worden ist. Sie können sich dieses File einmal auf dem Bildschirm (oder Drucker) ausgeben lassen. Geben Sie hierzu ein:

TYPE DUMP.ASM

Sie erkennen im Vorspann des Programmes, daß die Firma DIGITAL RESEARCH die Urheberrechte für dieses Programm besitzt. Das ist aber nicht weiter verwunderlich, da genau diese Firma ja auch CP/M entwickelt hat. Sie sehen auch einen weiteren wesentlichen Unterschied zwischen Assembler und Maschinensprache: Sie können Kommentare einfügen. Diese Kommentare beginnen mit einem Semikolon, damit der Assembler weiß, daß er ab dieser Position nicht mehr übersetzen muß. Ansonsten würde der Assembler ja natürlich versuchen, diesen Kommentar zu verstehen, sprich zu übersetzen, und könnte nur verständnislos den Kopf schütteln und sagen: Hier stimmt aber etwas nicht!! Auch wenn Ihnen dieses Programm sehr lang vorkommen sollte, glauben Sie es mir, es ist verdammt kurz. Wenn Sie bedenken, daß Betriebsysteme auch in Assembler kodiert werden:

Es gibt Betriebsysteme, die beim Ausdruck 2000 Seiten und mehr lang sind. Auch "kleinere" Programme, wie beispielsweise die Textverarbeitung TEXTOMAT auf Ihrem Commodore 128 kommen noch leicht auf 400 bis 500 Seiten. Sie sehen, daß die Programmierung in Maschinensprache eine sehr aufwendige Sache ist. Doch wollen wir dieses Assemblerprogramm einmal assemblierenm, damit Sie in Zukunft den Assembler selbständig nutzen können. Achten Sie darauf, daß Ihre Arbeitsdiskette jetzt nicht schreibgeschützt ist, denn wir müssen auch auf die Diskette schreiben, auf der sich der Quellcode befindet. Sie haben sicherlich schon eine Sicherheitskopie von Ihrer CP/M-Diskette gemacht?! (Wenn nicht, dann sollten Sie dies schleunigst tun.) Wir assemblieren nun "unser" Maschinenprogramm:

MAC DUMP

Hier erkennen Sie schon ein wichtiges Merkmal des Assemblers: Sie brauchen keine Extension einzugeben, d.h. Sie müssen nicht DUMP.ASM angeben, da der Assembler die Erweiterung .ASM selbständig anhängt. Auf Ihrem Bildschirm erscheint:

```
CP/M MACRO ASSEM 2.0
0257
002H USE FACTOR
END OF ASSEMBLY

A>
```

Schnell ist er, unser Assembler, nicht wahr? Er erstellt nun drei Dateien:

1) Ein Protokoll unter dem Namen DUMP.PRN
2) Ein Hex-Dump unter dem Namen DUMP.HEX
3) Eine Symboltabelle unter DUMP.SYM

Sehen Sie sich zunächst einmal mittels **TYPE DUMP.PRN** das Protokoll an. Hier wird das Assembler-Programm noch einmal aufgelistet, mit allen Informationen, die der Assembler hinzufügt. In der ersten Spalte steht immer die Adresse, an der gerade kodiert wird - es sei denn, es wird ein Label definiert: Dann steht in der ersten Spalte der definierte Wert des Labels. Das Programm beginnt also bei Adresse $0100 und endet bei $0257; folglich wissen wir jetzt auch, was der Assembler uns da für eine Zahl ausgegeben hat (s.o.). Fast alle CP/M-Programme beginnen an Adresse $0100. Nachdem Sie sich das Protokoll gut angesehen oder es vielleicht sogar zu Papier gebracht haben, sehen Sie sich noch die zweite Datei

an, die unser fleißiger Assembler erstellt hat. Dazu geben Sie wieder ein (Sie wissen es sicherlich mittlerweile):

```
TYPE DUMP.HEX
```

Sie sehen auf dem Bildschirm nun eine Reihe von Zahlen, die der Assembler erstellt hat; es handelt sich hierbei um die Programmcodes. Nebenbei werden noch einige weitere Informationen angezeigt, wie die Adresse, an der der Code später stehen soll etc. Diese Datei ist schon erheblich kürzer als unsere Datei DUMP.PRN, nicht wahr? Interessant vielleicht noch die Symboltabelle, die die Dateikennung .SYM trägt. Nur können wir dieses Maschinenprogramm immer noch nicht starten, es ist ja noch kein COM-File. Jetzt gibt es noch ein weiteres Programm, das diese Arbeit übernimmt. Es macht aus einer HEX-Datei eine COM-Datei, also ein von CP/M aufrufbares und startbares Programm. Dieses Programm trägt den Namen **HEXCOM**. Wir rufen dieses Programm einmal auf:

```
HEXCOM DUMP
```

Sie sehen, daß wir auch hier keine (Kennung) Extension angeben müssen bzw. dürfen. HEXCOM hängt automatisch die Extension ".HEX" an. Auf dem Bildschirm erscheint:

```
HEXCOM VERS: 3.00

FIRST ADDRESS 0100
LAST ADDRESS 0212
BYTES READ 0113
RECORDS WRITTEN 03
```

Jetzt ist es endlich soweit: Sie haben eine COM-Datei erstellt, die Datei DUMP.COM, das wir unter CP/M einfach mittels Eingabe von

DUMP DUMP.COM (DUMPt sich selber aus)

aufrufen können. Aber es ist nicht ganz einfach, COM-Dateien zu programmieren, da man sich dann auch sehr gut im CCP und dem BDOS auskennen muß, weil auf diese Routinen zurückzugreifen ist.

Ich will aber noch einige Informationen über den Assembler loswerden, damit Sie ihn auch richtig bedienen können (wenn Sie es wollen).

Die Quellprogramme für den Assembler müssen folgende Syntax aufweisen:

<Zeilennummer><Marke>:<Kommando><Argument>;<Kommentar>

Die <Zeilennummer> ist nur optional, d.h. muß nicht vorhanden sein (so in unserer Beispieldatei). Sie wird vom Assembler sowieso überlesen und nur vorgesehen, weil einige Editoren diese Zeilennummern automatisch einfügen, wie beispielsweise der Editor ED auf der CP/M-Diskette. Die <Marke>: muß natürlich auch nicht vorkommen, kann aber an dieser Stelle stehen. Unbedingt vorhanden sein muß in einer Zeile, die nicht ausschließlich aus einem Kommentar besteht, der <Befehl> und das <Argument>. Daran schließt sich dann eventuell noch der ;<Kommentar> an.

Der Assembler wandelt vor der Kodierung alle Kleinbuchstaben in Großbuchstaben um, so wie Sie es von CP/M ohnehin schon

gewohnt sind. Dies vereinfacht die Programmierung schon ungemein. Ausgeschlossen von dieser Regel sind Kleinbuchstaben, die in Anführungszeichen stehen und einen fixen Text darstellen, wie beispielsweise die Anweisung:

```
DB 'File Dump Version 1.4$'
```

Die einzelnen Komponenten der Assemblerzeile müssen wenigstens durch ein Leerzeichen getrennt sein. Es ist üblich, die <Marke> ganz links beginnen zu lassen und dann mittels eines Tabulators in der Textverarbeitung, mit der Sie die Assembler-Quellcodes schreiben, die verschiedenen Komponenten anzuspringen. Dies ist zwar, wie bereits erwähnt, nicht unbedingt nötig, dient aber der Übersichtlichkeit des Programmes ungemein, da man dann Gleiches immer unter Gleichem stehen hat. Ein Beispiel:

```
DISKR:    ;READ DISK FILE RECORD
          PUSH H! PUSH D! PUSH B
          LXI  D,FCB
          MVI  C,READF
          CALL BDOS
          POP B! POP D! POP H
          RET
```

Diese Passage finden Sie in unserem DUMP-Programm. DISKR ist ein Label, das durch einen Doppelpunkt markiert wird. Es folgt dann der <Befehl> und das <Argument>, immer untereinander. In einer Zeile haben wir auch einen Kommentar, geführt durch ein Semikolon ";".

Sie können in dieser kleinen Routine auch eine weitere Besonderheit des ASM-Assemblers erkennen: Verschiedene

Assembler-Kommandos lassen sich durch das Ausrufungszeichen trennen.

Man kann bei ASM Kommentare mit einem Semikolon kennzeichnen oder eine ganze Zeile als Kommentar definieren, indem man in der ersten Spalte mit einem Stern beginnt. Dies wurde von den Entwicklern von ASM eingeführt, um eine gewisse Kompatibilität zu den bestehenden 8080-Assemblern zu wahren. So kann man schön einzelne Routinen in den Assembler-Programmen mit einer Kopfzeile versehen, etwa:

```
*********************************************************
***          Read Disk File Record                    ***
*********************************************************
```

wäre ein Beispiel hierfür.

Achten Sie bei der Verwendung von Labels darauf, daß Sie keine festen Schlüsselwörter integrieren. Schlüsselwörter sind beispielsweise die Mnemocodes des 8080-Prozessors, also Wörter wie MVI, MOV, STA etc.

Im ASM-Assembler können Sie, wie in vielen anderen Assemblern auch, die momentane Position des Program Counters PC während des Assemblier-Vorganges bestimmen. Beispielsweise wird dies automatisch bei Programmlabels gemacht, wie in folgendem Beispiel:

```
posit: EQU $
```

Argumente, die im ASM vorkommen, können durch verschiedene arithmetische Operanden verknüpft werden, so beispielsweise:

```
MVI A,12+2*3
```

Arithmetische Operanden sind:

+X	Positive Zahl
-X	Negative Zahl, entspricht 0-X (Immer 16 Bit !)
X+Y	Addition zweier 16-Bit-Werte
X-Y	Subtraktion zweier 16-Bit-Werte
X*Y	Produkt von X*Y
X/Y	Division der Argumente
X MOD Y	Restfunktion der Division X/Y

Weiterhin können diese zwei Schiebebefehle verwendet werden:

X SHL Y	Verschiebt den 16-Bit-Wert X um Y Positionen nach links. Herausgeschobene Bits gehen verloren.
X SHR Y	Verschiebt den 16-Bit-Wert X um Y Postionen nach rechts. Herausgeschobene Bits gehen verloren.

Ferner gibt es noch die logischen Operatoren:

NOT X	Logische Negation des Argumentes X
X AND Y	Logische Und-Verknüpfung der Argumente X und Y
X OR Y	Logische Oder-Verknüpfung der Argumente X und Y
X XOR Y	Exklusiv-Oder-Verknüpfung der Argumente X und Y

All diese Möglichkeiten erscheinen Ihnen auf den ersten Blick vielleicht als überflüssig. Sind sie aber nicht, ganz im Gegenteil: Sie sind sogar sehr nützlich. Wenn Sie beispielsweise den Akkumulator mit dem höherwertigen Byte einer Adressemarke laden wollen, so können Sie das ganz einfach tun, indem Sie diese Marke um 8 Bits logisch nach rechts verschieben:

```
MVI A,Label SHR 8
```

Wenn Sie lediglich die unteren 8 Bits haben wollen, so müssen Sie ausdrücklich die oberen 8 Bits ausblenden, weil Sie sonst das 8-Bit-Register, den Akkumulator, mit einem 16-Bit-Wert laden würden. Diese Ausblendung sieht so aus:

```
MVI A,Label AND 0FFH
```

Diese Anwendung wurde als Beispiel genannt, weil sie wohl am häufigsten vorkommen dürfte, es sind natürlich tausende von anderen Anwendungen denkbar.

Weiterhin existieren noch die Pseudo-Opcodes

ORG, EQU, END, DS, DB, DW, SET, IF und ENDIF.

(Pseudo-Opcodes sind Assembler-Kommandos, die während des Assemblier-Vorganges vom Assembler gelesen und verarbeitet werden. Der Name "Pseudo-Opcodes" deshalb, weil diese Befehle, wie die normalen Opcodes, im Assembler-Programm vorkommen, nicht aber assembliert und von der CPU verstanden werden können.)

Wir wollen die einzelnen Pseudo-Opcodes nicht näher erläutern, lediglich kurz die Wirkung und Funktionsweise erwähnen, da dies nicht die Aufgabe eines CP/M-Buches sein sollte.

ORG <Startadresse>

Dieses Kommando definiert die Startadresse des zu assemblierenden Programmes. ORG sollte immer das erste Kommando in einem Programm sein.

EQU <Wert>

EQU ordnet einem Label (einer Marke) einen festen Wert zu. Der <Wert> kann auch ein arithmetischer Ausdruck sein.

Beispiel: **Diskan: EQU Diskr+055H**

DS

Dieses Kommando hält im Programm einen definierten Bereich frei, um beispielsweise Daten dort abzulegen. Der Programm-Counter wird entsprechend erhöht.

Beispiel: **Diskm: DS 100**

In diesem Beispiel werden 100 Bytes frei gehalten, Startadresse ist Diskm, Endadresse ist Diskm+99. Das nächste Kommando beginnt bei Diskm+100.

DB

Hier werden einzelne zu definierende Bytes im Speicher abgelegt. Die einzelnen Bytes müssen durch Komma getrennt sein. Es können auch Zeichenketten vorkommen.

Beispiele:

```
Disktxt: DB "Bitte Diskette einlegen"
Diskt2: DB 23,32,0,3,122
```

DW

DW definiert Wörter, also 16-Bit-Werte. Auch hier können die einzelnen 16-Bit-Werte durch Komma voneinander getrennt werden.

Beispiel:

```
Diskadr: DW 03AB9H,marke1
```

END

END definiert das Ende des zu assemblierenden Programmes. Für Sonderzwecke ist es möglich, hier auch eine Startadresse für die Abarbeitung des assemblierten Programmes anzugeben.

SET

Mit dem Kommando SET wird, ähnlich dem EQU-Kommando, ebenfalls einer Marke ein Wert zugeordnet. Allerdings mit einem Unterschied: Während beim EQU-Kommando die Zuordnung fix ist, können durch das SET-Kommando definierte Marken während des Assemblierens noch verändert werden. Durch SET definierte Marken werden während des Assemblierens auch nicht in der linken Spalte angezeigt.

Beispiel:

```
Marke1: SET Alfa
Marke1: SET Marke1 + 32
```

Doch damit sind wir noch nicht an die Leistungsgrenzen von MAC gestoßen. So wie viele andere leistungsstarke Assembler verfügt auch ASM über die Möglichkeit, konditioniert zu assemblieren, d.h. eine bestimmte Assemblerpassage nur unter bestimmten Bedingungen zu dekodieren und in Maschinensprache umzuwandeln. Dies erreichen Sie, indem Sie sich der Kommandos IF und ENDIF bedienen. (Allerdings werden diese Kommandos erfahrungsgemäß nur selten genutzt.)

Nehmen wir einmal an, Sie wollen ein Programm schreiben, um zwei Werte einzulesen, diese dann zu addieren bzw. miteinander zu multiplizieren. Der Algorithmus ist für beide Programme gleich. Sie können nun ein Programm schreiben, das während des Assemblierens ein Flag abfragt, ob das Programm addieren oder multiplizieren soll.

```
;
Multi  EQU  0 ;Programm soll addieren
ORG 0100H
;
;Lies Werte ein
CALL Einlesen
;
IF Multi
  CALL Multir ;Multiplikationsroutine aufrufen
ENDIF
IF NOT Multi
  CALL Addir  ;Additionsroutine aufrufen
ENDIF
CALL Ausgabe
END
```

Wollen Sie die Routine nun multiplizieren lassen, so setzen Sie in der ersten Zeile den Wert für Multi lediglich auf NOT 0, das ist alles.

Nachdem wir nun den Assembler so ausführlich behandelt haben, wollen wir uns noch einmal kurz dem Aufruf des ASM zuwenden:

Wie Sie sich erinnern, müssen Sie beim Aufruf von MAC nicht die Extension ".ASM" mit angeben. Es werden die drei Dateien <Dateiname>.PRN, <Dateiname>.HEX und <Dateiname>.SYM auf Diskette erzeugt. Dies kann man aber unterdrücken! Ferner kann man angeben, von welchem Laufwerk die zu assemblierende Datei geladen werden soll, bzw. auf welches Laufwerk die einzelnen zu erstellenden Dateien gebracht werden sollen. Es existieren die 5 Optionen: **A, H, L, P und S.** Werden Optionen angegeben, so ist ein Dollarzeichen als Merkmal zu setzen. Beispiel:

MAC <Datei> $HB PZ

Die einzelnen Buchstaben haben folgende Bedeutung:

A Laufwerk für .ASM-Datei (A-O)
H Laufwerk für .HEX-Datel (A-O, Z)
L Laufwerk für .LIB-Datel (A-O)
P Laufwerk für .PRN-Datei (A-O, Z, P, X)
S Laufwerk für .SYM-Datei (A-O, Z, P, X)

Als Laufwerksangabe sind die Laufwerksbezeichnungen A-O erlaubt. Ferner kann man bei einigen Dateien noch vorsehen, daß eine Datei gar nicht erstellt wird (Z-Option, für Zero), daß eine Ausgabe auf den Bildschirm erfolgt (X-Option) oder auf Drucker (P-Option).

Wird keine Option eingegeben, so werden die Laufwerksangaben auf das Standardlaufwerk bezogen und alle Dateien erstellt. Wollen Sie an dieser Standardeinstellung (sog. DEFAULT) etwas ändern, so geben Sie den entsprechenden identifizierenden Buchstaben und die Option direkt dahinter an. Ein Dollar-Zeichen ist obligatorisch. Zwei Optionen müssen durch ein Leerzeichen getrennt sein.

RMAC ist nun MAC von der Bedienungsstruktur recht ähnlich. Allerdings wird bei der Assemblierung keine HEX-Datei erstellt, sondern eine .REL-Datei, die noch mittels des Linkers LINK bearbeitet werden muß. Bei RMAC macht man die Angaben zu den .LIB-Dateien erst beim Linker. Entsprechend gibt es bei RMAC auch nur drei Optionen:

R	**Laufwerk für .REL-Datei (A-O, Z)**
S	**Laufwerk für .SYM-Datei (A-O, X, P, Z)**
P	**Laufwerk für .PRN-Datei (A-O, X, P, Z)**

Die Angaben in der zweiten Spalte haben dieselbe Bedeutung wie bei MAC. Auch bei RMAC sind eventuelle Optionen durch ein Dollarzeichen zu vermerken und die einzelnen Optionen müssen durch ein Leerzeichen getrennt werden.

Da wir in diesem Buch lediglich die Bedienungsstruktur von MAC und RMAC erläutern wollen, um Ihnen eine Kaufentscheidung zu geben, wäre es an dieser Stelle sinnlos, Ihnen auch noch einige Beispiele zu RMAC zu geben: Sie könnten Sie ja doch nicht ausprobieren. Sollten Sie RMAC aber schon haben, indem Sie das Additional Utility Paket erworben haben, so sind Sie ja auch glücklicher Besitzer einer Beschreibung. Wir wollen nun noch auf den Disassembler/Monitor SID eingehen, sowie auf die Möglichkeit, SUBMIT-Dateien zu erstellen.

VIII.3 Arbeiten mit SUBMIT

Häufig kommt es vor, daß man bestimmte Befehlsfolgen immer und immer wieder eingeben muß. Stellen Sie sich vor, Sie wollen eine bestimmte Quelldatei assemblieren, dann die HEX-Datei in eine COM-Datei umwandeln, und schließlich genau diese neue COM-Datei starten, um es auszuprobieren.

Genau dann ist es empfehlenswert, eine SUBMIT-Datei zu erstellen, die alle nötigen Befehle enthält und hintereinander automatisch ausführt. SUBMIT heißt nichts anderes als übergeben: Das Programm SUBMIT übergibt CCP, dem Console Command Processor, der für die Eingabe über Tastatur und die Ausführung der eingegebenen Kommandos zuständig ist, diese Kette von Kommandos. Dies geschieht so:

Wenn Sie SUBMIT sagen, welche Datei als SUBMIT-Datei übergeben werden soll, so erstellt SUBMIT eine Zwischendatei $$$.SUB. In dieser Zwischendatei stehen all die Dinge der SUBMIT-Datei, mit einigen Ergänzungen. Bevor CCP einen Befehl von der Tastatur übernimmt, wird nachgesehen, ob es nicht die temporäre Datei $$$.SUB auf dem angemeldeten Laufwerk gibt. Sollte dies der Fall sein, so wird eine Befehlszeile in den Buffer von CCP geladen, und diese Zeile aus der Datei gelöscht. Nachdem die Befehlszeile kopiert worden ist, wird der Befehl ausgeführt. Dies wird so lange wiederholt, bis die Datei $$$.SUB leer geworden ist, erst dann wird wieder ein Kommando von der Tastatur eingelesen.

(Da Sie in der zumeist englischen Fachpresse und -literatur niemals den Begriff *Datei* finden werden, wollen wir nun diesen Begriff durch *File* ersetzen).

Wie Sie wissen, wird Ihnen beim DIR-Kommando leider nicht ausgegeben, wieviel Platz noch auf der Diskette ist. Diese

Information erhalten Sie über das SHOW-Kommando. Wir wollen nun eine SUBMIT-Datei erstellen, die das DIR-Kommando und das SHOW-Kommando zusammenfaßt, um uns mit einem Befehl Auskunft über Kapazität und Inhalt der Diskette zu verschaffen. Dazu erstellen wir mit einer Textverarbeitung, beispielsweise mittels ED, diese SUBMIT-Datei und nennen Sie DISK.SUB. Alle Submit-Dateien müssen das Kürzel ".SUB" tragen, dies ist oberstes Gebot.

Also gut, unsere erste Submit-Datei sieht also so aus:

```
SHOW
DIR
```

Legen Sie hierzu am besten die zweite Seite der CP/M-Systemdiskette ins Laufwerk und rufen Sie ED auf:

```
ED DISK.SUB
```

ED wird sich umgehend melden; ED merkt, daß Sie eine neue Datei anlegen wollen und teilt dies auch brav mit:

```
NEW FILE
: *_
```

Der Cursor blinkt und ED erwartet Ihre Eingabe. ED ist sehr schwierig zu bedienen. Da wir hier nur zwei Zeilen eingeben, geht hoffentlich nichts schief, denn nichts ist schwieriger, als mit ED etwas zu korrigieren. Geben Sie nun I ein, um ED

zu signalisieren, daß Sie etwas in die Datei einfügen wollen. Es erscheint auf dem Bildschirm:

```
: *i
1: _
```

Geben Sie nun die beiden Kommandozeilen ein:

1: show a: (RETURN drücken)
2: dir (RETURN drücken)
3: (<CTRL>-Z drücken)

Nachdem Sie dies geschafft haben, sollten Sie ED durch Eingabe des E-Kommandos verlassen:

: *E (RETURN drücken)

ED speichert die Datei DISK.SUB nun auf Diskette ab. Sollten Sie ED nicht trauen, so können Sie sich die Datei einmal mittels des TYPE-Kommandos ansehen. Zugegeben: Es ist eine sehr kurze SUBMIT-Datei, aber es ist noch kein Meister vom Himmel gefallen! Lassen Sie uns diese erste SUBMIT-Datei einmal starten mit:

SUBMIT DISK

Sie sehen, daß wir nicht die Extension ".SUB" angeben dürfen. Unser Laufwerk hört sich nun sehr vielbeschäftigt an, aber wir wissen ja auch, was alles gemacht wird.

(Erstellen der $$$.SUB-Datei, kopieren des Inhaltes von DISK.SUB, zeilenweises Löschen der Kommandos aus der Datei $$$.SUB etc.) Und es klappt, es wird sowohl der Diskettenplatz als auch das Inhaltsverzeichnis auf dem Bildschirm ausgegeben.

Doch erinnern wir uns, was wir eigentlich geplant hatten. Wir wollten ein File assemblieren, es mittels HEXCOM-Kommando zum COM-File machen und es anschließend starten. Diese Reihenfolge der Kommandos sähe etwa so aus:

```
MAC <Filename> $<Options>
HEXCOM <Filename>
<Filename>
```

Die Begriffe <Filename> und <Options> stehen hier praktisch als Platzhalter, als Variablen - doch wie kann man diese Variablen füllen? Die Entwickler des SUBMIT-Programmes haben die Problematik erkannt und sich etwas einfallen lassen, um die SUBMIT-Dateien etwas komfortabler zu gestalten. Sie können beim Aufruf der SUBMIT-Datei diese Vorgaben übergeben, indem Sie diese Vorgaben durch mindestens ein Leerzeichen voneinander trennen. Diese Vorgaben können einfache Zahlen sein oder aber auch komplexe Begriffe wie etwa Filenamen. Die Vorgaben werden durchnumeriert, beginnend bei eins. In der SUBMIT-Datei muß man diese Vorgaben mit einem Dollarzeichen ($) markieren, direkt gefolgt von der Nummer der Vorgabe. Dabei können einzelne Vorgaben beliebig oft in der SUBMIT-Datei wiederholt werden. Bei der Erstellung des Files $$$.SUB werden dann die Platzhalter automatisch durch die Vorgaben ersetzt, so daß CCP diese selbstverständlich abarbeiten kann. Unsere SUBMIT-Datei sieht nun so aus:

```
MAC $1 $$$2
HEXCOM $1
$1 $3
```

Sie sehen, daß die erste Eingabe, also die erste Vorgabe, in der SUBMIT-Datei dreimal vorkommt. Wenn wir diese SUBMIT-Datei beispielsweise mit unserem Dump-Programm starten und kein Protokoll-File wollen (ein HEX-File muß erstellt werden, weil HEXCOM dieses ja braucht), so könnte der Aufruf so aussehen, wenn wir die SUBMIT-Datei unter dem Namen "ASSEM.SUB" gespeichert haben:

```
SUBMIT ASSEM DUMP AB DUMP.COM
```

Die letzte Angabe weist darauf hin, welches File ausgedumpt werden soll, in unserem Beispiel DUMP.COM selbst. Wenn Sie es ausprobiert haben und alles klappt, dann haben Sie den ersten Schritt zur Benutzung der SUBMIT-Dateien getan. Erster Schritt deswegen, weil SUBMIT noch mehr kann!

Vielleicht haben Sie sich schon gefragt: Was tun, wenn ein Dollarzeichen in der SUBMIT-Datei vorkommen soll oder muß? Ganz einfach: Es wird das gemacht, was man in einem solchen Fall meistens in der Computerei macht, man schreibt zwei Dollarzeichen hintereinander. In unserem Beispiel haben Sie ja schon gesehen, wie schnell es notwendig wird, ein Dollarzeichen in eine SUBMIT-Datei zu integrieren. Um beispielsweise eine temporäre Datei zu löschen, wäre normalerweise folgendes Kommando notwendig:

```
ERA *.$$$
```

In einer SUBMIT-Datei sieht das etwas erschreckender aus:

```
ERA *.$$$$$$
```

- bedeutet aber dasselbe. Da Dollarzeichen aber in der Regel recht selten vorkommen, ist der zusätzliche Arbeitsaufwand nur minimal.

Wenn eine Befehlszeile aus dem File $$$.SUB in den Speicher von CCP kopiert wird, wird diese Zeile auch erst einmal auf dem Bildschirm dargestellt. Danach wird die Tastatur abgefragt, ob eine Taste betätigt worden ist. Ist dies der Fall, oder wird das Kommando im Befehlsspeicher nicht oder nicht richtig ausgeführt, so bricht SUBMIT den Befehlsablauf ab und löscht das FILE $$$.SUB.

Innerhalb der SUBMIT-Dateien kann man aber nicht nur Angaben für den CCP machen, sondern auch Angaben für angesprochene Programme wie beispielsweise ED oder PIP. Diesen Eingaben müssen allerdings das Kleinerzeichen (<) vorangestellt sein, sonst geht's nicht. Wir wollen dies einmal an einer Anwendung mit PIP verdeutlichen.

```
PIP
<E:=A:*.COM
<
DIR *.COM
```

Sie sehen, daß die letzte Eingabezeile für PIP lediglich aus einem Kleinerzeichen besteht, um die transiente Routine PIP zu verlassen. In PIP ist diese Anwendung noch nicht so sinnvoll, weil man das Kommando direkt hätte anders gestal-

ten können, etwa: PIP b:=a:*.COM um alle COM-Files von Laufwerk A: auf Laufwerk B: zu kopieren, beim ED allerdings wird die Anwendung sehr interessant. Sie sehen, daß es reicht, in der ersten Spalte das Kleinerzeichen voranzustellen, um Eingaben von COM-Dateien Daten zu übegeben. Dies war unter CP/M 2.2 noch nicht so einfach. Angaben, die sonst also über Tastatur gemacht werden, kommen nun aus dem File $$$.SUB.

Ich hoffe, Sie haben die Anwendung und Handhabung von SUBMIT-Dateien soweit verstanden, daß Sie demnächst bei entsprechenden Bedarf eine eigene SUBMIT-Datei erstellen können.

Wir haben jetzt den Assembler und die SUBMIT-Dateien besprochen. Kurz erwähnt werden sollten noch ein Disassembler, der sich auch auf der Diskette befindet. Sie rufen diesen Disassembler mit SID auf. SID ist der unmittelbare Nachfolger der CP/M 2.2-Version DDT. Wollen Sie beispielsweise das Kommando-File HEXCOM.COM einmal disassembliert ausgedruckt bekommen, so geben Sie einfach ein:

```
SID HEXCOM.COM
```

Bevor Sie <RETURN> drücken, geben Sie natürlich noch mit <Ctrl>/P das Kommando, daß alles auf Drucker protokolliert werden soll.

Disassemblieren bedeutet das Gegenteil von Assemblieren: Es werden die Maschinencodes in menschenwürdigere Mnemocodes umgewandelt. SID ist aber nicht nur ein Disassembler, sondern auch Monitor. Doch disassemblieren Sie zunächst einmal HEXCOM durch Eingabe des Kommandos für Disassemblieren:

```
CP/M 3 SID - Version 3.0
NEXT MSZE PC END
0580 0580 0100 D2FF
#L
0100 JMP 0180
0103 NOP
#L180
0180 LXI H,0000
0183 DAD SP
```

Mit dem D-Kommando können Sie sich einen Bereich ausdumpen lassen und mit dem G-Kommando das Programm starten. Mittels A können Sie im Direkt-Assembler-Verfahren Quellcode eingeben. Es gibt noch einige weitere Kommandos, auf die wir hier nicht näher eingehen können. Auch diese können Sie dem Commodore-Handbuch entnehmen.

VIII.4 Die Speicherverteilung

Wir haben bereits einige Male Schlagwörter wie BDOS, CCP, BIOS etc. erwähnt. Es dürfte den meisten von Ihnen auch nicht verborgen geblieben sein, daß diese Begriffe sehr wichtig sind, zumal Sie diese Begriffe sicherlich schon hier und da einmal gehört oder davon gelesen haben. Sicherlich haben sich schon einige Leser Gedanken darüber gemacht, wo sich dies alles im Speicher des Rechners befindet. Lassen Sie uns eben repetieren, was wir unter den einzelnen Begriffen zu verstehen haben:

CCP

CCP steht für Console Command Processor. Dieser Bereich regelt die Eingabe von Tastatur und beinhaltet die residenten Befehle, also: DIR, ERA, REN, SAVE, TYPE und USER.

BDOS

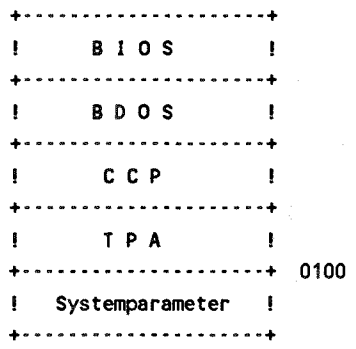
BDOS steht für Basic Disk Output System. Dieser Teil regelt den Datenaustausch zwischen Rechner und Diskettenlaufwerken.

BIOS

Basic Input/Output System soll BIOS abkürzen. BIOS und BDOS arbeiten eng zusammen, warum man die beiden Partner auch häufig unter einem zusammenfassenden Namen findet: FDOS.

TPA

Man nennt den Anwenderprogrammbereich im Speicher des Rechners TPA. Die auszuführenden Programme und die Daten, die mit diesen Programmen bearbeitet werden sollen, werden in diesem Speicher verwaltet.



So sieht im groben die Speichereinteilung von CP/M aus. Der Benutzerbereich (TPA) beginnt bei Adresse $0100.

Da die Erklärung, wie man unter CP/M programmiert, den in diesem Buch vorgesehenen Rahmen sprengen würde, wollen wir nur kurz auf die Programmierung unter CP/M unter Zuhilfenahme der BDOS- und BIOS-Routinen eingehen.

Das BDOS beinhaltet eine Sammlung von Unterprogrammen, die das grundlegende Arbeiten mit Dateien auf Diskette und einigen Peripheriegeräten ermöglicht. Da CP/M schon "einige Jahre auf dem Buckel" hat, existieren beispielsweise auch noch Routinen zur Ein- und Ausgabe von und auf Lochkarte. Sicherlich wird davon auch heute noch kräftig Gebrauch gemacht, auf unserem Commodore 128 können wir solche Routinen aber getrost vernachlässigen.

Um eine BDOS-Routine aufzurufen, muß man folgende standardisierten Vereinbarungen befolgen:

Im Register C des Prozessors wird die Kodenummer der gewünschten Unterroutine an BDOS übergeben. Im Register E bzw. bei 16-Bit-Werten im Registerpaar DE werden der Unterroutine, wenn nötig, Daten übermittelt. Wenn man die Register entsprechend versorgt hat, springt man BDOS an Adresse 5 durch einen Unterprogrammbefehl (CALL) an. BDOS ermittelt dann anhand des C-Registers die Adresse, an der die gewünschte Routine steht. Sie sehen, daß Programme auf diese Art und Weise auf wirklich allen Rechnern lauffähig gemacht werden können, da man fixe Regeln beachtet. Alle rechnerabhängigen Dinge wie Ein- und Ausgabe von Tastatur und Bildschirm werden von für den Rechner speziell entwickelten BDOS-Routinen erledigt. So muß man bei der Entwicklung eines neuen Rechners "nur" diese Routinen schreiben, und schon laufen alle CP/M-Programme auf diesem Rechner.

Zwei Voraussetzungen müssen natürlich gegeben sein. Der Rechner muß einen 8080-Prozessor oder einen kompatiblen besitzen, und es muß möglich sein, diese vorhandenen Programme auch einzulesen. Sollte zweiteres nicht möglich sein, so könnten immerhin theoretisch die CP/M-Programme auf diesem Rechner fahren. Allerdings ist es erfahrungsgemäß das kleinere Problem, vorhandene Programme zu transferieren. Dies kann a) durch Programmierung des Controllers geschehen (s.o.), oder b) indem man durch ein relativ einfaches Programm die vorhandenen Programme über eine gemeinsame Schnittstelle zweier Rechner, beispielsweise über den RS-232, übermittelt. Aber selbst wenn keine gemeinsame Schnittstelle existiert, so finden Tüftler immer einen Weg; beispielsweise kann der Joystick-Port als Eingangsport zur Übermittlung eines Programmes durchaus sehr dienlich sein.

Doch zurück zu BDOS. Sollte BDOS Werte aus der BDOS-Routine an das Hauptprogramm zurückliefern, so werden sich 8-Bit-Ergebnisse im Akkumulator des Prozessors auffinden, 16-Bit-Werte stehen als Rückgabe im HL-Registerpaar. Bei 16-Bit-Rückgaben findet sich zusätzlich das niederwertige Bit ebenfalls im Akku wieder, das höherwertige Bit befindet sich im B-Register.

Wollen Sie beispielsweise ein Zeichen auf die Konsole ausgeben, so benutzt man dazu die BDOS-Routine Nr. 2. Wir geben nun (in Z-80-Mnemonik) ein Fragezeichen aus:

```
LD   C,2        ;Konsolenausgabe
LD   E,"?"      ;Fragezeichen laden
CALL 5          ;BDOS anspringen
```

Wenn Sie wirklich ernsthaft in Maschinensprache unter CP/M programmieren wollen, so existiert speziell für Programmierer unter CP/M Fachliteratur. Wir geben Ihnen hier eine Liste der BDOS-Routinen mit Ein- und Ausgabeparameter zum Experimentieren. Zum seriösen Programmieren benötigt man sicherlich einiges mehr an Wissen über die einzelnen Routinen.

Routinenname	C	übernimmt	liefert
Warmstart auslösen	0		
Konsoleneingabe	1		A:Eingabe
Konsolenausgabe	2	E:Zeichen	
Lochstreifen lesen	3		A:Zeichen
Lochstreifen stanzen	4	E:Zeichen	
Zeichen an Drucker	5	E:Zeichen	
Direkte Konsolen Ein-	6	E: 255	A:0
und Ausgabe		E:Zeichen	A:Zeichen

IOBYTE abfragen	7		A:IOBYTE
IOBYTE setzen	8	E:IOBYTE	
Eine Zeichenkette ausg.	9	DE:=>Zeichenkette	
Eingabe aus Buffer	10	DE:=>Buffer	A:Zeichen
Konsolenstatus	11		A:=0 (keine Taste)
CP/M Version	12		HL:Version
Diskettensystemreset	13		
Bezugslaufwerk festl.	14	E:LW-Nummer	
Datei eröffnen	15	DE:Infos	A:255 (Error)
Datei schließen	16	DE:Infos	A:255 (Error)
Ersten Eintrag suchen	17	DE:Infos	A:255 (Error)
		sonst:	A:Zeichen
Folgeeintrag	18		A:255/Zeichen
Datei löschen	19	DE:Infos	A:255 (Error)
Aufzeichnung lesen	20	DE:Infos	A:0 (OK)
Aufzeichnung schreiben	21	DE:Infos	A:0 (OK)
Datei erzeugen	22	DE:Infos	A:255 (Error)
Datei umbenennen	23	DE:Infos	A:255 (Error)
aktive Laufwerke ermit.	24		HL:LW-Vektor
Bezugslaufwerk ermit.	25		A:Laufw.Nr.
Datenbuffer fixieren	26	DE:Buffer	
Belegungstabelle ermit.	27		HL:Adresse
Bezugslaufwerk schützen	28		
Geschütze Laufwerke	29		HL:LW-Vektor
Dateioptionen setzen	30	DE:Infos	A:255 (Error)
Diskparameter ermit.	31		HL:Adresse
Benutzernr. verwalten	32	E:255	A:Nummer
		E:Nummer	
Aufzeichnung lesen	33	DE:Infos	A:0 (OK)
Aufzeichnung schreiben	34	DE:Infos	A:0 (OK)
Dateigröße ermitteln	35	DE:Infos	(im Buffer)
Beschreiber setzen	36	DE:Infos	(im Buffer)
Laufwerk rücksetzen	37	DE:LW-Vektor	A:0

Diese Tabelle soll nur anzeigen, wie in etwa eine Programmierung mit BDOS aussehen kann, und welche Möglichkeiten Sie mit BDOS haben. Wie bereits erwähnt, reicht sie selbstverständlich nicht zur Programmierung aus, da interne Kenntnisse der einzelnen Routinen nötig sind.

Beinhaltet BDOS Routinen wie "Bezugslaufwerk schützen", so sind die Aufgaben von BIOS noch spezieller. BIOS besteht ebenso wie BDOS aus einer Liste an Routinen - die aber ausschließlich der Ein- und Ausgabe von Daten dienen. Teilweise findet man im BDOS Routinen, die dieselben Aufgaben haben wie Routinen im BIOS, so beispielsweise Konsolenstatus ermitteln.

BIOS trägt in ganz entscheidendem Maße zum Erfolg von CP/M bei, denn es ermöglicht den CP/M-Programmen, die ausgesprochen rechnerabhängigen Ein- und Ausgaben von und auf Konsole mittels spezieller Routinen durchführen zu lassen.

BIOS ist in vier grobe Bereiche untergliedert:

- *Schnittstelle zum BDOS und den CP/M-Programmen (beispielsweise bei den SUBMIT-Dateien),*
- *Schnittstelle zu den externen Speichermedien (Floppys),*
- *Ein- und Ausgabe über die durch BDOS angesprochene Peripherie,*
- *Pufferung von Daten, die dann entsprechend rechtzeitig zur Verfügung gestellt werden (SUBMIT-Dateien).*

Wie Sie wissen, sind BDOS und BIOS im Speicher verschiebbar. Die Routinen im BDOS werden durch Übergabe der Routinennummer im C-Register angesprochen - aufgerufen wird dann die fixe Speicheradresse &0005. Beim Aufruf von BIOS-Routinen

sieht das ein wenig anders aus: Es gibt eine Sprungtabelle, deren Reihenfolge zwar vorgeschrieben ist, allerdings kann die Startadresse dieser Sprungtabelle verschoben werden. Doch zunächst wollen wir uns diese Sprungtabelle einmal betrachten:

```
00+Offset: JMP BOOT      ;Kaltstart
03+Offset: JMP WBOOT     ;Warmstart
06+Offset: JMP CONST     ;Frage Konsolenstatus ab
09+Offset: JMP CONIN     ;Konsoleneingabe
0C+Offset: JMP CONOUT    ;Konsolenausgabe
0F+Offset: JMP LIST      ;Ausgabe auf Drucker
12+Offset: JMP PUNCH     ;Lochstreifenstanzer
15+Offset: JMP READER    ;Lochstreifenleser
18+Offset: JMP HOME      ;Kopf auf Spur 0
1B+Offset: JMP SELDSK    ;Laufwerk auswählen
1E+Offset: JMP SETTRK    ;Spur setzen
21+Offset: JMP SETSEC    ;Aufzeichnungsabschnitt auswählen
24+Offset: JMP SETDMA    ;Auswählen des Datenpuffers
27+Offset: JMP READ      ;Aufzeichnungsabschnitt lesen
2A+Offset: JMP WRITE     ;Aufzeichnungsabschnitt schreiben
2D+Offset: JMP LISTS     ;Frage Druckerstatus ab
30+Offset: JMP SECTRAN   ;Aufzeichnungsnummer übersetzen
```

Der Offset steht hier für den Versatz im Speicher, auf den ich noch genauer eingehen will. Wird eine der angegebenen Funktionen nicht benötigt - dies dürfte beim CPC bei den zwei Routinen der Fall sein, die die Lochstreifen betreffen -, so werden die entsprechenden Speicheradressen einfach mit

RET ! NOP ! NOP

aufgefüllt, damit sich der Speicher hinter dieser Routine nicht verschiebt. Ein Anspringen dieser Routine würde also sofort mit einem RETURN quittiert - es passiert also nichts. Um die Startadresse (den Offset) des BIOS zu ermitteln, muß man wissen, daß an Adresse 0 der Grundseite des Systems ein Sprung in den Warmstartvektor (den zweiten Vektor in der Tabelle) des BIOS enthalten ist. Dadurch wird es leicht, die Startadresse zu ermitteln. Wir wollen dies einmal tun. Starten Sie zunächst CP/M, dann laden Sie den Disassembler:

SID

Um uns den Speicherbereich ab Adresse 0 disassemblieren zu lassen, geben wir folgendes in den Rechner ein:

L0000

Und wir erhalten auch prompt die Antwort von SID:

```
0000  JMP  F403
0003  RST  07
0004  NOP
0005  JMP  D300
0008  NOP
```

etc. Was uns interessiert, ist also die erste Zeile: Es wird nach Adresse $F403 gesprungen, die Sprungtabelle von BIOS beginnt also an Adresse $F403. Das ist schon alles, Wissenswerte für unsere Programme. Wollen wir einen Vektor anspringen, so errechnen wir die Einsprungadresse durch:

Adresse:=Sprungnummer * 3 + $F403

Genau wie bei den BDOS-Routinen können Sie auch bei den BIOS-Routinen Parameter übergeben oder entgegennehmen. Sollen Werte an die BIOS-Routinen geliefert werden, so werden 8-Bit-Werte im C-Register übergeben, 16-Bit-Werte werden im Registerpaar BC erwartet. Sie können hier schon einen Unterschied zu den BDOS-Routinen erkennen, die die Parameter im E-Register bzw. im Registerpaar DE erwarten. Parameter, die die BIOS-Routinen an das Hauptporgramm zurückliefern, werden parallel zu den BDOS-Routinen bei 8-Bit-Werten im Akku, bei 16-Bit-Werten im Registerpaar HL übergeben. Tüftler haben sicherlich schon eine sehr schöne Möglichkeit entdeckt: Da die einzelnen BIOS-Routinen über Vektoren im RAM angesprungen werden, kann man diese natürlich verändern, d.h. auf eine eigene Routine umleiten, die gegebenenfalls dann ins Originalprogramm verzweigt.

Wir wollen eben noch auf die weiteren Eigenschaften der Grundseite (Adresse $0000 bis $0100) des Systems eingehen. Durch das Kommando 10 haben wir uns den Bereich $0000 bis $0016 disassemblieren lassen. Dabei fallen noch einige weitere JMP-Befehle auf: Besonders wichtig ist der Sprungbefehl an Adresse 5 - es müßte jetzt eigentlich bei Ihnen klingeln, denn an Adresse 5 haben wir doch unseren Unterprogrammaufruf gerichtet, wenn wir eine BDOS-Routine aufrufen wollten! Also muß BDOS bei unserem Commodore 128 an Adresse $D300 liegen, da der Sprungbefehl an Adresse $0005 uns dorthin verweist. Dann wollen wir uns diesen Speicherbereich doch einmal etwas näher ansehen:

ID3000

Damit es nicht so einfach ist, dürfen wir noch ein weiteres Mal springen, und zwar an Adresse $D9A4. Lassen Sie uns nicht verzagen, und diesen Sprung auch noch nachvollziehen:

LD9A4

Um es kurz zu machen: Es wird noch nach Adresse $ECF2 gesprungen, dann an Adresse $00E5. Erst dann wird allmählich mit der Dekodierung des C-Registers begonnen, um die anzuspringende Routine zu ermitteln. Es würde sicherlich zu weit führen, diesen Ablauf weiter zu verfolgen, Sie können dies mit Hilfe von SID und einer Nachtschicht in Ihrer Stube sehr leicht selber tun.

Doch das soll unsere "kleine" Einführung in die wichtigsten Programme und Utilities des "Additional Utility"-Paketes beenden. Wir hoffen, wir konnten Ihnen anhand der Beispiele verdeutlichen, welche Aufgaben die einzelnen Programme haben und ob diese für Sie sinnvoll sind, denn das muß jeder Benutzer für sich selbst entscheiden.

IX. Das ROM-Listing

Im 128 Intern nur kurz angeschnitten wurde der Teil des ROMs, der Z-80-Mnemocodes beinhaltet. Alles in allem umfaßt dieser ROM-Teil 4 KBytes, der physikalisch im Adreßbereich $D000 bis $DFFF liegt; Kenner spitzen hier schon die Ohren, denn sie wissen, daß es sich hier um einen recht kritischen Speicher-Bereich handelt.

Ist die Z-80 eingeschaltet (ein Bit im **MODE CONFIGURATION REGISTER** ist hierfür verantwortlich), so adressiert die MMU, das ist der für die Adressierung zuständige Baustein, das ab $D000 liegende ROM, wenn die Z-80 den Bereich $0000 bis $0FFF adressiert; hier liegt normalerweise bei einem Z-80-Betriebssystem das ROM. Als Anwender oder Programmierer unter Z-80 merken Sie hiervon allerdings garnichts, da die MMU dies alles vollkommen automatisch regelt.

Wird der Rechner eingeschaltet oder ein RESET ausgeführt, so wird die Z-80 auf jeden Fall zunächst einmal kurz eingeschaltet. Es werden die notwendigen Vorbereitungen für einen eventuellen CP/M-Start getroffen; Sie wissen ja, daß der Commodore 128 CP/M automatisch bootet und startet, sollte sich eine CP/M-Diskette beim RESET im Laufwerk befinden. Ist das Z-80-Betriebssystem mit seinen Vorbereitungen fertig, so wird wieder die 8502 eingeschaltet. Diese macht dann genau an der Stelle weiter, an der sie ihre Arbeit beendet hat - dies ist normalerweise die Stelle, nach der die Z-80 eingeschaltet worden ist. Klingt alles zunächst ein wenig kompliziert, ist es eigentlich aber gar nicht.

Es ist sicherlich sehr sinnvoll, das Z-80-Betriebssystem dokumentiert abzudrucken. Wenn Sie unter der Z-80 programmieren wollen, so ist es sicherlich sinnvoll zu wissen, wie das Z-80-ROM aussieht; schließlich müssen Sie damit ja kooperieren. Ferner ist aus dem Programmierstil leicht

ersichtlich, wie man die Z-80 beim Commodore 128 programmieren muß, denn es gibt einige Eigenheiten, die auf das Adreßchaos zurückzuführen sind.

Will man beispielsweise den Bereich $D000 bis $DFFF adressieren, also einen der Bausteine wie VIC, VDC, SID oder einen der anderen, so können Sie dies nicht mit den "normalen" Adressierungskommandos tun. Hierzu sind die Port-Kommandos OUT und IN zu verwenden. Um beispielsweise die 8502 einzuschalten, ist folgender Befehlsablauf notwendig:

```
LD  BC,$D505   ;Adresse Mode Configuration Register
LD  A,$B1      ;Code, um 8502 einzuschalten
OUT (C),A      ;entspricht LD (BC),A
```

entsprechend müssen Sie zum Auslesen von Speicherstellen das IN-Kommando verwenden. Im Registerpaar BC muß immer die anzusprechende Adresse enthalten sein.

Wenn Sie das hier abgedruckte ROM-Listing sorgsam studieren, so werden Sie feststellen, daß auch einige 8502-Code-Sequenzen enthalten sind. Das Umschalten der Prozessoren beispiesweise muß es ja in beiden Mnemocodes geben, sonst klappt es nicht. Ferner muß sich das Umschalten unbedingt innerhalb des Common Area abspielen, sonst meldet sich der Rechner nach dem Einschalten der Z-80 mit an Sicherheit grenzender Wahrscheinlichkeit nicht mehr wieder.

Wozu aber dient nun das Z-80-ROM, werden Sie sich berechtigterweise fragen. Nun, ganz einfach. Es dient in erster Linie dem Laden und Starten der notwendigen CP/M-Dateien. Ferner befinden sich sämtliche Bildschirmroutinen in diesem ROM, sowohl für den 40- als auch für den 80-Zeichen-Bildschirm. Hierauf greift das BIOS und das BDOS

sicherlich zurück, da nicht alle Routinen tatsächlich innerhalb des ROMs genutzt werden. Alle Systemmeldungen, die Sie während des Bootens von CP/M auf dem Bildschirm erhalten, erfolgen aus diesem ROM. Verfallen Sie aber bitte nicht dem Irrtum, daß das gesamte CP/M oder auch nur ganze Teile von CP/M sich in diesem ROM befinden; es handelt sich hierbei lediglich um extrem systemspezifische Routinen.

Das Laden von Diskette allerdings erfolgt in guter alter 8502-Manier. Hierfür schaltet die Z-80 kontrolliert kurz die 8502 ein, welche nach getaner Arbeit wieder brav zur Z-80 umschaltet. Dies ist eigentlich recht sinnvoll, da man auf diese Weise auf die bestehenden Kernal-Routinen zurückgreifen kann.

Wir wollen Sie nun aber - bevor Sie sich des Z-80-ROMs näher annehmen - kurz aufklären, denn sonst wachsen Ihnen auch noch graue Haare, und das muß ja nicht sein. Sie müssen wissen, daß beim Einschalten der Z-80 nichts mehr so bleibt, wie es ist. Der Bereich $D000 bis $DFFF wird nach $0000 heruntergespiegelt; ein Adressieren des Bereiches ab $D000 ist ausschließlich mittels OUT (C),x und IN (C),x möglich. Aber auch der Bildschirm und der Zeichengenerator werden verschoben; dies ist ganz logisch, denn dort, wo der Videoram sich normalerweise befindet, wird ja ROM adressiert: $0400 bis $07FF können Sie dem ROM-Listing entnehmen, der VIC-Chip wäre recht verwirrt.

Das Videoram wird nach $2C00 verschoben; der Zeichengenerator wird um 256 Bytes nach oben verschoben, beginnt also bei $D100. Einzig das Farbram bleibt dort, wo es ist, es läßt sich bekanntlich ja nicht verschieben.

Damit das Ganze aber nicht so einfach ist, gibt es neben dem 40-Zeichen- und dem 80-Zeichen-Bildschirm noch einen dritten, dem 40-Zeichen-Bildschirm übergeordneten Bildschirm.

Beim 40-Zeichen-Bildschirm wird von der Anzeige her ein 80-Zeichen-Bildschirm simuliert, der in der Horizontalen gescrollt werden kann. Um dies zu ermöglichen, muß man natürlich einen zusätzlichen Speicher haben. Wir haben diesen Bildschirm 80-Zeichen-Simulator getauft; es wird immer nur ein Fenster aus diesem Bildschirm auf dem 40-Zeichen-Bildschirm dargestellt.

Dieser 80-Zeichen-Bildschirm liegt im Bereich ab $1400; das dazugehörige Farbram finden Sie ab $1C00. Allerdings kann man vom Gebrauch dieses Pseudo-80-Zeichen-Bildschirmes nur abraten, den Überblick dürfte wohl jeder früher oder später verlieren.

So, das wäre eigentlich schon das Wichtigste zum ROM; nun aber noch eine kleine Auflistung der verwendeten Systemadressen im RAM, damit Sie nicht so lange rumrätseln müssen.

$2400	Zwischenspeicher allerlei
$2402	angepaßte Spalte für 80-Zeichen-Simulation
$2404+	Maske für Textausgabe
$2406+	Adresse Cursor 40-Zeichen-Schirm
$2408	Zeilen pro Bildschirm (24 Default)
$2409+	Cursoradresse + $1400 für 80-Zeichen-Simulation
$240B	Cursorspalte (40-Zeichen)
$240C	Cursorzeile (40-Zeichen)
$2400	Zeichenfarbe
$240E	Hintergrundfarbe
$240F	Rahmenfarbe
$2410	Füllzeichen (entweder $00 oder $80 für reverses Space)
$2411+	Cursoradresse
$2413	Cursorspalte
$2414	Cursorzeile
$2415	Attribut
$2416	Hintergrundfarbe 80-Zeichen-Schirm
$2417	Vordergrundfarbe 80-Zeichen-Schirm

```
$FD01     Flag für Vektoren setzen Ja/Nein
$FD03     zu lesende Spur
$FD04     zu lesender Sektor
$FD05     noch zu ladende Anzahl Blöcke
$FD06     Fehlerflag ($00,$0D,$FF)
$F008     Logische Filenummer
$FD0D     Zeiger auf Anpassungstabelle
$FD10     Attack/Decay
$F011     Volume
$FD12     Frequenz (Hi)
$FD13     Sustain/Release
$FD14     Ausschalten Ton
$FD15     Einschalten Ton
$FD18+    Basisadresse zu ladender Block
```

Mit "+" gekennzeichnete Adressen sind als 16-Bit-Werte zu verstehen.

Der Bootsektor der CP/M-Diskette soll auch noch kurz angeschnitten werden, damit Sie ihn sich nicht selbst anzusehen brauchen. Natürlich beginnt er mit der Identifizierung als Bootsektor, beinhaltet also zunächst die drei Buchstaben "CBM" und dann fünf Nullen. Das dann zu startende Programm sieht so aus:

```
SEI           Interrupt unterbinden
JSR $FF84     IOINIT
LDA #$3E      Konfigurationsbyte
STA $FF00     definieren
LDA #$C3      Code für JP unter Z-80
STA $FFEE     Code speichern
LDA #$08      Lo-Byte ist $08
STA $FFEF     speichern
LDA #$00      Hi-Byte ist $00
STA $FFF0     speichern; entspricht einem RST $08
JMP $FFD0     Z-80 einschalten
```

Das ist alles, was der Booting-Sektor macht. Alles andere spielt sich innerhalb des Z-80-ROMs ab und ist Ihnen ab sofort nicht mehr verborgen. Es ist Wert darauf gelegt worden, daß nahezu jede Zeile sinnvoll dokumentiert wurde.

Speicherbereiche, die kopiert werden, sind mit zwei Adressen versehen. Zunächst die physikalisch/logische Adresse und dann die vorgesehene Zieladresse. Relative Sprünge beziehen sich auf ihre vorgesehene Speicherumgebung.

Nun aber das versprochene **ROM-Listing:**

```
****************************************** RST 00 (Kaltstart)

0000:  3E 3E        LD    A,$3E         Konfigurationsbyte (RAM,I/O)
0002:  32 00 FF     LD    ($FF00),A     ins Konfigurationsregister
0005:  C3 3B 00     JP    $0038         Rest des Kaltstarts

****************************************** RST 08; Z-80 wieder eingeschaltet

0008:  31 77 3C     LD    SP,$3C77      SP initialisieren
000B:  3E 3F        LD    A,$3F         Konfigurationsbyte
000D:  C3 8C 01     JP    $018C         Rest der RST-08-Routine

****************************************** RST 10

0010:  E1           POP   HL            Rücksprungadresse von Stack
0011:  6E           LD    L,(HL)        Folgebyte als Offset holen
0012:  C3 20 00     JP    $0020         RST-20-Routine anspringen

0015:  00           NOP                 Füllbytes
0016:  00           NOP
0017:  00           NOP

****************************************** RST 18

0018:  E1           POP   HL            Rücksprungadresse von Stack
0019:  6E           LD    L,(HL)        Lo-Byte der Rücksprungadresse
001A:  C3 28 00     JP    $0028         RST-28-Routine anspringen

001D:  00           NOP                 Füllbytes
001E:  00           NOP
001F:  00           NOP

****************************************** RST 20

0020:  3A 0F FD     LD    A,($FD0F)
```

```
0023:  A7           AND   A           Setze Flags
0024:  28 02        JR    Z,$0028     Bei Nullflag Sprung
0026:  2C           INC   L           sonst erhöhe den
0027:  2C           INC   L           Sprungzeiger um 2

****************************************** RST 28; Sprungadresse aus Tabelle holen

0028:  26 01        LD    H,$01       Hi-Byte der Tabelle ist 1
002A:  7E           LD    A,(HL)      Folgebyte als Offset holen
002B:  23           INC   HL          Zeiger nun auf Hi-Byte
002C:  66           LD    H,(HL)      Hi-Byte auch holen
002D:  6F           LD    L,A         Lo-Byte nach L
002E:  E9           JP    (HL)        Und indirekter Sprung

002F:  00           NOP               Füllbyte

****************************************** RST 30; Dummy; Erstellungsdatum

0030:  30 35 2F 31 32 2F 38 35        .ASC "05/12/85"
                                      =12. Juni 1985

****************************************** RST 38

0038:  C3 FD FD     JP    $FDFD       RST 38 bei $FDFD fortführen

****************************************** RST 0 Contn'd

003B:  01 2F D0     LD    BC,$D02F    Register 47 des VIC-Chip (Tast.)
003E:  11 FC FF     LD    DE,$FFFC    $FF in die Tastatur schreiben
0041:  ED 51        OUT   (C),D       Keine Erweiterungstasten
0043:  03           INC   BC          Register 48=Taktregister
0044:  ED 59        OUT   (C),E       auf $FC setzen -> 1 MHz-Modus
0046:  01 05 D5     LD    BC,$D505    Mode-Configuration-Register
0049:  3E B0        LD    A,$B0       /EXROM und /GAME testen
004B:  ED 79        OUT   (C),A       sowie 128er-Modus einschalten
004D:  ED 78        IN    A,(C)       Mode-Configuration-Register
```

```
004F:  2F          CPL             wieder auslesen und negieren
0050:  E6 30       AND   $30       /EXROM oder /GAME gesetzt?
0052:  28 05       JR    Z,$0059 Nein, dann keine Cartridge

****************************************** 64er-Modus einschalten und
                                          Kontrolle an Cartridge übergeben

0054:  3E F1       LD    A,$F1     8502 einschalten und 64er-Modus
0056:  ED 79       OUT   (C),A     auswählen
0058:  C7          RST   $00       und Kaltstart ausführen
0059:  01 0F DC    LD    BC,$DC0F  CRB-Register im CIA 1
005C:  3E 08       LD    A,$08     auswählen und dann
005E:  ED 79       OUT   (C),A     Timer B anhalten sowie
0060:  0D          DEC   C         auch den Timer A des
0061:  ED 79       OUT   (C),A     CIA 1 stoppen.
0063:  0E 03       LD    C,$03     DDRB-Datenrichtungsregister
0065:  AF          XOR   A         für Port B: Alle Bits auf Ein-
0066:  ED 79       OUT   (C),A     gabe legen
0068:  0D          DEC   C         Zeiger auf ODRA und hier
0069:  3D          DEC   A         alle Bits auf
006A:  ED 79       OUT   (C),A     Ausgabe legen.
006C:  0D          DEC   C         Durch zweimaliges dekremen-
006D:  0D          DEC   C         tieren zeigt BC auf Port A
006E:  3E 7F       LD    A,$7F     Port A mit $7F (Siehe auch
0070:  ED 79       OUT   (C),A     Tastaturmatrix) beschreiben
0072:  03          INC   BC        Zeiger auf Port B (Eingabe)
0073:  ED 78       IN    A,(C)     und auslesen
0075:  E6 20       AND   $20       Commodore-Taste ausmaskieren
0077:  01 05 D5    LD    BC,$D505  Zeiger für Mode-Config.-Reg.
007A:  28 D8       JR    Z,$0054 Taste war gedrückt-> 64er-Modus
007C:  21 B4 0F    LD    HL,$0FB4  Es werden nun die Register
007F:  01 0A D5    LD    BC,$D50A  der MMU mit den Werten ab
0082:  16 0B       LD    D,$0B     $0FAA belegt
0084:  7E          LD    A,(HL)    Es ist zu beachten, daß alle
0085:  ED 79       OUT   (C),A     11 Register der MMU
0087:  2B          DEC   HL        von hinten mit den Werten
```

```
0088:  00          DEC   C            ab $0FB4 abwärts beschrieben
0089:  15          DEC   D            werden!
008A:  20 F8       JR    NZ,$0084 Ende der Schleife
008C:  21 1A 0D    LD    HL,$0D1A     Den Bereich von $001A
008F:  11 00 11    LD    DE,$1100     nach $1100 kopieren
0092:  01 08 00    LD    BC,$0008     Acht Bytes sind zu
0095:  ED B0       LDIR               kopieren (8502-Code!)
0097:  21 E5 0E    LD    HL,$0EE5     Ebenfalls den Bereich
009A:  11 D0 FF    LD    DE,$FF00     von $0EE5 an nach Common
0090:  01 1F 00    LD    BC,$001F     Area ab $FFD0 kopieren
00A0:  ED B0       LDIR               31 Bytes sind zu kopieren
00A2:  21 00 11    LD    HL,$1100     $1100 als Sprungvektor
00A5:  22 FA FF    LD    ($FFFA),HL   Sprungvektor in
00A8:  22 FC FF    LD    ($FFFC),HL   alle vier Adressen
00AB:  22 FE FF    LD    ($FFFE),HL   kopieren, unter anderem
00AE:  22 DD FF    LD    ($FFDD),HL   auch an $FFDD (frisch kopiert!)
00B1:  C3 E0 FF    JP    $FFE0        und Sprung in Z-80-Teil

*****************************************

00B4:  CD 6D 03    CALL  $0360        Lade Blöcke
00B7:  3A 06 3C    LD    A,($3C06)    Hole Blockanzahl
00BA:  22 18 FD    LD    ($FD18),HL   Zieladresse des Ladeblockes
00BD:  11 ED 00    LD    DE,$00ED
00C0:  F5          PUSH  AF           Rette Blockanzahl
00C1:  CD D3 00    CALL  $00D3        Vergleiche (HL) mit (DE)
00C4:  CC FA 02    CALL  Z,$02FA      Wenn ok, dann Aufruf
00C7:  F1          POP   AF           hole Blockanzahl zurück
00C8:  2A 18 FD    LD    HL,($FD18)   Ladeadresse holen
00CB:  11 20 00    LD    DE,$0020     32 als Offset
00CE:  19          ADD   HL,DE        hinzuaddieren
00CF:  3D          DEC   A            erniedrige Blockzähler
00D0:  20 E8       JR    NZ,$00BA     noch nicht fertig, dann schleifen
00D2:  C9          RET                Ende der Unterroutine

***************************************** Vergleiche (HL) mit (DE)
```

```
00D3:  06 0C        LD    B,$0C       12 Bytes sind zu vergleichen
00D5:  EB           EX    DE,HL       Austausch der beiden Vergleichsregister
00D6:  1A           LD    A,(DE)      Hole ersten Wert in Akku
00D7:  E6 7F        AND   $7F         und lösche irrelevantes 8. Bit
00D9:  BE           CP    (HL)        vergleich mit (HL) durchführen
00DA:  C0           RET   NZ          und bei Unterschied abbrechen
00DB:  23           INC   HL          sonst beide Adressen
00DC:  13           INC   DE          eins hochzählen
00DD:  10 F7        DJNZ  $0006       und weiterschleifen
00DF:  3A 06 3C     LD    A,($3C06)   Hole Blockanzahl
00E2:  FE 40        CP    $40         64 Blocks zu laden?
00E4:  1A           LD    A,(DE)      Hole Blockanzahl aus Tabelle
00E5:  20 01        JR    NZ,$00E8    $3C06 ist kleiner als 64
00E7:  1F           RRA               sonst verdoppele Akku
00E8:  32 35 3C     LD    ($3C35),A   und merken
00EB:  AF           XOR   A           Akku und Flags löschen
00EC:  C9           RET               Ende der Routine

****************************************** Text

00ED:  00           NOP
00EE:  43 50 4D 2B 20 20 20 20        "CPM+    "
00F6:  53 59 53 00                    "SYS" <Ende>

****************************************** Vergleich HL mit DE

00FA:  7C           LD    A,H         Hi-Byte von HL
00FB:  BA           CP    D           vergleiche mit D
00FC:  C0           RET   NZ          und Ende, wenn ungleich
00FD:  7D           LD    A,L         sonst vergleiche auch noch
00FE:  8B           CP    E           die Lo-Bytes von HL und DE
00FF:  C9           RET               und Rückkehr mit Flags

****************************************** Sprung-Tabelle für RST 28
```

```
0100:  84 06        .Word $0684       64 Einsprungadressen
0102:  6E 09        .Word $096E
0104:  AB 06        .Word $06AB
0106:  BC 09        .Word $09BC
0108:  C2 06        .Word $06C2
010A:  DD 09        .Word $0900
010C:  D1 06        .Word $0601
010E:  F1 09        .Word $09F1
0110:  DD 06        .Word $06DD
0112:  31 0A        .Word $0A31
0114:  E8 06        .Word $06E8
0116:  3C 0A        .Word $0A3C
0118:  F1 06        .Word $06F1
011A:  45 0A        .Word $0A45
011C:  7A 07        .Word $077A
011E:  48 0A        .Word $0A48
0120:  80 07        .Word $0780
0122:  62 0A        .Word $0A62
0124:  91 07        .Word $0791
0126:  8E 0A        .Word $0A8E
0128:  CA 07        .Word $07CA
012A:  BA 0A        .Word $0ABA
012C:  DC 07        .Word $07DC
012E:  DF 0A        .Word $0ADF
0130:  1E 08        .Word $081E
0132:  2D 0B        .Word $0B2D
0134:  1B 07        .Word $0718
0136:  78 08        .Word $0B7B
0138:  10 07        .Word $0710
013A:  62 0B        .Word $0B62
013C:  1C 09        .Word $091C
013E:  95 09        .Word $0995
0140:  27 09        .Word $0927
0142:  A2 09        .Word $09A2
0144:  4E 07        .Word $074E
0146:  AE 0B        .Word $0BAE
```

```
0148:  EB 00        .Word $00EB
014A:  EB 00        .Word $00EB
014C:  E8 00        .Word $00EB
014E:  EB 00        .Word $00EB
0150:  E3 03        .Word $03E3
0152:  6B 04        .Word $0468
0154:  FA 0C        .Word $0CFA
0156:  EB 00        .Word $00EB
0158:  EB 00        .Word $00EB
015A:  EB 00        .Word $00EB
015C:  EB 00        .Word $00EB
015E:  EB 00        .Word $00EB
0160:  3C 0C        .Word $0C3C
0162:  4A 0C        .Word $0C4A
0164:  CF 0B        .Word $0BCF
0166:  0C 0C        .Word $0C0C
0168:  26 05        .Word $0526
016A:  32 05        .Word $0532
016C:  2C 05        .Word $052C
016E:  EB 00        .Word $00EB
0170:  7F 0C        .Word $0C7F
0172:  C2 0C        .Word $0CC2
0174:  C7 0C        .Word $0CC7
0176:  E4 0C        .Word $0CE4
0178:  EB 00        .Word $00EB
017A:  30 08        .Word $0830
017C:  AE 08        .Word $08AE
017E:  60 06        .Word $0660
0780:  C3 0A        .Word $0AC3

0182:  09           ADD   HL,BC       Addiere BC als Offset
0183:  C3 33 09     JP    $0933       Hole A:Attribut, B:Zeichen an (HL) in VDC

****************************************** HL als Updateadresse im VDC

0186:  C3 53 09     JP    $0953       HL als Updateadresse im VDC
```

```
****************************************** VDC-Status abwarten und Register <Akku> anwählen

0189:  C3 45 09      JP    $0945         VDC-Status abwarten und <Akku> anwählen

****************************************** RST 8 Contn'd

018C:  32 00 FF      LD    ($FF00),A     Akku ins Konfigurationsbyte
018F:  21 00 30      LD    HL,$3000      Es wird der Bereich
0192:  11 01 30      LD    DE,$3001      $3000 bis $FEFF
0195:  01 FF CE      LD    BC,$CEFF      mit dem Wert $00
0198:  75            LD    (HL),L        gefüllt
0199:  ED B0         LDIR
0198:  21 22 0D      LD    HL,$0022      Der Bereich ab $0022
019E:  11 00 30      LD    DE,$3000      wird nach $3000 kopiert
01A1:  01 C3 01      LD    BC,$01C3      Es handelt sich hierbei um
01A4:  ED B0         LDIR                8502-Code!
01A6:  21 E5 0E      LD    HL,$0EE5      Bereich ab
01A9:  11 D0 FF      LD    DE,$FFD0      $0EE5 in Common Area
01AC:  01 1F 00      LD    BC,$001F      ab $FFD0 kopieren
01AF:  ED B0         LDIR                31 Bytes
01B1:  3E C9         LD    A,$C9         Code für RETurn
0183:  32 EE FF      LD    ($FFEE),A     RST 8 durch RET ersetzen
01B6:  CD E0 FF      CALL  $FFE0         8502 einschalten und dann weitermachen
01B9:  21 B4 0F      LD    HL,$0F84      Die MMU-Register
018C:  01 0A 05      LD    BC,$D50A      werden mit der Tabelle
01BF:  16 0B         LD    D,$0B         ab $0FAA gefüllt.
01C1:  7E            LD    A,(HL)        s.o.
01C2:  ED 79         OUT   (C),A         s.o.
01C4:  2B            DEC   HL            s.o.
01C5:  0D            DEC   C             s.o.
01C6:  15            DEC   D             s.o.
01C7:  20 F8         JR    NZ,$01C1      Ende der $chleife
01C9:  21 00 10      LD    HL,$1000      Den Bereich
01CC:  11 01 10      LD    DE,$1001      $1000 bis $2FFF
01CF:  01 FF 1F      LD    BC,$1FFF      mit dem Wert $00
```

```
0102:  75           LD    (HL),L      füllen
0103:  ED B0        LDIR              s.o.
01D5:  3E 1A        LD    A,$1A       Register 26 (Farben) des VDC
01D7:  CD 45 09     CALL  $0945       anwählen und dann Vorder/
01DA:  3E 90        LD    A,$90       Hintergrundfarben
01DC:  ED 79        OUT   (C),A       im VDC auf $90 setzen (Hellroter Cursor)
01DE:  3E 83        LD    A,$83       Hellblau und Alternate-Zeichensatz
01E0:  32 15 24     LD    ($2415),A   als Attribut definieren (VDC)
01E3:  3E 0E        LD    A,$0E       Zeichenfarbe für VIC-Chip
01E5:  32 0D 24     LD    ($2400),A   definieren
01E8:  CD BD 05     CALL  $05BD       Präpariere VDC-Zeichensatz
01EB:  3E 19        LD    A,$19       Der Bildschirm wird mit 24
01ED:  32 08 24     LD    ($2408),A   Zeilen definiert (DEVICE)
01F0:  CD 26 05     CALL  $0526       Nachfolgenden Text ausgeben
01F3:  FF           .Byte $FF         Bildschirm löschen ($FF)
01F4:  81 0A        .Byte $81,$0A     Zeile 1, Spalte 10

01F5:  42 4F 4F 54 49 4E 47 20        BOOTING
01F0:  43 50 2F 4D 20 50 4C 55        CP/M PL
0205:  53 00                          US<Ende>

0208:  01 18 D0     LD    BC,$D018    Basisadresse von Video-RAM
020B:  3E B6        LD    A,$B6       B->10-13 Video-RAM, 6->11-13
020D:  ED 79        OUT   (C),A       CHARROM; Videoram ab $2C00
020F:  CD D2 02     CALL  $0202       Bootsektor checken und holen
0212:  C2 FF 04     JP    NZ,$04FF    evtl. Fehler ausgeben
0215:  21 B2 0F     LD    HL,$0FB2    Tabellenanfang
0218:  22 02 3C     LD    ($3C02),HL  in $3C02 merken
021B:  CD B4 00     CALL  $0084       Ersten Teil laden
021E:  CD B4 00     CALL  $0084       Zweiten Teil laden
0221:  2A 09 3C     LD    HL,($3C09)  Hole Wert
0224:  7C           LD    A,H         Setzen des Zeroflags
0225:  B5           OR    L           wenn 16-Bit-Wert null ist
0226:  CA FF 04     JP    Z,$04FF     und Sprung, falls ja
0229:  21 09 3C     LD    HL,$3C09    Neuer Tabellenanfang
022C:  22 02 3C     LD    ($3C02),HL  merken
```

```
022F: CD 60 03      CALL  $0360         Daten laden
0232: 21 00 34      LD    HL,$3400      12 Bytes
0235: 11 29 3C      LD    DE,$3C29      von $3400
0238: 01 0C 00      LD    BC,$000C      nach $3C29
023B: ED B0         LDIR                kopieren
023D: CD 26 05      CALL  $0526         Leerzeile ausgeben
0240: 8A            .Byte $8A,$00,$00   Zeile 10, Spalte 0 und <Ende>
0243: 21 80 34      LD    HL,$3480      Zeiger auf Ausgabetext
0246: CD 34 05      CALL  $0534         und Text ab (HL) ausgeben
0249: 21 00 35      LD    HL,$3500      Zeiger $3500
024C: 22 04 3C      LD    ($3C04),HL    merken
024F: CD 26 05      CALL  $0526         Nachfolgenden Text ausgeben
0252: 83            .Byte $83,$0C       Zeile 3, Spalte 12

0254: 44 41 54 41 20 54 41 42           DATA TAB
025C: 4C 45 53 00                       LE$<Ende>

0260: 2A 33 3C      LD    HL,($3C33)    Nacheinander CP/M-Segmente
0263: 22 09 FD      LD    ($FD09),HL    in den Arbeitsspeicher
0266: 21 32 3C      LD    HL,$3C32      einladen
0269: CD 31 03      CALL  $0331         Hole Blockzahl und Startadresse
026C: 22 0B FD      LD    ($FD0B),HL    Startadresse merken
026F: CD 44 03      CALL  $0344
0272: 11 80 00      LD    DE,$0080      Recordoffset (CP/M-Intern)
0275: 19            ADD   HL,DE         von 128 addieren
0276: 20 F7         JR    NZ,$026F      Wenn noch nicht fertig, dann nochmal
0278: CD 26 05      CALL  $0526         Nachfolgenden Text ausgeben
027B: 84 0C         .Byte $84,$0C       Zeile 4, Spalte 12

0270: 43 4F 4D 4D 4F 4E 20 43           COMMON C
0285: 4F 44 45 00                       ODE<Ende>

0289: 21 2A 3C      LD    HL,$3C2A      Basisadresse für Tabelle
028C: CD 24 03      CALL  $0324         und Common Code einladen
028F: CD 26 05      CALL  $0526         Nachfolgenden Text ausgeben
0292: 85 0C         .Byte $85,$0C       Zeile 5, Spalte 12
```

```
0294:  42 41 4E 4B 45 44 20 43          BANKED C
029C:  4F 44 45 00                      ODE<Ende>

02A0:  21 2C 3C      LD     HL,$3C2C    Basisadresse für Tabelle
02A3:  CD 24 03      CALL   $0324       und Banked Code einladen
02A6:  CD 26 05      CALL   $0526       Nachfolgenden Text ausgeben
02A9:  86 0C         .Byte  $86,$0C     Zeile 6, Spalte 12

02AB:  42 49 4F 53 38 35 30 32          BIOS8502
02B3:  20 43 4F 44 45 00                CODE<Ende>

02B9:  21 30 3C      LD     HL,$3C30    Basisadresse für Tabelle
02BC:  CD 24 03      CALL   $0324       und BIOS8502 Daten laden
02BF:  3A 30 3C      LD     A,($3C30)
02C2:  47            LD     B,A
02C3:  3A 2F 3C      LD     A,($3C2F)
02C6:  90            SUB    B           Differenz bilden
02C7:  32 DE FF      LD     ($FFDE),A   Differenz als Hi-Byte
02CA:  AF            XOR    A           Akku löschen (=0)
02CB:  32 DD FF      LD     ($FFDD),A   als Lo-Byte merken
02CE:  2A 2D 3C      LD     HL,($3C2D)  Sprungadresse für CP/M-Einsprung
0201:  E9            JP     (HL)        holen und anspringen

****************************************** Lies ab Spur 1/Sektor 0

0202:  21 00 FE      LD     HL,$FE00    Ladeadresse für ersten zu ladenden
0205:  22 18 FD      LD     ($FD18),HL  Block merken
0208:  AF            XOR    A           Akku löschen
02D9:  32 04 FD      LD     ($FD04),A   Sektor#=0
02DC:  3C            INC    A           Akku=1
02DD:  32 03 FD      LD     ($FD03),A   Spur definieren
02E0:  CD 4F 04      CALL   $044F       Lies Spur/Sektor
02E3:  CD 68 04      CALL   $0468       Wenn gelesen, teste auf Bootsektor
02E6:  C0            RET    NZ          Kein Bootsektor
02E7:  3C            INC    A           Letztes Zeichen des Blockes+1
```

```
02E8:  21 00 38     LD    HL,$3800     Startadresse
02EB:  3E 20        LD    A,$20        32 als Blockzähler
02ED:  20 03        JR    NZ,$02F2     Wenn Akku vor INC<>$FF, Sprung
02EF:  26 3C        LD    H,$3C        sonst Hi-Byte auf $3C ändern
02F1:  87           ADD   A,A          und Blockzahl verdoppeln (Recordanzahl)
02F2:  22 07 3C     LD    ($3C07),HL   Merke Ladeadresse
02F5:  32 06 3C     LD    ($3C06),A    Merke Blockzahl
02F8:  AF           XOR   A            Akku und Flags (Wichtig) löschen
02F9:  C9           RET                und Ende der Routine

******************************************

02FA:  11 09 3C     LD    DE,$3C09     Zieladresse für Kopie
02FD:  3A 35 3C     LD    A,($3C35)    Flag holen
0300:  B7           OR    A            Setze Flags
0301:  28 07        JR    Z,$030A      Flag ist nicht gesetzt
0303:  11 19 3C     LD    DE,$3C19     sonst Startadresse +16
0306:  30           DEC   A            und Flag (Zähler) erniedrigen
0307:  C2 80 04     JP    NZ,$0480     und Fehler ausgeben

030A:  2A 18 FD     LD    HL,($FD18)   Hole Zieladresse
030D:  01 10 00     LD    BC,$0010     16 als Inkrement
0310:  09           ADD   HL,BC        addieren
0311:  ED B0        LDIR               und 16 Bytes kopieren
0313:  3A 09 3C     LD    A,($3C09)    Zähler holen und
0316:  B7           OR    A            Flags setzen
0317:  C8           RET   Z            Bei null ist alle Arbeit getan
0318:  2A 18 3C     LD    HL,($3C18)   Sonst hole Adresse
031B:  AF           XOR   A            und vergleiche mit $0000
031C:  BD           CP    L            Akku ist null
031D:  28 02        JR    Z,$0321      Lo-Byte ist null
031F:  BC           CP    H            vergleiche mit Hi-Byte
0320:  C8           RET   Z            Hi-Byte ist null
0321:  C3 29 02     JP    $0229        Zeroflag wird als Parameter übergeben

****************************************** Lade Records von Floppy
```

```
0324:  CD 31 03     CALL  $0331        Hole Anzahl und Startadresse
0327:  11 80 FF     LD    DE,$FF80     128 als Zweierkomplement
032A:  19           ADD   HL,DE        addieren (abziehen!)
032S:  CD 44 03     CALL  $0344        Record laden
032E:  20 F7        JR    NZ,$0327     noch ein Record
0330:  C9           RET                sonst Ende der Routine

****************************************** Hole Anzahl und Startadresse

0331:  5E           LD    E,(HL)       Anzahl holen
0332:  16 00        LD    D,$00        und Hi-Byte löschen
0334:  7B           LD    A,E          Testet Anzahl
0335:  B7           OR    A            auf null
0336:  CA 1B 05     JP    Z,$051B      BAD wenn null
0339:  EB           EX    DE,HL        sonst Anzahl nach HL
033A:  29           ADD   HL,HL        und verdoppeln (Recordanzahl)
033B:  22 00 3C     LD    ($3C00),HL   in $3C00 merken
033E:  EB           EX    DE,HL        wieder nach DE
033F:  2B           DEC   HL           Tabellenzeiger erniedrigen
0340:  66           LD    H,(HL)       Hi-Byte holen
0341:  2E 00        LD    L,$00        und Lo-Byte löschen
0343:  C9           RET                Ende der Routine

****************************************** Record laden und Zähler erniedrigen

0344:  E5           PUSH  HL           Auf Stack retten
0345:  2A 07 3C     LD    HL,($3C07)   Hole erste Vergleichsadresse
0348:  ES           EX    DE,HL        und in DE merken
0349:  2A 04 3C     LD    HL,($3C04)   Zweite Vergleichsadresse holen
034C:  CD FA 00     CALL  $00FA        Vergleiche (HL) mit (DE)
034F:  CC 6D 03     CALL  Z,$036D      und Sprung, wenn gleich
0352:  EB           EX    DE,HL        sonst Register wieder austauschen
0353:  21 80 00     LD    HL,$0080     Record-Offset von 128
0356:  19           ADD   HL,DE        hinzuaddieren
0357:  22 04 3C     LD    ($3C04),HL   und wieder merken
```

```
035A:  E1          POP   HL          Adresse zurückholen
035B:  E5          PUSH  HL          und erneut auf Stack sichern
035C:  EB          EX    DE,HL       Ziel und Quelle vertauschen
035D:  01 80 00    LD    BC,$0080    und 128 Bytes kopieren
0360:  ED B0       LDIR              (1 Record)
0362:  2A 00 3C    LD    HL,($3C00)  Hole Recordzähler
0365:  2B          DEC   HL          um eins erniedrigen
0366:  22 00 3C    LD    ($3C00),HL  und wieder abspeichern
0369:  7D          LD    A,L         Teste, ob Recordanzahl
036A:  B4          OR    H           gleich null ist und setze Flags
036B:  E1          POP   HL          Adresse zurückholen
036C:  C9          RET               und Routine beenden

****************************************** Daten ab $3400 + Offset laden

0360:  21 00 34    LD    HL,$3400    Ladeadresse (Basis)
0370:  22 18 FD    LD    ($FD18),HL  für 8502-Code merken
0373:  E5          PUSH  HL          und auf Stack sichern
0374:  2A 02 3C    LD    HL,($3C02)  Zeiger auf Blockladetabelle
0377:  16 00       LD    D,$00       Hi-Byte löschen
0379:  5E          LD    E,(HL)      und Lo-Byte aus Tabelle holen
037A:  23          INC   HL          Zeiger auf Tabelle erhöhen
037B:  22 02 3C    LD    ($3C02),HL  und wieder merken
037E:  EB          EX    DE,HL       Anzahl Blocks nach HL
037F:  29          ADD   HL,HL       und verdoppeln -> Recordanzahl
0380:  29          ADD   HL,HL       nochmal verdoppeln
0381:  3A 06 3C    LD    A,($3C06)   Hole Blockanzahl
0384:  0F          RRCA              /2
0385:  0F          RRCA              /4
0386:  0F          RRCA              /8
0387:  FE 04       CP    $04         Ist Blockzahl 32 - 63??
0389:  28 01       JR    Z,$038C     Ja, dann Sprung
038B:  29          ADD   HL,HL       sonst verdopple HL erneut
038C:  22 16 FD    LD    ($FD16),HL  uns als Blocknummer (laut CP/M) merken
038F:  30          DEC   A           erniedrige Anzahl zu ladender Blocks
0390:  32 05 FD    LD    ($FD05),A   und ebenfalls merken
```

```
0393:  F5          PUSH  AF          Rette Anzahl auf Stack
0394:  3E 01       LD    A,$01       Anzahl der zu ladenden Datenblöcke
0396:  32 BD 31    LD    ($31B0),A   auf 1 setzen
0399:  CD E3 03    CALL  $03E3       Mache aus Block# -> Track/Sektor
039C:  2A 16 FD    LD    HL,($FD16)  Hole Blocknummer
039F:  23          INC   HL          um eins erhöhen und
03A0:  22 16 FD    LD    ($FD16),HL  wieder ablegen
03A3:  F1          POP   AF          Hole Zähler und Flags
03A4:  28 2A       JR    Z,$03D0     Wenn Ende, dann Sprung
03A6:  3A 08 FD    LD    A,($FD08)   Sonst hole Fehlerflag
03A9:  A7          AND   A           und setze Flags
03AA:  28 24       JR    Z,$03D0     keine Fehler
03AC:  2A 03 FD    LD    HL,($FD03)  Track/Sektor holen
03AF:  E5          PUSH  HL          und auf Stack sichern
03B0:  CD E3 03    CALL  $03E3       Block# in Track/Sector wandeln
03B3:  E1          POP   HL          und errechneten Track/Sektor zurückholen
03B4:  3A 03 FD    LD    A,($FD03)   Hole Spur#
03B7:  BD          CP    L           vergleiche mit errechneter Spur
03B8:  20 13       JR    NZ,$03CD    ist leider ungleich
03BA:  E5          PUSH  HL          Rette Track/Sektor
03BB:  2A 16 FD    LD    HL,($FD16)  Hole Block#
03BE:  23          INC   HL          und erhöhe Blocknummer
03BF:  22 16 FD    LD    ($FD16),HL  Neue Blocknummer merken
03C2:  21 BD 31    LD    HL,$31BD    Zeiger auf zu ladende Blockzahl
03C5:  34          INC   (HL)        und ebenfalls erhöhen
03C6:  21 05 FD    LD    HL,$FD05    noch zu ladende Blockzahl
03C9:  35          DEC   (HL)        ebenfalls um eins erhöhen (Fehlerkorrektur)
03CA:  20 E4       JR    NZ,$03B0    noch wenigstens ein Block
03CC:  E1          POP   HL          Hole Track/Sektor von Stack
03CD:  22 03 FD    LD    ($FD03),HL  Merke Track/Sektor

****************************************** Fehler aufgetreten

0300:  CD 4F 04    CALL  $044F       Lies Block/Blöcke von Diskette
03D3:  21 19 FD    LD    HL,$FD19    Hi-Byte Zieladresse
03D6:  3A BD 31    LD    A,($31BD)   Hole Anzahl zu ladender Blöcke
```

```
0309:  86           ADD   A,(HL)       zur Zieladresse hinzuaddieren
03DA:  77           LD    (HL),A       und merken
03DB:  3A 05 FD     LD    A,($FD05)    Hole Anzahl noch zu ladender Blöcke
03DE:  A7           AND   A            Flags setzen
03DF:  20 AE        JR    NZ,S038F     weitermachen, wenn noch nicht Ende
03E1:  E1           POP   HL           sonst hole Track/Sektor
03E2:  C9           RET                und Ende der Routine

****************************************** Aus Block# Track/Sektor machen

03E3:  3E 23        LD    A,$23        35 als Offset für 1571 (2. Seite)
03E5:  32 00 24     LD    ($2400),A    und Offset merken
03E8:  2A 16 FD     LD    HL,(SFD16)   Hole Block#
03EB:  11 A8 02     LD    DE,$02A8     ab Blocknummer $2A8 -> 2.Seite adressieren
03EE:  B7           OR    A            Carry für Subtraktion löschen
03EF:  ED 52        SBC   HL,DE        absolute Blocknummer auf 2. Seite
03F1:   30  05      JR    NC,S03F8     durch Subtraktion ermitteln.  Sprung bei 1. Seite
03F3:  AF           XOR   A            Offset löschen
03F4:  32 00 24     LD    ($2400),A    und speichern
03F7:  19           ADD   HL,DE        Korrektur zur Subtraktion
03F8:  23           INC   HL           Block# + 2, da 1./2. Block
03F9:  23           INC   HL           für Directory reserviert sind
03FA:  11 65 01     LD    DE,$0165     (357) Blocknummer ab Spur 19
03FD:  01 00 15     LD    BC,$1500     Spur 0- hat 21 Sektoren/Spur
```

```
0400:  B7          OR    A          Carry für Subtraktion löschen
0401:  ED 52       SBC   HL,DE      Kontrolle auf 21 Sektoren/Spur
0403:  38 1B       JR    C,S0420    Ja, Block liegt im 21-Sektoren-Bereich
0405:  23          INC   HL         +1 zur Fehlerkorrektur
0406:  11 85 00    LD    DE,$0085   nächsten 133 Blöcke haben
0409:  01 11 13    LD    BC,$1311   17 Sektoren pro Spur
040C:  ED 52       SBC   HL,DE      Testen, ob Block in diesem Bereich
040E:  38 10       JR    C,S0420    Jawohl, dann Ende der Testreihe
0410:  11 6C 00    LD    DE,S006C   Nächsten Spuren haben
0413:  01 18 12    LD    BC,$1218   18 Sektoren/Spur (ab Spur $18)
0416:  ED 52       SBC   HL,DE      Teste, ob Block in diesem Bereich
0418:  38 06       JR    C,$0420    Ja, dann Ende der Testreihe
041A:  11 00 00    LD    DE,$0000   Korrekturfaktor auf null
0410:  01 1E 11    LD    BC,S111E   ab Spur 30 -> 17 Sektoren
0420:  19          ADD   HL,DE      Korrektur der Subtraktion
0421:  16 00       LD    D,$00      Hi-Byte von DE löschen
0423:  58          LD    E,B        Sektor nach E
0424:  B7          OR    A          Carry löschen
0425:  0C          INC   C          Spur erhöhen
0426:  ED 52       SBC   HL,DE      Sektoren pro Spur subtrahieren
0428:  30 FB       JR    NC,$0425   und evtl. weiterschleifen
042A:  19          ADD   HL,DE      Fehlerkorrektur
0428:  3A 00 24    LD    A,($2400)  Hole Offset für Seite 0/1
042E:  81          ADD   A,C        addiere Offset zur Spur
042F:  32 03 FD    LD    ($F003),A  Merke errechnete Spur
0432:  E5          PUSH  HL         Rette Sektor# auf Stack
0433:  21 B5 0F    LD    HL,$0FB5   Tabelle für angepaßte Sektornummern
0436:  01 15 00    LD    BC,$0015   zur Zugriffsoptimierung
0439:  7B          LD    A,E        Hole Sektornummer
043A:  B9          CP    C          und vergleiche mit Maximalwert
043B:  28 0A       JR    Z,$0447    Ist 21, dann Ende
043D:  09          ADD   HL,SC      sonst addiere Offset für Tabellenzeiger
043E:  0B          DEC   BC         Nächster Bereich hat 2 Sektoren
043F:  0B          DEC   BC         weniger pro Spur
0440:  B9          CP    C          Ist Maximalwert nun erreicht?
```

```
0441:  28 04        JR    Z,$0447      Ja, dann Ende
0443:  09           ADD   HL,BC        sonst addiere Offset für nächsten Bereich
0444:  0B           DEC   BC           Nächster Bereich hat einen Sektor
0445:  18 F9        JR    $0440        weniger und weiterversuchen
0447:  C1           POP   BC           Hole Sektor# von Stack
0448:  09           ADD   HL,BC        und Basis hinzuaddieren
0449:  7E           LD    A,(HL)       Hole angepaßte Sektornummer
044A:  32 04 FD     LD    ($FD04),A    und merken
0440:  3C           INC   A            Flags löschen und
044E:  C9           RET                Ende der Blockberechnung

****************************************** Lesen von Floppy

044F:  3E 03        LD    A,$03        Anzahl der Leseversuch auf
0451:  32 36 3C     LD    ($3C36),A    Floppy setzen
0454:  3E 01        LD    A,$01        Flag für Vektor setzen aus
0456:  32 01 FD     LD    ($FD01),A    (werden nicht nochmal gesetzt)
0459:  CD 8C 05     CALL  $058C        Block (Track/Sektor) auf Bildschirm anzeigen
045C:  CD E0 FF     CALL  $FFE0        8502 einschalten
045F:  3E 3F        LD    A,$3F        Konfigurationsbyte RAM Bank 0
0461:  32 00 FF     LD    ($FF00),A    einschalten
0464:  3A 06 FD     LD    A,($FD06)    Hole Lesefehlerflag
0467:  B7           OR    A            und teste es auf Lesefehler
0468:  20 32        JR    NZ,$049C     Lesefehler aufgetreten
046A:  C9           RET                Sonst Ende der Routine

****************************************** Testet geladenen Block auf Bootsektor

046B:  21 00 FE     LD    HL,$FE00     Startadresse geladener Block
046E:  7E           LD    A,(HL)       Hole erstes Zeichen
046F:  FE 43        CP    $43          ist es "C"?
0471:  C0           RET   NZ           Nein, dann kein Bootsektor
0472:  2C           INC   L            nächstes Zeichen
0473:  7E           LD    A,(HL)       holen und
0474:  FE 42        CP    $42          auf "B" vergleichen
0476:  C0           RET   NZ           Nicht, dann Ende
```

```
0477:  2C           INC   L           drittes Zeichen wird über-
0478:  7E           LD    A,(HL)      prüft,
0479:  FE 4D        CP    $4D         auf "N"
047B:  C0           RET   NZ          Nein, kein Bootsektor
047C:  2E FF        LD    L,$FF       Zeiger auf letztes Zeichen
047E:  7E           LD    A,(HL)      Hole dieses Zeichen
047F:  C9           RET               und Ende der Routine

****************************************** Fehler aufgetreten

0480:  CD 26 05     CALL  $0526       Nachfolgenden Text ausgeben
0483:  93 05        .Byte $93,$05     Zeile 19, Spalte 5

0485:  33 32 4B 20 4D 41 58 20        32K MA:
0480:  43 50 4D 28 2E 53 59 53        CPM+.SYS
0495:  20 53 49 5A 45 00               SIZE<Ende>

049B:  CF           RST   $08         Bootvorgang wiederholen

049C:  3C           INC   A           Test auf Akku=&FF
049D:  28 FC        JR    Z,$049B     Bei Akku=&FF ebenfalls Neu-Booting
049F:  3A 36 3C     LD    A,($3C36)   Sonst holde den Lesezähler
04A2:  3D           DEC   A           und um eins erniedrigen
04A3:  32 36 3C     LD    ($3C36),A   wieder ablegen
04A6:  20 AC        JR    NZ,$0454    Nochmal versuchen
04A8:  CD 26 05     CALL  $0526       Nachfolgenden Text ausgeben
04AB:  93 05        .Byte $93,$05     Zeile 19, Spalte 5

04AD:  52 45 41 44 20 45 52           READ ER
04B4:  52 4F 52 00                    ROR<Ende>

04B8:  CD 26 05     CALL  $0526       Nachfolgenden Text ausgeben

04BB:  20 2D 20 48 49 54 20 52         - HIT R
04C5:  45 54 55 52 4E 20 54 4F        ETURN TO
04CD:  20 52 45 54 52 59               RETRY
```

```
0401:  94 OF          .Byte $94,$0F      Zeile $14,$0F
0403:  44 45 4C 20 54 4F 20 45           DEL TO E
040B:  4E 54 45 52 20 43 31 32           NTER C12
04E3:  38 20 4D 4F 44 45 00              8 MODE<Ende>

04EA:  01 00 DC       LD    BC,$DC00     Port A CIA1 (Tastaturdekodierung)
04E0:  3E FE          LD    A,$FE        DEL- und <CR>-Taste werden ausmaskiert
04EF:  ED 79          OUT   (C),A        und abgecheckt
04F1:  OC             INC   C            Zeiger auf Port B des CIA1
04F2:  ED 78          IN    A,(C)        Hole Ergebnis
04F4:  E6 02          AND   $02          Bit für <CR> testen
04F6:  28 A3          JR    Z,S049B      <CR> ist gedrückt -> Reboot
04F8:  ED 78          IN    A,(C)        Sonst hole Wert erneut
04FA:  E6 01          AND   $01          Teste DEL-Bit
04FC:  20 EC          JR    NZ,$04EA     Nicht gedrückt, dann weiterversuchen
04FE:  C7             RST   $00          Sonst in den C-128-Modus

04FF:  CD 26 05       CALL  $0526        Nachfolgenden Text ausgeben
0502:  93 05          .Byte $93,$05      Zeile 19, Spalte 5

0504:  4E 4F 00                          NO<Ende>

0507:  CD 26 05       CALL  $0526        Nachfolgenden Text ausgeben
050A:  20 43          .Byte $20,$43      $20=Ab Cursorposition

050B:  43 50 4D 2B 2E 53 59 53           CPM+.SYS
0513:  20 46 49 4C 45 00                  FILE<Ende>

0519:  18 90          JR    S04B8        Neuer Versuch oder 128er-Modus
051B:  CD 26 05       CALL  $0526        Nachfolgenden Text ausgeben

051E:  93             .Byte $93,$05      Zeile $13, Spalte $05

0520:  42 41 44 00                       BAD<Ende>
```

```
0524:  18 E1        JR    $0507       Neuer Versuch oder 128er-Modus

****************************************** Nachfolgenden Text ausgeben

0526:  E3           EX    (SP),HL     Rücksprungadresse von Stack
0527:  CD 34 05     CALL  $0534       Text ab HL bis $00 ausgeben
052A:  E3           EX    (SP),HL     Ende des Textes als neue Rück-
052B:  C9           RET               sprungadresse und RETurn

****************************************** Text ab DE ausgeben

052C:  21 FF FF     LD    HL,$FFFF    zeichenmaske für auszugebende
052F:  22 04 24     LD    ($2404),HL  Zeichen setzen
0532:  D5           PUSH  DE          Textadresse auf Stack
0533:  E1           POP   HL          und in HL einlesen

****************************************** Text ab HL ausgeben

0534:  56           LD    D,(HL)      Hole Zeichen
0535:  23           INC   HL          Erhöhe Zeiger auf Textstelle
0536:  3A 05 24     LD    A,($2405)   Maske holen
0539:  A7           AND   A           Flags setzen
053A:  28 05        JR    Z,$0541     Maske erlaubt alles, Ende
053C:  AA           XOR   D           Sonst mit Maske verknüpfen
053D:  32 05 24     LD    ($2405),A   und zurückschreiben
0540:  57           LD    D,A         Zeichen nach D

0541:  7A           LD    A,D         aktuelles Zeichen
0542:  B7           OR    A           Flags setzen
0543:  C8           RET   Z           Null ist Endekennzeichen
0544:  FE 24        CP    $24         Dollarzeichen?
0546:  C8           RET   Z           Wenn ja, dann Ende
0547:  E5           PUSH  HL          Rette den aktuellen Zeiger
0548:  21 33 05     LD    HL,$0533    Sprung von Adresse $0533
054B:  E5           PUSH  HL          simulieren
054C:  FE 0A        CP    $0A         Wenn Linefeed, dann
```

```
054E:  C8          RET   Z           Sprung an diese Adresse
054F:  FE 0D       CP    $0D         Ist Zeichen <CR>?
0551:  20 0B       JR    NZ,$055E    Nein, dann nach S055E
0553:  CD 45 0A    CALL  $0A45       Spalte=0 - 40 Zeichen
0556:  CD F1 06    CALL  S06F1       Spalte=0 - 80 Zeichen
0559:  CD F1 09    CALL  S09F1       Zeilenzeiger erhöhen
055C:  DF          RST   $18         Sprung nach $0601 (Zeilenzeiger erhöhen)
0550:  0C          .Byte $0C         Sprungvektor
055E:  FE FF       CP    $FF         Ist Zeichen $FF?
0560:  20 17       JR    NZ,$0579    Nein, dann überspringe Löschteil
0562:  11 00 18    LD    DE,$1800    Zeiger auf Statuszeile (Zeile/Spalte)
0565:  CD 85 05    CALL  S0585       Setze Cursorpos
0568:  CD 48 0A    CALL  S0A48       Statuszeile löschen (40er)
0568:  CD 7A 07    CALL  $077A       Statuszeile löschen (VDC)
056E:  11 00 00    LD    DE,$0000    Zeiger auf erste Bildschirmposition
0571:  CD 85 05    CALL  $0585       Setze Cursorpos
0574:  CD 62 0A    CALL  $0A62       Cursorposition bis Bildende löschen
0577:  DF          RST   $18         Dasselbe für VDC
0578:  20          .Byte $20         Sprungvektor 32

0579:  E6 80       AND   A,$80       Teste Bit 7
057B:  28 39       JR    Z,$05B6     Ist gelöscht, dann Zeichen normal ausgeben
057D:  C1          POP   BC          Simulierte RS-Adresse zurück
057E:  E1          POP   HL          Zeiger zurück
057F:  5E          LD    E,(HL)      Hole Spalte
0580:  23          INC   HL          Zeiger auf nächstes Zeichen
0581:  E5          PUSH  HL          Rette Zeiger
0582:  C5          PUSH  BC          Rette simulierte CALL-Adresse
0583:  CB BA       RES   7,D         Bit 7 von Zeile löschen
0585:  D5          PUSH  DE          Rette Zeile/Spalte
0586:  CD BC 09    CALL  $09BC       40-Zeichen-Cursor setzen
0589:  D1          POP   DE          Hole Zeile/Spalte
058A:  DF          RST   $18         80-Zeichen-Cursor anspringen
0588:  04          .Byte $04         Sprungvektor ist 4

****************************************** zu lesenden Block (Track/Sektor) anzeigen
```

```
058C:  11 4A 18     LD    DE,$184A     Zeile 24, Spalte 74 setzen als Cursor
058F:  CD AB 06     CALL  S06AB        80-Zeichen-Cursorpos. setzen
0592:  11 22 18     LD    DE,$1822     Zeile 24, Spalte 34
0595:  CD BC 09     CALL  S09BC        40-Zeichen-Cursor setzen
0598:  3A 03 FD     LD    A,($FD03)    zu lesende Spur
059B:  CD A6 05     CALL  $05A6        in ASCII wandeln
059E:  16 20        LD    D,$20        Leerzeichen
05A0:  CD B6 05     CALL  $05B6        ausgeben
05A3:  3A 04 FD     LD    A,($FD04)    zu ladender Sektor

****************************************** macht aus <Akku> ASCII

05A6:  06 2F        LD    B,$2F        ASCII "0" - 1
05A8:  04           INC   B            Zehnerstelle erhöhen
05A9:  D6 0A        SUB   $0A          Zieht (wenn möglich) 10 ab
05AB:  30 FB        JR    NC,S05A8     Zehnerstelle ist noch nicht null
05AD:  C6 3A        ADD   A,$3A        Fehlerkorrektur plus ASCII "0"
05AF:  F5           PUSH  AF           Rette Einerstelle
05B0:  78           LD    A,B          Zehnerstelle (ASCII) nach <Akku>
05B1:  CD B5 05     CALL  S05B5        und ausgeben
0584:  F1           POP   AF           Hole Einerstelle (ASCII)

****************************************** Zeichen <Akku) ausgeben

05B5:  57           LD    D,A          Zweichen nach <D>
05B6:  D5           PUSH  DE           und merken
05B7:  CD 6E 09     CALL  $096E        Zeichen ausgeben
05BA:  D1           POP   DE           Hole auszugebendes Zeichen (VDC)
05BB:  DF           RST   $18          Zeichen auf 40-Zeichen-Bildschirm
05BC:  00           .Byte $00          Vektor 0

****************************************** Präpariere Zeichensatz 80 Zeichen

05BD:  21 04 30     LD    HL,$3004     Adresse im VDC-RAM
05C0:  CD 30 09     CALL  $0930        Wert an $3004 holen
```

```
05C3:  04          INC   B          und auf 0 prüfen durch
05C4:  05          DEC   B          dekrementieren und inkrementieren
05C5:  C8          RET   Z          Ist null, dann wurde schon präpariert
05C6:  21 00 38    LD    HL,$3800   $3800 bis
05C9:  01 00 04    LD    BC,$0400   S3FFF mit dem
05CC:  16 00       LD    D,$00      Wert 0
05CE:  CD 47 08    CALL  $0847      füllen (löschen)
05D1:  21 A0 37    LD    HL,$37A0   ASCII 122
05D4:  11 A0 38    LD    DE,$38A0   wird
05D7:  01 08 00    LD    BC,$0008   ASCII 138
050A:  CD B0 08    CALL  S08B0
05DD:  21 90 36    LD    HL,$3690   ASCII 105
05E0:  11 90 38    LD    DE,$3890   wird
05E3:  01 08 00    LD    BC,$0008   ASCII 137
05E6:  CD B0 08    CALL  $0880
05E9:  21 E0 35    LD    HL,$35E0   ASCII 94 (PI)
05EC:  11 E0 38    LD    DE,$38E0   wird
05EF:  01 18 00    LD    BC,$0018   ASCII 95 (_)
05F2:  CD B0 08    CALL  $08B0
05F5:  21 10 30    LD    HL,$3010   A-Z (ASCII 1-26)
05F8:  11 10 36    LD    DE,$3610   nach
05FB:  01 98 01    LD    BC,$0198   ASCII 97 ff.
05FE:  CD B0 08    CALL  S08B0
0601:  21 00 30    LD    HL,$3000   $3000 bis $31FF
0604:  01 00 02    LD    BC,$0200   im VDC-RAM
0607:  16 00       LD    D,S00      löschen
0609:  CD 47 08    CALL  $0847
060C:  21 00 20    LD    HL,$2000   "@" (ASCII 64)
060F:  11 00 34    LD    DE,$3400   in VDC-RAM
0612:  01 08 00    LD    BC,$0008   kopieren
0615:  CD B0 08    CALL  S08B0
0618:  21 B0 21    LD    HL,S21B0   [ bis ] (ASCII 27 bis 29)
061B:  11 B0 35    LD    DE,$3580   nach
061E:  01 28 00    LD    BC,$0028   ASCII 91
0621:  CD B0 08    CALL  $08B0
0624:  21 C0 21    LD    HL,$21C0   ASCII 28 (Pfundzeichen)
```

```
0627:  11 00 38     LD    DE,$3800     nach
062A:  01 08 00     LD    BC,$0008     ASCII 128
0620:  C0 B0 08     CALL  $08B0
0630:  21 E0 21     LD    HL,$21E0     ASCII 30 (^) wird 129
0633:  11 10 38     LD    DE,$3810     sowie
0636:  01 18 00     LD    BC,$0018     ASCII 31 (_) wird 130
0639:  CD S0 08     CALL  $08B0
063C:  21 00 24     LD    HL,$2400     Großbuchstaben und Sonderzeichen
063F:  11 00 3C     LD    DE,$3C00     nach $3C00 (ASCII 192)
0642:  01 F8 03     LD    BC,$03F8
0645:  CD B0 08     CALL  $0880
0648:  11 1A 0F     LD    DE,$0F1A     ASCII 227
064B:  21 C0 35     LD    HL,$35C0     wird
064E:  CD 70 06     CALL  $0670        ASCII 92
0651:  21 E0 35     LD    HL,$35E0     ASCII 228-230
0654:  06 03        LD    B,$03        wird
0656:  CD 62 06     CALL  $0662        ASCII 94-96
0659:  21 B0 37     LD    HL,$3780     ASCII 231-235
065C:  06 05        LD    B,$05        wird
065E:  18 02        JR    $0662        ASCII 123 ff.
0660:  E1           POP   HL           Hole Rücksprungadresse nach HL
0661:  E3           EX    ($P),HL      und eine Rücksprungadresse löschen

****************************************** Kopiere (DE)->(HL) VDC (BC) mal

0662:  C5           PU$H  BC           Zähler retten
0663:  E5           PUSH  HL           Zieledresse merken
0664:  CD 70 06     CALL  $0670        (DE)->(HL) VDC-RAM, 8 Bytes
0667:  E1           POP   HL           Zieladresse zurückholen
0668:  01 10 00     LD    BC,$0010     Und 16 addieren (anstatt 8)
066B:  09           ADD   HL,BC        wegen internen VDC-Aufbau
066C:  C1           POP   BC           Hole Zähler zurück
0660:  10 F3        DJNZ  $0662        noch ein Zeichen zu kopieren?
066F:  C9           RET                Nein, dann Ende

****************************************** (DE)->(HL) VDC; 8 Bytes
```

```
0670:  CD 53 09      CALL  $0953        HL als Update anmelden
0673:  26 08         LD    H,$08        8 Bytes sollen kopiert werden
0675:  1A            LD    A,(DE)       Hole das Zeichen aus RAM
0676:  ED 79         OUT   (C),A        und speichere es nach VDC
0678:  0D            DEC   C            Zeiger auf Status-Flag
0679:  13            INC   DE           Zeiger auf Tabelle erhöhen
067A:  ED 78         IN    A,(C)        Hole $tatus-Flag
067C:  17            RLA                Teste Ready-Bit
067D:  30 FB         JR    NC,$067A     Noch nicht fertig
067F:  0C            INC   C            Zeiger wieder auf $D601
0680:  25            DEC   H            Erniedrige Zähler
0681:  20 F2         JR    NZ,$0675     und springe, wenn noch keine 8 Bytes
0683:  C9            RET                sonst beende Routine

****************************************** <A> Zeichen ausgeben inkl. Cursorbewegung

0684:  2A 11 24      LD    HL,($2411)   Cursoradresse holen
0687:  CD 07 09      CALL  $0907        Zeichen ausgeben auf 80-Zeichen
068A:  3A 13 24      LD    A,($2413)    Cursorspalte
068D:  FE 4F         CP    $4F          rechten Rand (79) erreicht?
068F:  28 3C         JR    Z,$06CD      Ja, dann nächste Zeile
0691:  3C            INC   A            sonst erhöhe den Spaltenzeiger
0692:  32 13 24      LD    ($2413),A    und merke die neue Spalte
0695:  2A 11 24      LD    HL,($2411)   Hole Cursoradresse
0698:  23            INC   HL           und ebenfalls um eine Stelle
0699:  22 11 24      LD    ($2411),HL   erhöhen und wieder abspeichern

****************************************** HL als Cursoradresse setzen

069C:  3E 0E         LD    A,$0E        Cursoradresse Hi-Byte
069E:  CD 45 09      CALL  $0945        anmelden bei VDC
06A1:  ED 61         OUT   (C),H        Higher Byte an VDC übergeben
06A3:  3E 0F         LD    A,$0F        Register 15 ist Cursoradresse Lo
06A5:  CD 45 09      CALL  $0945        anmelden bei VDC
06A8:  ED 69         OUT   (C),L        und auch Lo-Byte übergeben
```

```
06AA:  C9           RET              Ende der Übertragung

****************************************** 80-Zeichen-Cursorposition setzen
                                         D:Spalte, E:Zeile

06AB:  7A           LD    A,D        Hole Zeile
06AC:  FE 19        CP    $19        Größer als 24
06AE:  D0           RET   NC         Ja, dann ungültig und Ende
06AF:  7B           LD    A,E        Hole Spalte
06B0:  FE 50        CP    $50        größer als 79?
06B2:  D0           RET   NC         Ja, dann ungültig und Ende
06B3:  EB           EX    DE,HL      zwecks Speicherung HL nach DE
06B4:  22 13 24     LD    ($2413),HL und Cursorposition merken

06B7:  2A 13 24     LD    HL,($2413) Cursorposition holen
06SA:  CD CE 0C     CALL  $0CCE      Zeile*80 + Spalte
06BD:  22 11 24     LD    ($2411),HL Adresse merken
06C0:  18 DA        JR    S069C      Cursoradresse an VDC übergeben

****************************************** Zeile um eins erniedrigen

06C2:  3A 14 24     LD    A,($2414)  Hole Cursorzeile
06C5:  B7           OR    A          setze die CPU-Flags
06C6:  C8           RET   Z          Zeile ist bereits erste, dann Schluß
06C7:  3D           DEC   A          sonst erniedrige die Zeile
06C8:  32 14 24     LD    ($2414),A  und merke diese
06CB:  18 EA        JR    $0687      neue Cursoradresse berechnen

****************************************** Spalte=0 (erste Spalte) setzen inkl.
                                         Zeilenbeeinflussung

06CD:  AF           XOR   A          Akku wird null
06CE:  32 13 24     LD    ($2413),A  und als Spalte merken
0601:  3A 14 24     LD    A,($2414)  Hole aktuelle Cursorzeile
06D4:  FE 17        CP    $17        Ist Zeile die 23.?
06D6:  28 21        JR    Z,S06F9    Ist erreicht, dann Sprung
```

```
060B:  30 1A        JR    NC,$06F4     Zeile ist 24
060A:  3C           INC   A            sonst Zeile um eins erhöhen
06DB:  18 EB        JR    $06C8        und Zeile merken

****************************************** Spalte um eins erniedrigen

060D:  3A 13 24     LD    A,(S2413)    Hole aktuelle Spalte
06E0:  B7           OR    A            setze Flags zum Testen auf null
06E1:  C8           RET   Z            Wenn bereits erste Spalte, dann Ende
06E2:  30           DEC   A            sonst erniedrige den Spaltenzeiger
06E3:  32 13 24     LD    ($2413),A    und merke neue Spalte
06E6:  18 CF        JR    $0687        Neue Cursoradresse berechnen

****************************************** Spalte um eins erhöhen

06E8:  3A 13 24     LD    A,($2413)    Hole aktuelle Spalte
06EB:  3C           INC   A            und um eins erhöhen
06EC:  FE 50        CP    $50          ist Spalte 80 erreicht?
06EE:  20 F3        JR    NZ,$06E3     Nein, dann merke Spalte
06F0:  C9           RET                sonst beende Routine

****************************************** Spalte auf 0 (erste Spalte) setzen

06F1:  AF           XOR   A            Akku gleich Null setzen
06F2:  18 EF        JR    $06E3        und dann als neue Spalte merken

****************************************** Zeile=23 setzen

06F4:  3E 17        LD    A,$17        Zeile auf 23. setzen
06F6:  32 14 24     LD    ($2414),A    und im Speicher merken
06F9:  21 50 00     LD    HL,$0050     Bildschirm um eine
06FC:  11 00 00     LD    DE,$0000     Zeile nach oben scrollen
06FF:  01 30 07     LD    BC,$0730     durch Kopieren der 2 in die
0702:  CD B0 08     CALL  $08B0        erste Zeile etc.
0705:  21 30 07     LD    HL,$0730     Zeiger auf letzte Zeile (nicht Statuszeile)
0708:  01 50 00     LD    BC,$0050     Anzahl ist 80 Zeichen
```

```
0708:  CD 41 08     CALL  $0841        Zeile löschen
070E:  18 A7        JR    $0687        und neue Cursorposition anmelden

****************************************** Neues Attribut definieren (B:zu löschende,
                                            C:zu setzende Eigenschaften)

0710:  3A 15 24     LD    A,($2415)    Hole Attribut
0713:  2F           CPL                komplementiere Attribut
0714:  B0           OR    B            zu löschende Bits (Eigenschaften)
0715:  2F           CPL                zurückkomplementieren
0716:  B1           OR    C            zu setzende Bits (Eigenschaften)
0717:  32 15 24     LD    ($2415),A    Abspeichern des neuen Attributes
071A:  C9           RET                Ende der Routine

******************************************

071B:  78           LD    A,B          Zeichen holen
071C:  D6 20        SUB   $20          ASCII 32 subtrahieren (Leer)
071E:  FE 20        CP    $20          ASCII 32?
0720:  38 0A        JR    C,$072C      Kleiner, dann Sprung
0722:  0E 20        LD    C,$20        Merker für ASCII 32 abgezogen
0724:  CD E5 0C     CALL  $0CE5        ASCII + Code umwandeln
0727:  D8           RET   C            Geschafft
0728:  7E           LD    A,(HL)       Hole Zeichen aus Tabelle
0729:  E6 0F        AND   $0F          Bits 4-7 ausmaskieren
072B:  80           ADD   A,B          und <B> als Offset dazu
072C:  32 00 24     LD    ($2400),A    merke Zeichen
072F:  0E 20        LD    C,$20        ASCII 32 subtrahiert-Zeiger
0731:  C6 30        ADD   A,$30        ASCII 48 ("0") addieren
0733:  21 0A 0F     LD    HL,$0F0A     Tabelle für Farvanpassung
0736:  CD E8 0C     CALL  $0CE8        umwandeln
0739:  7E           LD    A,(HL)       Hole Farbwert
073A:  80           ADD   A,B          und addiere Attribut
073B:  FE 10        CP    $10          Übertrag ins höherwertige Nibble?
073D:  38 1B        JR    C,$075A      Vordergrundfarbe wird definiert
073F:  E6 0F        AND   $0F          sonst Bits 4-7 ausmaskieren
```

```
0741:  32 16 24     LD    ($2416),A    Hintergrundfarbe merken
0744:  F5           PUSH  AF           Rette Farbcode
0745:  3E 1A        LD    A,$1A        Register 26=Vorder/Hintergrundfarbe
0747:  CD 45 09     CALL  $0945        VDC-Status abwarten und anmelden
074A:  F1           POP   AF           Hole Farbewert
074B:  ED 79        OUT   (C),A        und Farbe setzen
0740:  C9           RET                Ende der Routine

****************************************** B:Backgnd, D:Attribut A:Foregnd

074E:  3A 16 24     LD    A,($2416)    Hintergrundfarbe holen
0751:  47           LD    B,A          und nach <B>
0752:  3A 15 24     LD    A,($2415)    Attribut holen
0755:  57           LD    D,A          und nach D
0756:  3A 17 24     LD    A,($2417)    Vordergrundfarbe holen
0759:  C9           RET                Ende der Routine

****************************************** Neue Vordergrundfarbe definieren

075A:  47           LD    8,A          Farbe nach B
075B:  3A 15 24     LD    A,($2415)    Hole aktuelles Attribut
075E:  E6 F0        AND   $F0          Farbennibble ausmaskieren
0760:  B0           OR    B            und neue Farbe setze
0761:  32 15 24     LD    ($2415),A    neues Attribut merken
0764:  3A 00 24     LD    A,($2400)    Hole Hintergrundfarbe und
0767:  32 17 24     LD    ($2417),A    merken
076A:  2A 11 24     LD    HL,($2411)   Hole Cursoradresse
0760:  11 00 08     LD    DE,$0800     und Offset für Attribut-RAM
0770:  19           ADD   HL,DE        addieren
0771:  CD 53 09     CALL  $0953        HL als Update anmelden
0774:  3A 15 24     LD    A,($2415)    Attribut holen
0777:  ED 79        OUT   (C),A        und an Cursoradresse speichern
0779:  C9           RET                Ende der Routine

****************************************** Cursorposition bis Zeilenende löschen
```

```
077A:  CD C7 0C      CALL  $0CC7       Hole Cursorposition, Anzahl Zeichen
077D:  03            INC   BC          erhöhe die Anzahl
077E:  18 0E         JR    $078E       und lösche den Rest der Zeile

****************************************** Cursorposition bis Bildschirmende löschen

0780:  CD C7 0C      CALL  $0CC7       Hole Cursorposition, Anzahl Zeichen
0783:  EB            EX    DE,HL       Cursoradresse nach DE
0784:  21 80 07      LD    HL,$0780    Adresse Anfang der Statuszeile
0787:  AF            XOR   A           Lösche Carry für Subtraktion
0788:  ED 52         SBC   HL,DE       Errechne Anzahl Zeichen bis Statuszeile
078A:  F8            RET   M           Wenn negativ, dann Ende (Fehler)
078B:  44            LD    B,H         Sonst BC gleich
078C:  40            LD    C,L         Anzahl Zeichen bis Statuszeile
0780:  EB            EX    DE,HL       und aktuelle Cursorposition wieder nach HL
078E:  C3 41 08      JP    $0841       Lösche bis Anfang Statuszeile

****************************************** 1 Zeichen einfügen Restzeile verschieben

0791:  CD C7 0C      CALL  $0CC7       Hole Cursorposition und Anzahl Zeichen
0794:  21 4F 00      LD    HL,$004F    addiere 79 zu Zeilenanfang
0797:  19            ADD   HL,DE       zum Zeilenanfang
0798:  3D            DEC   A           Anzahl Zeichen bis Zeilenende-1
0799:  28 2C         JR    Z,$07C7     keines mehr, dann Schluß
079B:  54            LD    D,H         Adresse des Zeilenende
079C:  5D            LD    E,L         nach DE
079D:  2B            DEC   HL          Adresse Zeilenende-1
079E:  C5            PUSH  BC          Rette Anzahl
079F:  E5            PUSH  HL          Rette Quelle
07A0:  D5            PUSH  DE          Rette Ziel
07A1:  CD AD 07      CALL  $07AD       Kopiere (HL)->(DE) in VDC-RAM
07A4:  01 00 08      LD    BC,$0800    Addiere nun Offset für
07A7:  E1            POP   HL          Attribut-RAM im VDC
07A8:  09            ADD   HL,BC       zur Quelladresse
07A9:  EB            EX    DE,HL       und in DE merken
07AA:  E1            POP   HL          Hole Zieladresse von Stack
```

```
07AB:  09           ADD   HL,BC       ebenfalls den Offset addieren
07AC:  C1           POP   BC          Anzahl holen

07AD:  C5           PUSH  BC          und wieder sichern
07AE:  CD 53 09     CALL  $0953       HL als Update anmelden
07B1:  ED 78        IN    A,(C)       Hole aktuellen Inhalt
07B3:  EB           EX    DE,HL       Ziel- und Quelladresse vertauschen
07B4:  F5           PUSH  AF          Rette das ermittelte Zeichen
07B5:  CD 53 09     CALL  $0953       HL wieder als Update anmelden
07B8:  F1           POP   AF          das ermittelte Zeichen zurückholen
07B9:  ED 79        OUT   (C),A       und in Zieladresse kopieren
07BB:  ES           EX    DE,HL       Ziel- und Quelladresse wieder ok.
07BC:  C1           POP   BC          Hole Anzahl zurück
07BO:  2B           OEC   HL          Erniedrige den Quellzeiger
07BE:  1B           DEC   DE          Erniedrige den Zielzeiger
07BF:  OB           DEC   BC          Erniedrige den Zähler
07C0:  78           LD    A,B         Feststellen, ob Register-
07C1:  B1           OR    C           paar BC gleich null ist
07C2:  20 E9        JR    NZ,$07AD    nein, dann nächstes Zeichen kopieren
07C4:  2A 11 24     LD    HL,($2411)  sonst Cursoradresse holen
07C7:  C3 05 09     JP    $0905       und ein Leerzeichen ausgeben

****************************************** Zeichen an Cursorposition löschen

07CA:  CD C7 OC     CALL  $OCC7       Hole Cursoradresse und Anzahl Zeichen
07CD:  D5           PUSH  DE          Zeilenanfang auf Stack sichern
07CE:  54           LD    D,H         aktuelle Cursorposition
07CF:  5D           LD    E,L         nach DE kopieren
07D0:  23           INC   HL          Quelle um eins erhöhen
07D1:  CD BO 08     CALL  $08B0       (HL)->(DE) BC mal = Ein Zeichen löschen
07D4:  E1           POP   HL          Zeilenanfang nach HL
0705:  11 4F 00     LD    DE,$004F    und 79 für Zeilenende hinzu-
07D8:  19           ADD   HL,DE       addieren
07D9:  C3 05 09     JP    $0905       am Zeilenende ein Leerzeichen ausgeben

****************************************** An Cursorzeile eine Zeile einfügen
```

```
07DC:  11 62 0F     LD    DE,$0F62      Zeiger auf Tabelle
07DF:  3E 17        LD    A,$17         Zeile 23 in Akku
07E1:  2A 13 24     LD    HL,($2413)    Hole Zeile/Spalte
07E4:  BC           CP    H             Zeile 23 erreicht?
07E5:  CA 05 07     JP    Z,$0705       Ja, dann lösche 23. Zeile
07E8:  38 1A        JR    C,$0804       kleiner als 23, dann Sprung
07EA:  21 E0 06     LD    HL,$06E0      Quelladresse (vorvorletzte Zeile)
07ED:  11 30 07     LD    DE,$0730      Zieladresse (vorletzte Zeile)
07F0:  06 18        LD    B,$18         24 Zeilen sind zu kopieren
07F2:  CD 0A 08     CALL  $080A         Zeile (HL) nach (DE) kopieren
07F5:  3A 14 24     LD    A,($2414)     Hole Zeile
07F8:  B8           CP    B             aktuelle Zeile erreicht?
07F9:  20 F7        JR    NZ,$07F2      Nein, dann weiterkopieren
07FB:  CD C7 0C     CALL  $0CC7         Hole Cursorposition und Anzahl Zeichen
07FE:  EB           EX    DE,HL         HL:Zeilenanfang
07FF:  01 50 00     LD    BC,$0050      80 als Anfang Folgezeile
```

```
0802:  18 30        JR    $0841       Cursorzeile löschen

0804:  3C           INC   A           Diese Stelle wird nie
0805:  BD           CP    L           erreicht, da sich der Cursor
0806:  C0           RET   NZ          dann in der Statuszeile befände
0807:  C3 2C 05     JP    $052C       Text ab DE ausgeben

****************************************** Zeile (HL) nach (DE) kopieren im VDC

080A:  C5           PUSH  BC          Anzahl auf Stack sichern
080B:  E5           PUSH  HL          Quelladresse auf Stack sichern
080C:  D5           PUSH  DE          Zieladresse auf Stack sichern
080D:  01 50 00     LD    BC,$0050    80 Zeichen pro Zeile
0810:  C0 B0 08     CALL  $0880       Zeile kopieren
0813:  01 B0 FF     LD    BC,$FFB0    Komplement von 80 addieren ergibt Subtraktion
0816:  E1           POP   HL          Hole Zieladresse
0817:  09           ADD   HL,BC       subtrahiere 80 Zeichen
0818:  EB           EX    DE,HL       und nach DE
0819:  E1           POP   HL          Hole Quelladresse
081A:  09           ADD   HL,BC       ebenfalls 80 subtrahieren
081B:  C1           POP   BC          Hole Anzhal
081C:  05           DEC   B           den Zähler lediglich erniedrigen
081D:  C9           RET               und Ende der Routine

****************************************** Cursorzeile löschen und Rest raufziehen

081E:  3A 14 24     LD    A,($2414)   Hole Zeile
0821:  FE 18        CP    $18         Ist Zeile 24 erreicht?
0823:  D0           RET   NC          Ja, dann SchluB
0824:  CD C7 0C     CALL  $0CC7       Hole Cursorposition und Anzhal Zeichen
0827:  21 50 00     LD    HL,$0050    80 zur Startadresse der Cursor-
082A:  19           ADD   HL,DE       zeile hinzuaddieren
0828:  EB           EX    DE,HL       und in DE merken
082C:  E5           PUSH  HL          Rette Startadresse auf Stack
082D:  21 80 07     LD    HL,$0780    Startadresse der Statuszeile
```

```
0830:  AF           XOR   A          Lösche Carry für Subtraktion
0831:  ED 52        SBC   HL,DE      Minus Startadresse der Folgezeile
0833:  44           LD    B,H        ergibt Anzahl Zeichen bis Bildschirm-
0834:  4D           LD    C,L        ende nach BC
0835:  E1           POP   HL         Hole Startadresse zurück
0836:  EB           EX    DE,HL      Vertausche Quell- und Zieladresse
0837:  CD B0 08     CALL  $08B0      (HL)->(DE) im VDC-RAM
083A:  C3 05 07     JP    $0705      Letzte Zeile vor Statuszeile löschen

****************************************** VDC-RAM mit Wert füllen

0830:  E1           POP   HL         Rücksprungadresse holen
083E:  E3           EX    (SP),HL    und vorhergehende Rücksprungadresse klschen
083F:  18 06        JR    $0847      Sprung nach Routine

0841:  3A 15 24     LD    A,($2415)  Hole Attribut
0844:  5F           LD    E,A        Attribut in E merken
0845:  16 20        LD    D,$20      ASCII-Code für Leerzeichen

****************************************** (hl) in VDC-RAM mit D füllen, Attribut mit E
                                         (HL+800) wird nur ggf. gefüllt

0847:  78           LD    A,B        Hi-Byte von Anzahl
0848:  A7           AND   A          und Hi-Byte auf null testen
0849:  28 0D        JR    Z,$0858    Ist null, dann nur Lo-Byte füllen
084B:  E5           PUSH  HL         Rette Zieladresse
084C:  D5           PUSH  DE         Rette Füllwerte
084D:  C5           PUSH  BC         Rette Anzahl
084E:  AF           XOR   A          Akku=0 bedeutet 256 Zeichen
084F:  CD 5B 08     CALL  $0858      Fülle (HL) mit D, 256 mal
0852:  C1           POP   BC         Hole Anzahl zurück
0853:  D1           POP   DE         Hole Füllwerte zurück
0854:  E1           POP   HL         Hole Zieladresse zurück
0855:  24           INC   H          Hi-Byte erhöhen
0856:  10 F3        DJNZ  $0848      und falls mehr als 256 Zeichen, dann Sprung
0858:  79           LD    A,C        Teste Lo-Byte auf
```

```
0859:  A7           AND   A          null (keine mehr zu füllen)
085A:  C8           RET   Z          und beende, wenn Schluß

****************************************** Zeichen <D> an <HL> <a> mal speichern

085B:  F5           PUSH  AF         Rette Anzahl
085C:  E5           PUSH  HL         Rette Zieladresse
085D:  D5           PUSH  DE         Rette Füllzeichen
085E:  CD 6C 08     CALL  $086C      Zeichen D speichern
0861:  D1           POP   DE         Hole Füllzeichen
0862:  01 00 08     LD    BC,$0800   Offset für Attribut-RAM
0865:  E1           POP   HL         Zieladresse zurückholen
0866:  09           ADO   HL,BC      und Offset addieren
0867:  CD FE 08     CALL  $08FE      HL auf gültige Adresse abtasten
086A:  Fl           POP   AF         wenn ok, dann hole Zähler
086B:  53           LD    D,E        und Attribut als Füllzeichen

****************************************** Zeichen <O> <A> mal an <HL> im VDC-RAM

086C:  F5           PUSH  AF         Rette Zähler auf Stack
0860:  CD 53 09     CALL  $0953      HL als Update anmelden
0870:  ED 51        OUT   (C),D      und Füllzeichen übergeben
0872:  F1           POP   AF         hole Anzahl von Stack
0873:  3D           DEC   A          erniedrige den Zähler
0874:  C8           RET   Z          und beende, wenn genug gefüllt
0875:  F5           PUSH  AF         sonst rette Zähler
0876:  3E 18        LD    A,$18      Register 24 (Copy-Bit)
0878:  CD 45 09     CALL  $0945      anmelden
087B:  ED 78        IN    A,(C)      Registerinhalt holen
087D:  E6 7F        AND   $7F        und Copy-Bit ausmaksieren
087F:  ED 79        OUT   (C),A      wieder in VDC-Speicher
0881:  3E 1E        LD    A,$1E      Register 31 (Wordcount) auswählen
0883:  CD 45 09     CALL  $0945      und anmelden
0886:  F1           POP   AF         hole Anzahl von Stack
0887:  ED 79        OUT   (C),A      und Restanzahl an VDC übergeben
0889:  06 00        LD    8,$00      Hi-Byte von BC gleich null setzen
```

```
088B:  4F          LD    C,A          und Lo-Byte mit Restanzahl
088C:  03          INC   BC           addiere eins
0880:  09          ADD   HL,BC        addiere Startadresse
088E:  05          PUSH  DE           rette Startadresse
088F:  E5          PUSH  HL           rette errechnete Schlußadresse
0890:  3E 12       LD    A,$12        Update Hi-Byte-Register
0892:  CD 45 09    CALL  $0945        anmelden
0895:  ED 60       IN    H,(C)        und Wert auslesen
0897:  3E 13       LD    A,$13        Update Lo-Byte-Register
0899:  CD 45 09    CALL  $0945        anmelden
089C:  ED 68       IN    L,(C)        und Wert auslesen
089E:  D1          POP   DE           hole errechnete Schlußadresse
089F:  C1          POP   BC           hole Startadresse
08A0:  CD FA 00    CALL  $00FA        Vergleiche HL mit DE
08A3:  D0          RET   NC           Alles klar, kein Fehler!
08A4:  C5          PUSH  BC           Anzahl auf Stack
08A5:  CD 53 09    CALL  $0953        HL als Update anmelden
08A8:  C1          POP   BC           hole Restanzahl
08A9:  ED 41       OUT   (C),B        und Fehlerkorrektur
08AB:  23          INC   HL           falls errechnete Schlußadresse
08AC:  18 F2       JR    $08A0        und tatsächliche Schlußadresse ungleich
08AE:  E1          POP   HL           Rücksprungadresse holen
08AF:  E3          EX    (SP),HL      und um eins tiefer auf Stack

****************************************** (HL) -> (DE) im VDC-RAM <BC> mal

08B0:  78          LD    A,B          Hole Hi-Byte von Anzahl
08B1:  A7          AND   A            uns teste Hi-Byte auf null
08B2:  28 0E       JR    Z,$08C2      Wenn null, dann Anzahl<256
08B4:  E5          PUSH  HL           rette Quelladresse
08B5:  D5          PUSH  DE           Rette Zieladresse
08B6:  C5          PUSH  BC           Rette Anzahl auf Stack
08B7:  AF          XOR   A            Lösche Akku für 256 Zeichen
08B8:  CD C5 08    CALL  $08C5        (HL)->(DE) <A> mal
08BB:  C1          POP   BC           Hole Anzahl
08BC:  D1          POP   DE           Hole Zieladresse
```

```
08BD:  E1           POP   HL          Hole Quelladresse
08BE:  24           INC   H           Hi-Byte der Quelladresse erhöhen
08BF:  14           INC   D           Hi-Byte der Zeiladresse erhöhen
08C0:  10 F2        DJNZ  $0884       und wenn mehr als 256 Zeichen, dann weiter
08C2:  79           LD    A,C         Hole Lo-Anzahl
08C3:  A7           AND   A           teste auf null
08C4:  C8           RET   Z           und Schluß, wenn null
08C5:  EB           EX    DE,HL       sonst vertausche Quell- und Zieladresse
08C6:  F5           PUSH  AF          Rette Anzahl auf Stack
08C7:  E5           PUSH  HL          Rette Zieladresse
08C8:  D5           PUSH  DE          Rette Quelladresse
08C9:  CD D8 08     CALL  $0808       (DE)->(HL) im VDC-RAM <A> mal
08CC:  01 00 08     LD    BC,$0800    Offset für Attribut-RAM
08CF:  E1           POP   HL          Hole Quelladresse
0800:  09           ADD   HL,BC       addiere Offset
08D1:  EB           EX    DE,HL       Quelladresse+Offset nach DE
08D2:  E1           POP   HL          Hole Zieladresse
0803:  09           ADD   HL,BC       addiere Offset
0804:  CD FE 08     CALL  $08FE       Teste auf Speichergrenzen
0807:  F1           POP   AF          Hole Anzahl

****************************************** (DE) -> (HL) im VDC-Speicher <A> mal

08D8:  F5           PUSH  AF          Rette Anzahl auf Stack
08D9:  CD 53 09     CALL  $0953       Melde HL als Updateadresse an
08DC:  3E 18        LD    A,$18       Register 24 (Copy-Bit)
08DE:  CD 45 09     CALL  $0945       anmelden
08E1:  ED 78        IN    A,(C)       Hole Registerinhalt
08E3:  F6 80        OR    $80         und setze das Copybit
08E5:  ED 79        OUT   (C),A       Register an VDC mitteilen
08E7:  3E 20        LD    A,$20       Register 32 (Block-Start-Hi)
08E9:  CD 45 09     CALL  $0945       im VDC anmelden
08EC:  ED 51        OUT   (C),D       Hi-Adresse Quelle übergeben
08EE:  3E 21        LD    A,$21       Register 33 (Block-Start-Lo)
08F0:  CD 45 09     CALL  $0945       in VDC anmelden
08F3:  ED 59        OUT   (C),E       und Lo-Adresse Quelle übergeben
```

```
08F5:  3E 1E        LD    A,$1E       Register 31 (Wordcount)
08F7:  CD 45 09     CALL  $0945       anmelden
08FA:  F1           POP   AF          und Anzahl von Stack holen
08FB:  ED 79        OUT   (C),A       Anzahl VDC mitteilen
08FD:  C9           RET               Ende der Routine

****************************************** Testet, ob nach Addierung des Offset <HL> auf
                                             Attribut-RAM zeigt.

08FE:  7C           LD    A,H         Hi-Byte nach Akku holen
08FF:  FE 20        CP    $20         und auf Grenze checken
0901:  D8           RET   C           Wenn Carry gesetzt, ist <HL> ok
0902:  F1           POP   AF          Hole AF von Stack
0903:  F1           POP   AF          Hole Rücksprungadresse von Stack
0904:  C9           RET               und Rücksprung erfolgt nach CALL $0858

****************************************** Leerzeichen an (HL) ausgeben (VDC)

0905:  16 20        LD    D,$20       ASCII-Wert für <Space>
0907:  3A 15 24     LD    A,($2415)   Attribut holen

****************************************** Zeichen <D> mit Attribut <A> an (HL) ausgeben

090A:  E5           PUSH  HL          Rette Zieladresse
090B:  D5           PUSH  DE          Rette Zeichen/Attribut
090C:  11 00 08     LD    DE,$0800    Offset für Attribut-RAM
090F:  19           ADD   HL,DE       auf Zieladresse addieren
0910:  57           LD    D,A         Attribut als Füllzeichen
0911:  CD 16 09     CALL  $0916       <D> an (HL) ausgeben
0914:  D1           POP   DE          Hole Zeichen
0915:  E1           POP   HL          Hole Zieladresse
0916:  CD 53 09     CALL  $0953       HL als Update anmelden
0919:  ED 51        OUT   (C),D       und Zeichen an (HL) ausgeben
091B:  C9           RET               Ende der Routine

****************************************** Hole <C>:Attribut, <8>:Zeichen, an Cursorpos. <DE>
```

```
091C:  CD AB 06      CALL  $06AB         Cursorposition <DE> setzen
091F:  2A 11 24      LD    HL,($2411)    Hole Cursoradresse
0922:  CD 33 09      CALL  $0933         Hole Zeichen/Attribut
0925:  4F            LD    C,A           <C> ist Attribut
0926:  C9            RET                 Ende der Routine

****************************************** <B>:Zeichen, <C>:Attribut, an <DE> ausgeben

0927:  C5            PUSH  BC            Rette Zeichen/Attribut
0928:  CD AB 06      CALL  $06AB         Cursorposition <DE> setzen
092B:  2A 11 24      LD    HL,($2411)    Hole Cursoradresse
092E:  C1            POP   BC            Hole Zeichen/Atrribut
092F:  50            LD    O,B           <D> ist Zeichen
0930:  79            LD    A,C           <A> ist Attribut
0931:  18 D7         JR    $090A         Zeichen und Attribut ausgeben

****************************************** <A>:Attribut, <B>:Zeichen an (HL) holen

0933:  E5            PUSH  HL            Rette Adresse
0934:  11 00 08      LD    DE,$0800      Offset für Attributadresse
0937:  19            ADD   HL,DE         addieren
0938:  CD 3D 09      CALL  $0930         Hole Wert an VDC-Adresse (HL)
093B:  78            LD    A,B           Attribut nach <A>
093C:  E1            POP   HL            Hole Textadresse
093D:  F5            PUSH  AF            Rette Attribut
093E:  CD 53 09      CALL  $0953         (HL) als Update anmeldën
0941:  F1            POP   AF            Hole Attribut
0942:  ED 40         IN    B,(C)         Hole Wert an Adresse (HL)
0944:  C9            RET                 Ende der Routine
****************************************** VDC-Status abwarten und anw.

0945:  F5            PUSH  AF            Rette auszugebendes Register
0946:  01 00 D6      LD    BC,$D600      Startadresse VDC-Chip
0949:  ED 78         IN    A,(C)         Hole Status
094B:  17            RLA                 Shifte Status-Bit ins Carry
```

```
094C:  30 FB        JR    NC,$0949     Noch nicht fertig -> Sprung
094E:  F1           POP   AF           Hole Register wieder von Stack
094F:  ED 79        OUT   (C),A        und Register anwählen
0951:  0C           INC   C            Zeige auf $0601
0952:  C9           RET                und RETurn aus Routine

****************************************** hL als Update-Adresse

0953:  3E 12        LD    A,$12        Update-Adresse Hi
0955:  CD 45 09     CALL  $0945        anmelden
0958:  ED 61        OUT   (C),H        Hi-Byte übergeben
095A:  3E 13        LD    A,$13        Update-Adresse Lo
095C:  CD 45 09     CALL  $0945        anmelden
095F:  ED 69        OUT   (C),L        Lo-Byte übergeben
0961:  3E 1F        LD    A,$1F        Wordcount-Register
0963:  CD 45 09     CALL  $0945        anmelden
0966:  0D           DEC   C            Zeiger wieder auf $0600
0967:  ED 78        IN    A,(C)        Hole Status
0969:  17           RLA                Rolle Status ins Carry
096A:  30 FB        JR    NC,$0967     Noch nicht fertig
096C:  0C           INC   C            Jetzt ja, Zeiger auf $0601
0960:  C9           RET                RETurn aus Unterroutine

****************************************** Zeichen <D> auf 40-Zeichen-Schirm ausgeben

096E:  42           LD    B,D          Zeichen nach <B>
096F:  CD 7F 0C     CALL  $0C7F        ASCII-Codeumwandlung VIC
0972:  2A 09 24     LD    HL,($2409)   80-Zeichen-Adresse
0975:  47           LD    B,A          Zeichen nach <B>
0976:  3A 10 24     LD    A,($2410)    Zeichenoffset holen (Bit 7 1/0)
0979:  B0           OR    B            mit Zeichen verknüpfen
097A:  77           LD    (HL),A       und ins RAM schreiben
097B:  23           INC   HL           Zeiger erhöhen
097C:  22 09 24     LD    ($2409),HL   und merken
097F:  11 FF 07     LD    DE,$07FF     Offset für Farbrem
0982:  19           ADD   HL,DE        hinzuaddieren
```

```
0983:  3A 00 24      LD    A,($240D)    Hole Zeichenfarbe
0986:  77            LD    (HL),A       und Zeichenfarbe setzen
0987:  3A 0B 24      LD    A,(S2408)    Hole Cursorposition
098A:  FE 4F         CP    $4F          Letzte Spalte?
098C:  28 5F         JR    Z,$09ED      Ja, dann Zeilensprung
098E:  3C            INC   A            sonst erhöhe den Spaltenzeiger
098F:  32 0B 24      LD    ($240B),A    und merke neue Position
0992:  C3 4A 0C      JP    S0C4A        Zeile darstellen

****************************************** Hole  Zeichen <B> und Farbe <C> von  Cursorpos
(DE)

0995:  CD C1 09      CALL  $09C1        Zeile/Spalte definieren (DE)
0998:  2A 09 24      LD    HL,($2409)   Holt Cursoradresse 80-Zeichen-Simulator
099B:  46            LD    B,(HL)       Hole Zeichen an Cursorposition
099C:  11 00 08      LD    DE,$0800     Offset für Attribut-RAM
099F:  19            ADD   HL,DE        addieren
09A0:  4E            LD    C,(HL)       Hole Attribut an Cursorposition
09A1:  C9            RET                Ende der Routine

****************************************** <B>:Zeichen, <C>:Attribut an (DE)

09A2:  C5            PUSH  BC           Rette Zeichen/Attribut
09A3:  CD C1 09      CALL  $09C1        Setze Cursorposition
09A6:  C1            POP   BC           Hole Zeichen/Attribut
09A7:  2A 09 24      LD    HL,($2409)   Hole 80-Zeichen-Simulator-Adresse
09AA:  78            LD    A,B          Zeichen in <Akku>
09AB:  E6 7F         AND   $7F          Bit 7 löschen
09AD:  CB 71         BIT   6,C          Bit 6 testen
09AF:  28 02         JR    Z,$09B3      Ist ungesetzt
09B1:  C6 80         ADD   A,$80        Bit 7 setzen (reverses Zeichen)
09B3:  77            LD    (HL),A       und Zeichen setzen
09B4:  11 00 08      LD    DE,$0800     Offset für Attribut-RAM
09B7:  19            ADD   HL,DE        addieren
09B8:  71            LD    (HL),C       Attribut ebenfalls definieren
09B9:  C3 4A 0C      JP    $0C4A        Zeile darstellen
```

```
****************************************** Def. Zeile/Spalte

09BC:  21 04 24      LD    HL,$2404     Adresse 40-Zeichen-Anpassung
09BF:  CB F6         SET   6,(HL)       40-Zeichen-Bit setzen
09C1:  7A            LD    A,D          Hole Zeile
09C2:  FE 19         CP    $19          Größer als 24?
09C4:  D0            RET   NC           Ja, dann Ende (Fehler)
09C5:  7B            LD    A,E          Hole Spalte
09C6:  FE 50         CP    $50          Spalte 80 erreicht?
09C8:  D0            RET   NC           Ja, dann Ende (Fehler)
09C9:  EB            EX    DE,HL        Zeile/Spalte nach HL
09CA:  22 0B 24      LD    ($2408),HL   Setze Zeile/Spalte

****************************************** Neue Cursorposition

09CD:  2A 0B 24      LD    HL,($2408)   Hole Zeile/Spalte
09D0:  CD CE 0C      CALL  SOCCE        Neue Cursoradresse berechnen
09D3:  11 00 14      LD    DE,$1400     Offset für 80-Zeichen-Simulator
09D6:  19            ADD   HL,DE        hinzuaddieren
09D7:  22 09 24      LD    ($2409),HL   und angepaßte Adresse merken
09DA:  C3 4A 0C      JP    $0C4A        Zeile darstellen

****************************************** Zeile erniedrigen

09DD:  3A 0C 24      LD    A,($240C)    Hole Cursorzeile
09E0:  B7            OR    A            Setze Flags
09E1:  C8            RET   Z            Zeile 0! Davor geht nicht
09E2:  3D            DEC   A            sonst erniedrige Zeilenzeiger
09E3:  32 0C 24      LD    ($240C),A    und merke Zeile
09E6:  21 04 24      LD    HL,$2404     Setze Bit 6 an $2404
09E9:  CB F6         SET   6,(HL)       als OK-Zeichen
09EB:  18 E0         JR    S09CD        Neue Cursorposition berechnen

****************************************** Spalte=0 bzw. Spalte definieren (+1)
```

```
09ED:  AF          XOR   A           Akku=0 für erste Spalte
09EE:  32 0B 24    LD    ($240B),A   und Spalte definieren
09F1:  3A 0C 24    LD    A,($240C)   Hole Zeile
09F4:  FE 17       CP    $17         letzte Zeile?
09F6:  28 0A       JR    Z,$0A02     Ja, dann Sprung
09F8:  30 03       JR    NC,$09F0    Fehler korrigieren
09FA:  3C          INC   A           Zeile ums eins erhöhen
09FB:  18 E6       JR    $09E3       und merken

09F0:  3E 17       LD    A,$17       Zeile 23 (letzte Zeile)
09FF:  32 0C 24    LD    ($240C),A   merken
0A02:  21 50 14    LD    HL,$1450    Zweite Zeile Anfangsadresse
0A05:  11 00 14    LD    DE,$1400    Erste Zeile Anfangsadresse
0A08:  01 30 07    LD    BC,$0730    22 Zeilen zu kopieren
0A0B:  ED B0       LDIR               Scrolling
0A0D:  EB          EX    DE,HL       Zieladresse als Quelle
0A0E:  11 31 1B    LD    DE,$1B31    Zweites Zeichen letzte Zeile
0A11:  01 4F 00    LD    BC,$004F    79 Zeichen sind zu füllen
0A14:  CD 24 0B    CALL  $0824       Fülle letzte Zeile mit Füllzeichen
0A17:  21 50 1C    LD    HL,$1C50    Anfangsadresse zweite Zeile (Attribut)
0A1A:  11 00 1C    LD    DE,$1C00    Anfangsadresse erste Zeile (Attribut)
0A10:  01 30 07    LD    BC,$0730    22 Zeilen sind zu scrollen
0A20:  ED B0       LDIR               Scrollen im Farbram durchführen
0A22:  EB          EX    DE,HL       erstes Zeichen letzte Zeile nach HL
0A23:  11 31 23    LD    DE,$2331    Zweites Zeichen letzte Zeile (Farbram)
0A26:  01 4F 00    LD    BC,$004F    79 Zeichen sind zu füllen
0A29:  3A 0D 24    LD    A,($2400)   Farbe für Farbram holen
0A2C:  77          LD    (HL),A      setzen
0A2D:  ED B0       LDIR               und restliche Zeile auch füllen
0A2F:  18 B5       JR    $09E6       OK setzen

****************************************** Cursor um eine Stelle nach links

0A31:  3A 0B 24    LD    A,($2408)   Hole Spaltenposition
0A34:  B7          OR    A           Setze Flags
0A35:  C8          RET   Z           Erste Spalte, dann Ende
```

```
0A36:  3D           DEC   A           sonst Cursor nach links
0A37:  32 0B 24     LD    ($240B),A   Abspeichern der neuen Spalte
0A3A:  18 91        JR    $09CD       Cursoradresse berechnen

****************************************** Cursor um eine Stelle nach rechts

0A3C:  3A 0B 24     LD    A,($240B)   Hole Spalte
0A3F:  3C           INC   A           und um eine Stelle nach rechts
0A40:  FE 50        CP    $50         80. Spalte erreicht?
0A42:  20 F3        JR    NZ,$0A37    Nein, dann abspeichern neue Position
0A44:  C9           RET               sonst keine Cursorbewegung erfolgt

****************************************** Spalte = 0 setzen

0A45:  AF           XOR   A           Akku löschen und
0A46:  18 EF        JR    $0A37       als Spaltenwert abspeichern

****************************************** Cursorpos. bis Zeilenende löschen

0A48:  21 CF 0B     LD    HL,$0BCF    Return nach
0A4B:  E5           PUSH  HL          $0BCF simulieren
0A4C:  CD C2 0C     CALL  $0CC2       Cursorpos. ermitteln und Restzeichen/Zeile
0A4F:  11 00 14     LD    DE,$1400    Offset für 80-Zeichen-Simulator
0A52:  19           ADD   HL,DE       addieren
0A53:  CD B3 0A     CALL  $0AB3       Füllzeichen setzen
0A56:  79           LD    A,C         Restzeichen/Zeile
0A57:  A7           AND   A           Setze Flags
0A58:  C8           RET   Z           Keine weiteren Zeichen
0A59:  C5           PUSH  BC          Rette Anzahl
0A5A:  E5           PUSH  HL          Rette Quelladresse
0A5B:  54           LD    D,H         Registerpaar DE
0A5C:  5D           LD    E,L         gleich HL
0A5D:  13           INC   DE          plus 1
0A5E:  ED B0        LDIR              Rest bis Zeilenende löschen
0A60:  18 1C        JR    $0A7E       Attribut ebenfalls löschen
```

```
****************************************** Cursorpos. bis Sildschirmende löschen

0A62:  21 0C 0C      LD    HL,$0C0C     $0C0C als Rücksprungadresse
0A65:  E5            PUSH  HL           auf $tack simulieren
0A66:  11 7F 1B      LD    DE,$1B7F     letzte Spalte in letzter Zeile
0A69:  2A 09 24      LD    HL,($2409)   Hole Cursoradresse 80-Zeichen-Simulator
0A6C:  EB            EX    DE,HL        nach DE, $1B7F nach HL
0A6D:  AF            XOR   A            Carry für Subtraktion löschen
0A6E:  ED 52         SBC   HL,DE        Ermittle Anzahl Zeichen bis Bildende
0A70:  F8            RET   M            wenn negativ, dann Fehler
0A71:  EB            EX    DE,HL        sonst Ergebnis nach DE
0A72:  28 3F         JR    Z,$0AB3      wenn nur ein Zeichen, dann füllen
0A74:  42            LD    B,D          sonst Anzahl
0A75:  4B            LD    C,E          ins Registerpaar BC
0A76:  54            LD    D,H          und Registerpaar DE (Ziel)
0A77:  5D            LD    E,L          gleich Registerpaar HL
0A78:  13            INC   DE           plus eins
0A79:  C5            PUSH  BC           Anzahl auf Stack
0A7A:  E5            PUSH  HL           Quelle auf Stack
0A7B:  CD 24 0B      CALL  $0824        Fülle Textzeile mit Attribut
0A7E:  01 00 08      LO    BC,$0800     Offset für Farbram
0A81:  E1            POP   HL           Quelladresse zurückholen
0A82:  09            ADD   HL,BC        und Offset addieren
0A83:  C1            POP   BC           Hole Anzahl Zeichen
0A84:  54            LD    D,H          Zieladresse gleich
0A85:  50            LD    E,L          Quelladresse
0A86:  13            INC   DE           plus eins
0A87:  3A 0D 24      LD    A,($240D)    Hole Farbe für Farbram VIC
0A8A:  77            LD    (HL),A       und Farbe setzen
0A8B:  ED B0         LDIR               Rest bis Bildende füllen
0A8D:  C9            RET                Ende der Routine

****************************************** an Cursorpos. 1 Stelle einfügen

0A8E:  21 CF 0B      LD    HL,$0BCF     Zeile aus RAM in Bildschirm kopieren
0A91:  E5            PUSH  HL           als Rücksprungadresse auf Stack
```

```
0A92:  CD C2 0C      CALL  $0CC2         Cursorpos. und Restzeichen ermitteln
0A95:  21 4F 14      LD    HL,$144F      Letztes Zeichen erste Zeile
0A98:  19            ADD   HL,DE         zur Cursoradresse addieren
0A99:  3D            DEC   A             letzte Spalte?
0A9A:  28 17         JR    2,$0AB3       Ja, dann überspringe
0A9C:  54            LD    D,H           Sonst Zieladresse gleich
0A9D:  5D            LD    E,L           Quelladresse
0A9E:  2B            DEC   HL            minus eins
0A9F:  C5            PUSH  BC            Rette Anzahl auf Stack
0AA0:  D5            PUSH  DE            Rette Zieladresse auf Stack
0AA1:  ED B8         LDDR                Zeichen hinter Cursor nach rechts verschieben
0AA3:  EB            EX    DE,HL         HL:=Cursorpos
0AA4:  CD B3 0A      CALL  $0AB3         Hole Füllzeichen
0AA7:  E1            POP   HL            Hole Zieladresse zurück
0AA8:  01 00 08      LD    BC,$0800      Offset für Farbram
0AAB:  09            ADD   HL,BC         zur Quelladresse hinzuaddieren
0AAC:  C1            POP   BC            Anzahl zurückholen
0AAD:  54            LD    D,H           Zieladresse gleich
0AAE:  5D            LD    E,L           Quelladresse
0AAF:  2B            DEC   HL            minus eins
0AB0:  ED B8         LDDR                Farbram ebenfalls verschieben
0AB2:  C9            RET                 Ende der Routine

****************************************** ($2410) + $20 -> (HL); Füllzeichen setzen

0AB3:  3A 10 24      LD    A,($2410)     Adresse $2410 auslesen (Füllzeichen)
0AB6:  C6 20         ADD   A,$20         $20=32 hinzuaddieren
0AB8:  77            LD    (HL),A        und in (HL) ablegen
0AB9:  C9            RET                 RETurn aus Routine

****************************************** 1 Zeichen an Cursorpos. löschen

0ABA:  21 CF 0B      LD    HL,$0BCF      $0BCF als Rücksprungadresse auf Stack
0ABD:  E5            PUSH  HL            zusätzlich einfügen
0ASE:  CD C2 0C      CALL  $0CC2         Hole Cursoredresse/Restzeichen
0AC1:  11 00 14      LD    DE,$1400      Offset für 80-Zeichen-Simulator
```

```
0AC4:  19          ADD   HL,DE       zur Cursoradresse hinzuaddieren
0AC5:  3D          DEC   A           Teste Anzahl/Zeichen
0AC6:  28 EB       JR    Z,$0AB3     Wenn null, dann überspringe
0AC8:  54          LD    D,H         sonst Zieladresse gleich
0AC9:  5D          LD    E,L         Quelladresse
0ACA:  C5          PUSH  BC          Rette Anzahl
0ACB:  E5          PUSH  HL          Rette Quelladresse
0ACC:  23          INC   HL          Quelladresse=Quelladresse+1
0ACD:  ED B0       LDIR              Zeichen an Cursorpos. löschen
0ACF:  EB          EX    DE,HL       Cursoradresse nach HL
0AD0:  CD B3 0A    CALL  $0AB3       Füllzeichen setzen
0AD3:  E1          POP   HL          Hole Quelladresse
0AD4:  01 01 08    LD    BC,$0801    Offset für Farbram+1
0AD7:  09          ADD   HL,BC       addieren
0AD8:  C1          POP   BC          Hole Zeichenzahl von Stack
0AD9:  54          LD    D,H         Zieladresse gleich
0ADA:  50          LD    E,L         Quelladresse
0ADB:  23          INC   HL          plus eins
0ADC:  ED B0       LDIR              Farbram ebenfalls verschieben
0ADE:  C9          RET               Ende der Routine

****************************************** An Cursorpos. 1 Zeile einfügen

0ADF:  21 0C 0C    LD    HL,$0C0C    Rücksprungsadresse $0C0C
0AE2:  E5          PUSH  HL          auf Stapel einfügen
0AE3:  3A 0C 24    LD    A,($240C)   Hole Cursorzeile
0AE6:  FE 17       CP    $17         Zeile 23 (letzte)?
0AE8:  28 31       JR    Z,$0B1B     Ja, dann nur löschen
0AEA:  D0          RET   NC          Fehler, dann Ende
0AEB:  CD C2 0C    CALL  $0CC2       Hole Cursoradresse/Restzeichen
0AEE:  21 00 14    LD    HL,$1400    Offset für 80-Zeichen-Simulator
0AF1:  19          ADD   HL,DE       hinzuaddieren
0AF2:  E5          PUSH  HL          Rette Quelladresse auf Stapel
0AF3:  11 50 00    LD    DE,$0050    Offset für Anfang nächste Zeile
0AF6:  19          ADD   HL,DE       zur Quelladresse hibzuaddieren
0AF7:  EB          EX    DE,HL       Ergebnis nach (DE)
```

```
0AF8:  21 80 1B     LD    HL,$1880    Adresse letzte Zeile
0AFB:  AF           XOR   A           Carry löschen für Subtraktion
0AFC:  ED 52        SBC   HL,DE       Ermittelt Anzahl Zeichen bis Bildschirmende
0AFE:  44           LD    B,H         Anzahl kommt von
0AFF:  4D           LD    C,L         HL nach BC
0B00:  21 2F 1B     LD    HL,$1B2F    letzte Spalte vorletzte Zeile
0B03:  11 7F 1B     LD    DE,$1B7F    letzte Spalte letzte Zeile
0B06:  C5           PUSH  BC          Rette Anzahl auf Stapel
0B07:  ED B8        LDDR              Zeile an Cursorzeile einfügen
0B09:  C1           POP   BC          Hole ANzahl zurück
0B0A:  21 2F 23     LD    HL,$232F    Adresse Farbram
0B0D:  11 7F 23     LD    DE,$237F    Adresse Farbram (s.o.)
0B10:  ED B8        LDDR              ebenfalls verschieben
0B12:  E1           POP   HL          Hole Quelladresse
0B13:  54           LD    D,H         DE gleich
0B14:  5D           LD    E,L         Quelladresse
0B15:  13           INC   DE          plus eins
0B16:  01 4F 00     LD    BC,$004F    79 Zeichen
0B19:  18 09        JR    $0824       Lösche neue Zeile

0B1B:  21 30 1B     LD    HL,$1B30    lediglich die letzte
0B1E:  11 31 1B     LD    DE,$1B31    Zeile löschen,
0B21:  01 4F 00     LD    BC,$004F    da Cursor in letzter Zeile ist
0B24:  3A 10 24     LD    A,($2410)   Hole Füllzeichen
0B27:  C6 20        ADD   A,$20       addiere 32 (Leerzeichen)
0B29:  77           LD    (HL),A      und Füllzeichen setzen
0B2A:  ED B0        LDIR              Rest mit Füllzeichen füllen
0B2C:  C9           RET               und Ende der Routine

****************************************** Lösche Cursorzeile inkl. Bildschirmverschieben

0B2D:  21 0C 0C     LD    HL,$0C0C    Rücksprungsadresse $0C0C
0B30:  E5           PUSH  HL          auf Stapel hinzufügen
0B31:  3A 0C 24     LD    A,($240C)   Hole Cursorzeile
0B34:  FE 17        CP    $17         Letzte Zeile?
0B36:  28 E3        JR    Z,$0B18     Ja, dann lediglich letzte Zeile löschen
```

```
0B38:  D0           RET   NC          Bei NC Fehler und Ende
0B39:  CD C2 0C     CALL  $0CC2       Errechne Cursoradresse/Restzeichen
0B3C:  21 00 14     LD    HL,$1400    Offset für 80-Zeichen-Simulator
0B3F:  19           ADD   HL,DE       zur Cursoradresse addieren
0B40:  E5           PUSH  HL          Rette Quelladresse auf Stapel
0B41:  11 50 00     LD    DE,$0050    Offset für Start Folgezeile
0B44:  19           ADD   HL,DE       addieren
0B45:  EB           EX    DE,HL       Ergebnis nach DE
0B46:  21 80 1B     LD    HL,$1B80    Adresse Statuszeile
0B49:  AF           XOR   A           Lösche Carry für Subtraktion
0B4A:  ED 52        SBC   HL,DE       Ermiitelt Anzahl Zeichen bis Bildende
0B4C:  44           LD    B,H         Anzahl
0B4D:  4D           LD    C,L         nach BC
0B4E:  EB           EX    DE,HL       Startadresse->HL
0B4F:  D1           POP   DE          Hole Startadresse Cursorzeile
0B50:  C5           PUSH  BC          Rette Anzahl auf Stapel
0B51:  E5           PUSH  HL          Rette Quelladresse auf Stapel
0B52:  D5           PUSH  DE          Rette Zieladresse auf Stack
0B53:  ED B0        LDIR              Und Zeile löschen
0B55:  01 00 08     LD    BC,$0800    Offset für Farbram
0B58:  E1           POP   HL          zur Quelladresse
0B59:  09           ADD   HL,BC       hinzuaddieren
0B5A:  EB           EX    DE,HL       Ergebnis nach DE
0B5B:  E1           POP   HL          Hole Startadresse vorletzte Zeile Farbram
0B5C:  09           ADD   HL,BC       ebenfalls Offset addieren
0B5D:  C1           POP   BC          Hole Anzahl
0B5E:  ED B0        LDIR              und Farbram auch verschieben
0B60:  18 B9        JR    $081B       letzte Zeile löschen

****************************************** <B>:auszuschaltende, <C>:einzuschaltende Bits
                                           bei Zeichenfarbe

0B62:  78           LD    A,B         auszuschaltende Bits nach <A>
0B63:  E6 70        AND   $70         Bit 7 und Bits 0-3 löschen
0B65:  47           LD    B,A         Ergebnis nach <B>
0B66:  79           LD    A,C         Hole zu setzende Bits
```

```
0867:  E6 70        AND   $70          Ebenfalls Bits 0,1,2,3,4,7 löschen
0869:  4F           LD    C,A          Ergebnis nach C
086A:  3A 00 24     LD    A,($240D)    Hole Attribut
086D:  2F           CPL                komplementieren und
086E:  B0           OR    B            zu löschende Eigenschaften setzen
086F:  2F           CPL                erneut komplementieren und
0870:  B1           OR    C            zu setzende Eigenschaften setzen
0871:  32 00 24     LD    ($2400),A    neues Attribut merken
0874:  17           RLA                6. Bit ins 7. shlften
0875:  E6 80        AND   $80          (Reverszeichen) und ausmaskieren
0877:  32 10 24     LD    ($2410),A    $00 oder $80 als Füllzeichen
087A:  C9           RET                Ende der Routine

******************************************

087B:  78           LD    A,B          Hole auszugebendes Zeichen nach <Akku>
087C:  06 20        SUB   $20          Minus ASCII 32
087E:  FE 30        CP    $30          kleiner als 48?
0880:  38 0E        JR    C,$0B90      Ja, dann $prung
0882:  0E 30        LD    C,$30        sonst merke 48 als abgezogen
0884:  CD E5 0C     CALL  $0CE5        weitere Dekodierung
0887:  D8           RET   C            Alles klar
0888:  7E           LD    A,(HL)       Hole Farbe
0889:  0F           RRCA               /2
088A:  0F           RRCA               /2=/4
088B:  0F           RRCA               /2=/8
088C:  0F           RRCA               /2=/16
088D:  E6 0F        AND   $0F          Bits 4-7 ausmaskieren
088F:  80           ADD   A,B          und <B> wieder hinzuaddieren
0890:  FE 10        CP    $10          Übertrag ins höherwertige Nibble?
0892:  38 29        JR    C,$0BBD      Nein, neue Farbe definieren
0894:  FE 20        CP    $20          Rahmenfarbe oder Hintergrundfarbe?
0896:  38 0B        JR    C,$0BA3      Hintergrundfarbe definieren

****************************************** Rahmenfarbe setzen
```

```
0B98:  E6 0F        AND   $0F          Bits 7-4 ausmaskieren
0B9A:  32 0F 24     LD    ($240F),A    und Rahmenfarbe merken
0B9D:  01 20 D0     LD    BC,$D020     Adresse für Rahmenfarbe
0BA0:  ED 79        OUT   (C),A        Rahmenfarbe an VIC übergeben
0BA2:  C9           RET                Ende der Routine

****************************************** Hintergrundfarbe 40-Zeichen setzen

0BA3:  E6 0F        AND   $0F          Bits 4-7 ausmaskieren
0BA5:  32 0E 24     LD    ($240E),A    und Hintergrundfarbe merken
0BA8:  01 21 D0     LD    BC,$0021     Adresse für Hintergrundfarbe
0BAB:  ED 79        OUT   (C),A        Hintergrundfarbe an VIC übergeben
0BA0:  C9           RET                Ende der Routine

****************************************** Hole: <B>:Hintergrund, <C>:Rahmen, <D>:Zeichenfarbe

0BAE:  3A 0E 24     LD    A,($240E)    Hole Hintergrundfarbe
0BB1:  47           LD    B,A          in <B> merken
0BB2:  3A 0F 24     LD    A,($240F)    Hole Rahm
0BB5:  4F           LD    C,A          in <C> merken
0BB6:  3A 0D 24     LD    A,($240D)    Hole Zeichenfarbe
0BB9:  57           LD    D,A          in <D> merken
0BBA:  E6 0F        AND   $0F          unnötige Bits 7-4 ausmaskieren
0BBC:  C9           RET                Ende der Routine

****************************************** neue Farbe definieren

0BBD:  47           LD    B,A          Code nach <B>
0BBE:  3A 0D 24     LD    A,($240D)    Hole alte Farbe
0BC1:  E6 F0        AND   $F0          Bits 0 bis 3 ausmaskieren
0BC3:  B0           OR    B            und neue Farbe reinORen
0BC4:  32 0D 24     LD    ($2400),A    Neue Farbe abspeichern
0BC7:  2A 09 24     LD    HL,($2409)   Hole Cursoradresse
0BCA:  11 00 08     LD    DE,$0800     Offset für Attribut-RAM
0BCD:  19           ADD   HL,DE        addieren
0BCE:  77           LD    (HL),A       neue Farbe setzen
```

```
****************************************** Zeile aus RAM in Bildschirm kopieren

0BCF:  3A 04 24     LD    A,(S2404)     Merker für Ausgabe
0BD2:  47           LD    B,A           nach <8>
0BD3:  B7           OR    A             Setze Flags
0BD4:  FC 3C 0C     CALL  M,$0C3C       Wenn Bit 7 gesetzt, dann anpassen
0BD7:  3A 02 24     LD    A,($2402)     Hole angepaßte Spalte
0BDA:  B8           CP    B             gleich mit 80-Zeichen-Spalte?
0BDB:  32 04 24     LD    ($2404),A     merke Spalte
0B0E:  20 2C        JR    NZ,$0C0C      ungleich, dann Sprung
0BE0:  CD C2 0C     CALL  $0CC2         Cursorposition berechnen
08E3:  21 00 14     LD    HL,$1400      Offset $1400
0BE6:  19           ADD   HL,DE         addieren
0BE7:  EB           EX    DE,HL         und in DE merken
0BE8:  2A 02 24     LD    HL,($2402)    plus der angepaßten Spalte
0BEB:  19           ADD   HL,DE         ergibt Zieladresse
0BEC:  E5           PUSH  HL            auf Stack sichern
0BED:  3A 0C 24     LD    A,($240C)     Hole Zeile
0BF0:  6F           LD    L,A           und nach <L>
0BF1:  CD 70 0C     CALL  $0C70         Zeilenbeginn für 40-Zeichen berechnen
0BF4:  EB           EX    DE,HL         und in DE merken
0BF5:  E1           POP   HL            angepaßte Adresse holen
0BF6:  E5           PUSH  HL            und wieder retten
0BF7:  D5           PUSH  DE            rette Zeilenbeginn
0BF8:  3E 01        LD    A,$01         1 Zeile ist zu kopieren
0BFA:  CD 27 0C     CALL  $0C27         kopiere (S1400+SP) in Bildschirm
0BFD:  E1           POP   HL            40-Zeichen-Adresse zurück
0BFE:  01 00 E4     LD    BC,$E400      Offset für Farbram
```

```
0C01:  09           ADD   HL,BC       Offset addieren
0C02:  EB           EX    DE,HL       und in DE merken
0C03:  E1           POP   HL          hole Quelladresse
0C04:  01 00 08     LD    BC,$0800    Offset für Bildschirmfarbram
0C07:  09           ADD   HL,BC       zu Quelladresse addieren
0C08:  3E 01        LD    A,$01       1 Zeile
0C0A:  18 1B        JR    $0C27       und kopieren

****************************************** 40-Zeichen-Bildschirm kopieren

0C0C:  2A 02 24     LD    HL,($2402)  Hole angepaßte Cursoradresse
0C0F:  3A 08 24     LD    A,($2408)   Anzahl darzustellender Zeichen/Zeile
0C12:  E5           PUSH  HL          Rette Cursoradresse
0C13:  F5           PUSH  AF          Rette Anzahl/Zeichen
0014:  11 00 14     LD    DE,$1400    Offset für 80-Zeichen-Simulation
0C17:  19           ADD   HL,DE       addieren
0C18:  11 00 2C     LD    DE,$2C00    tats. Videoram
0C1B:  CD 27 0C     CALL  $0C27       Zeile kopieren
0C1E:  F1           POP   AF          Hole Anzahl
0C1F:  E1           POP   HL          Hole Quelladresse
0C20:  11 00 1C     LD    DE,$1C00    Offset für Farbram
0C23:  19           ADD   HL,DE       addieren
0C24:  11 00 10     LD    DE,$1000    Farbramadresse (angepaßt)

****************************************** (HL) -> (DE) 40 Zeichen in Bildschirm <A> Zeilen

0C27:  32 03 FF     LD    ($FF03),A   PCRC als Konfigurationsbyte
0C2A:  01 28 00     LD    BC,$0028    40 Zeichen sind zu kopieren
0C2D:  ED B0        LDIR              (HL) -> (DE)
0C2F:  D5           PUSH  DE          rette Zieladresse
0C30:  11 28 00     LD    DE,$0028    40 Zeichen sind kopiert worden
0C33:  19           ADD   HL,DE       addiere zu Quelladresse
0C34:  D1           POP   DE          hole Zieladresse
0C35:  3D           DEC   A           erniedrige den Zähler
0C36:  20 F2        JR    NZ,$0C2A    noch eine Zeile zu kopieren
```

```
0C38:  32 01 FF      LD    ($FF01),A    sonst PCRA als Konfigurationsbyte
0C3B:  C9            RET                Ende der Routine

****************************************** Spalte anpassen

0C3C:  3A 0B 24      LD    A,(S240B)    Hole Spalte
0C3F:  D6 20         SUB   $20          minus 32
0C41:  30 01         JR    NC,$0C44     Kein Übertrag entstanden
0C43:  AF            XOR   A            sonst lösche Akku (=0)
0C44:  E6 F8         AND   $F8          Bits 0-2 ausmaskieren
0C46:  32 02 24      LD    ($2402),A    angepaßte Spalte merken
0C49:  C9            RET                Ende der Routine

****************************************** Zeile kopieren

0C4A:  CD 69 0C      CALL  $0C69        $2406 löschen
0C4D:  CD CF 0B      CALL  $08CF        Zeile kopieren
0C50:  3A 02 24      LD    A,($2402)    Hole angepaßte Spalte
0C53:  47            LD    B,A          in <B> merken
0C54:  2A 0B 24      LD    HL,($240B)   Hole Zeile/Spalte
0C57:  70            LD    A,L          Spalte nach <A> zwecks Subtraktion
0C58:  90            SUB   B            minus angepaßter Spalte
0C59:  38 0E         JR    C,$0C69      zu klein, dann $2406 löschen
0C5B:  FE 28         CP    $28          zu groß - größer als 40?
0C5D:  30 0A         JR    NC,S0C69     dann $2406 löschen
0C5F:  4F            LD    C,A          Spalte nach <C>
0C60:  06 00         LD    B,$00        Hi-Byte von BC löschen
0C62:  6C            LD    L,H          <L>=Zeile
0C63:  CD 70 0C      CALL  $0C70        Und Startadresse Zeile berechnen
0C66:  09            ADD   HL,BC        addiere Spalte
0C67:  18 03         JR    $0C6C        und merken der Adresse
0C69:  21 00 00      LD    HL,$0000     Startadresse ist erste Position
0C6C:  22 06 24      LD    ($2406),HL   merken der Position
0C6F:  C9            RET                Ende der Routine

****************************************** Zeile*40 + Offset
```

```
0C70:  26 00        LD    H,$00       Hi-Byte löschen
0C72:  29           ADD   HL,HL       *2
0C73:  29           ADD   HL,HL       *2
0C74:  29           ADD   HL,HL       *2=*8
0C75:  54           LD    D,H         nach DE
0C76:  50           LD    E,L         merken
0C77:  29           ADD   HL,HL       *2
0C78:  29           ADD   HL,HL       *2=*32
0C79:  19           ADD   HL,DE       *32+*8 ergibt *40
0C7A:  11 00 2C     LD    DE,$2C00    Offset von $2C00 (Textanfang)
0C7D:  19           ADD   HL,DE       addieren
0C7E:  C9           RET               Ende der Routine

****************************************** ASCII-Code-Anpassung für 40-Zeichen

0C7F:  78           LD    A,B         Zeichen nach <Akku>
0C80:  FE 40        CP    $40         Ist es ASCII 64 (@)
0C82:  28 3C        JR    Z,$0CC0     Ja, dann Poke-Code=0
0C84:  08           RET   C           Bei kleiner, Ende
0C85:  FE 5B        CP    $5B         kleiner als ASCII "Z"+1?
0C87:  D8           RET   C           Ja, dann Rückkehr
0C88:  D6 40        SUB   $40         64 abziehen
0C8A:  FE 20        CP    $20         Ist es Apostroph?
0C8C:  28 19        JR    Z,$0CA7     Ja, dann Codiere
0C8E:  38 23        JR    C,$0CB3     Kleiner als Apostroph
0C90:  D6 20        SUB   $20         minus ASCII 32
0C92:  FE 1B        CP    $1B         Grenze kleine Buchstaben
0C94:  08           RET   C           Kleiner, dann Ende
0C95:  FE 1B        CP    $1B         erneut vergleichen
0C97:  28 11        JR    Z,$0CAA     kleiner ASCII "ä"?
0C99:  FE 1C        CP    $1C         kleines "ö"?
0C9B:  28 10        JR    Z,$0CAD     Ja, dann codiere
0C90:  FE 1D        CP    $10         kleines "ü"?
0C9F:  28 0F        JR    Z,$0CB0     Ja, dann codiere
0CA1:  FE 1E        CP    $1E         Ist es "B"?
```

```
0CA3:  C0          RET   NZ          Nein, dann Rückkehr
0CA4:  3E 40       LD    A,$40       Sonst codiere 64
0CA6:  C9          RET               und Ende der Routine

****************************************** ASCII-Code 126 generieren

0CA7:  3E 7E       LD    A,$7E       126
0CA9:  C9          RET               Ende der Routine

****************************************** ASCII-Code 115

0CAA:  3E 73       LD    A,$73       115
0CAC:  C9          RET               Ende der Routine

****************************************** ASCII-Code 93

0CAD:  3E 50       LD    A,$5D       93
0CAF:  C9          RET               Ende der Routine

****************************************** ASCII-Code 107

0CB0:  3E 68       LD    A,$6B       107
0CB2:  C9          RET               Ende der Routine

****************************************** ASCII-Code 28

0CB3:  FE 1C       CP    $1C         ASCII 28?
0CB5:  28 06       JR    Z,$0CB0     Ja, dann zuordnen
0CB7:  FE 1F       CP    $1F         ASCII 31?
0CB9:  C0          RET   NZ          Nein, dann Ende mit Flag
0CBA:  3E 64       LD    A,$64       Sonst ASCII 100
0CBC:  C9          RET               Ende der Routine

****************************************** ASCII-Code 127 zuordnen

0CBD:  3E 7F       LD    A,$7F       127
```

```
0CBF:  C9          RET              Ende der Routine

****************************************** <Akku> löschen

0CC0:  AF          XOR   A          <Akku> löschen
0CC1:  C9          RET              Ende der Routine

****************************************** <C>:79-Spalte und Cursorpos. berechnen

0CC2:  2A 0B 24    LD    HL,($240B) Hole Zeile/Spalte
0CC5:  18 03       JR    SOCCA      weitermachen

0CC7:  2A 13 24    LD    HL,($2413) Hole Zeile/Spalte
0CCA:  3E 4F       LD    A,$4F      dezimal 79
0CCC:  95          SUB   L          minus Spalte
0CCD:  4F          LD    C,A        in <C> merken (Restanzahl bis 80)

****************************************** Cursorposition berechnen <H>:Zeile, <L>:Spalte

0CCE:  45          LD    B,L        <B> gleich Spalte
0CCF:  6C          LD    L,H        <L> ist nun Zeile
0CD0:  26 00       LD    H,$00      lösche Hi-Byte
0CD2:  29          ADD   HL,HL      *2
0CD3:  29          ADD   HL,HL      *2
0CD4:  29          ADD   HL,HL      *2
0CD5:  29          ADD   HL,HL      *2=*16
0CD6:  54          LD    D,H        <DE> wird nun mit
0CD7:  5D          LD    E,L        Zeile mal 16 belegt
0CD8:  29          ADD   HL,HL      *2
0CD9:  29          ADD   HL,HL      *2 ergibt *64
0CDA:  19          ADD   HL,DE      plus *16 ergibt *80
0CDB:  EB          EX    DE,HL      Zeile mal 80 nach <DE>
0CDC:  68          LD    L,B        Spalte nach <L>
0CDD:  26 00       LD    H,$00      Hi-Byte löschen
0CDF:  19          ADD   HL,DE      und Zeile*80 addieren
0CE0:  06 00       LD    B,$00      Lösche Hi-Byte von <BC>
```

```
0CE2:  3C           INC   A           <A>:=reale Anzahl der möglichen Zeichen
0CE3:  C9           RET               Ende der Routine

****************************************  ASCII-Dekodierung Cont'd

0CE4:  78           LD    A,B         Zeichen nach <Akku>
0CE5:  2A 0D FD     LD    HL,($FD0D)  Tabellenzeiger
0CE8:  D6 30        SUB   $30         Minus ASCII 48 "0"
0CEA:  D8           RET   C           Kleiner, dann Ende
0CEB:  B9           CP    C           Sonst vergleiche mit <C> (abgezogener Wert)
0CEC:  3F           CCF               Carry-Flag negieren
0CED:  D8           RET   C           Und bei größer/gleich Ende
0CEE:  47           LD    B,A         Sonst Zeichen nach <B>
0CEF:  E6 0F        AND   $0F         Bits 0-3 ausmaskieren
0CF1:  5F           LD    E,A         Lo-Byte gleich <Akku>
0CF2:  16 00        LD    D,$00       und Hi-Byte löschen
0CF4:  19           ADD   HL,DE       zur Tabellenbasis addieren
0CF5:  78           LD    A,B         Zeichen wieder nach <Akku>
0CF6:  E6 30        AND   $30         Bits 6,7 und 0-3 ausmaskieren
0CF8:  47           LD    B,A         Zeichen wieder nach <B>
0CF9:  C9           RET               Ende der Routine

**************************************** Ton erklingen lassen

0CFA:  01 18 D4     LD    BC,$0418    SID Register 24
0CFD:  2A 10 FD     LD    HL,($FD10)  Hole Attack/Decay/Volume
0D00:  ED 61        OUT   (C),H       Gesamtlautstärke/Filter
0D02:  0E 05        LD    C,$05       Register 5 des SID: Attack/Decay
0D04:  ED 69        OUT   (C),L       definieren
0D06:  2A 12 FD     LD    HL,($FD12)  Hole Sustain/Release/Frequenz
0D09:  0C           INC   C           Register 6 des SID
0D0A:  ED 61        OUT   (C),H       Sustain/Release definieren
0D0C:  0E 01        LD    C,$01       Register 1 des SID: Frequenz
0D0E:  ED 69        OUT   (C),L       Frequenz (HI) definieren
0D10:  2A 14 FD     LD    HL,($FD14)  Hole Ein/Ausschelten
0D13:  0E 04        LD    C,$04       Register 4 des SID
```

```
0015:  ED 61          OUT   (C),H       Einschalten des Tones
0D17:  ED 69          OUT   (C),L       Bit zum $ustain löschen
0019:  C9             RET               Ende der Tonerklingungsroutine

****************************************** Wird nach $1100 kopiert

001A:  1100: A9 00        LDA #$00      Konfigurationsbyte
001C:  1102: 8D 00 FF     STA $FF00     setzen (alles ROM)
001F:  1105: 6C FC FF     JMP ($FFFC)   Reset des C-128-Modus

****************************************** Wird nach $3000 kopiert
                                           Lesen eines Diskettenblockes

0D22: 3000: A9 00         LDA #$00      Noch ist kein
0024: 3002: 80 06 FD      STA $FD06     Fehler aufgetreten
0D27: 3005: 20 11 30      JSR $3011     Laden des Blockes
0D2A: 3008: 78            SEI           Interrupt verhindern
002B: 3009: A9 3E         LDA #$3E      RAM und System I/O
0D2D: 300B: 8D 00 FF      STA $FF00     als Konfigurationsbyte
0D30: 300E: 4C D0 FF      JMP $FFD0     und Z-80 einschalten

0D33: 3011: D8            CLD           Lösche Dezimalflag
0D34: 3012: AD 01 FD      LDA $FD01     Flag für Vektoren setzen
0D37: 3015: D0 21         BNE $3038     gelöscht?
0D39: 3017: A2 00         LDX #$00      ROM und System I/O
0D3B: 3019: 8E 00 FF      STX $FF00     einschalten
0D3E: 301C: 8E 1A D0      STX $D01A     IMR im VIC-Chip löschen
0D41: 301F: A2 63         LDX #$63      Lo-Byte der anzuspringenden Routine
0D43: 3021: A0 31         LDY #$31      und Hi-Byte derselben
0D44: 3023: 8E 14 03      STX $0314     Lo-Byte als IRQ-Routine
0D48: 3026: 8C 15 03      STY $0315     Hi-Byte als IRQ-Routine
0D4B: 3029: 8E 16 03      STX $0316     Lo-Byte als BRK-Routine
004E: 302C: 8C 17 03      STY $0317     Hi-Byte als BRK-Routine
0D51: 302F: 8E 18 03      STX $0318     Lo-Byte als NMI-Routine
0054: 3032: 8C 19 03      STY $0319     Hi-Byte als NMI-Routine
0057: 3035: 4C D7 30      JMP $3007     Weitermachen bei $3007
```

```
**************************************** Lesen von Block (Spur/Sektor) & Ablage im Speicher

0D5A: 3038: AD 18 FD      LDA $FD18      Lo-Byte Zieladresse
0D5D: 303B: 85 20         STA $20        nach $20
005F: 303D: AD 19 FD      LDA $FD19      Hi-Byte Zieladresse
0D62: 3040: 85 21         STA $21        nach $21 kopieren
0064: 3042: AD 03 FD      LDA $FD03      Hole Spur
0D67: 3045: 8D BF 31      STA $318F      und in Diskettenkommando kopieren
0D6A: 3048: 20 78 31      JSR $3178      macht aus <Akku> ASCII "xx"
0D6D: 304B: 8E B1 31      STX $31B1      Zehnerstelle Track und
0D70: 304E: 8D B0 31      STA $3180      Einerstelle Track in Kommandozeile
0D73: 3051: AD 04 FD      LDA $FD04      Hole Sektornummer
0D76: 3054: 8D BE 31      STA $31BE      Sektornummer für FSD übergeben
0079: 3057: 20 78 31      JSR $3178      und in ASCII wandeln
0D7C: 305A: 8E AE 31      STX $31AE      Zehnerstelle Sektor sowie
0D7F: 305D: 8D AD 31      STA $31AD      Einerstelle in Kommandozeile
0082: 3060: AD 08 FD      LDA $FD08      Teste, ob Kanal eröffnet wurde
0D85: 3063: D0 2B         BNE $3090      Nein, dann Sprung nach $3090
0087: 3065: 8D 00 FF      STA $FF00      Konfigurationsbyte auf 0F (ROM) setzen
008A: 3068: A2 0B         LDX #$0B       Kanal 11 (#) als Eingabe-
008C: 306A: 20 C6 FF      JSR $FFC6      kanal definieren
0D8F: 3060: B0 16         BCS $30A7      Bei Fehler nach $30A7
0091: 306F: 20 CC FF      JSR $FFCC      CLRCH; Ein/Ausgabe wieder normal
0094: 3072: 20 31 31      JSR $3131      Track/Sektor lesen
0097: 3075: 20 99 31      JSR $3199      Kanal 11 (#) als Eingabekanal
0D9A: 3078: A0 00         LDY #$00       Y-Index auf null
0D9C: 307A: 20 CF FF      JSR $FFCF      BASIN; Zeichen von Floppy holen
0D9F: 307D: 91 20         STA (S20),Y    und Zeichen im RAM ablegen
00A1: 307F: C8            INY            nächstes Byte
00A2: 3080: D0 F8         BNE $307A      Ende noch nicht erreicht
00A4: 3082: 4C CC FF      JMP $FFCC      CLRCH; Ein/Ausgabe wieder normal

0DA7: 3085: A9 FF         LDA #$FF       Block konnte nicht geholt werden
0DA9: 3087: 2C            .Byte $2C      Skip; Überspringe folgendes Kommando
0DAA: 3088: A9 0D         LDA #$00       Kennzeichen für Fehler
```

```
0DAC: 308A: 8D 06 FD      STA SFD06     aufgetreten setzen
0DAF: 308D: 4C 08 30      JMP $3008     wieder in Z-80-Teil

************************************* Daten von Floppy lesen

0DB2: 3090: A9 00         LDA #$00      ROM und System I/O als
0DB4: 3092: 8D 00 FF      STA SFF00     Konfigurationsbyte setzen
0DB7: 3095: A2 0F         LDX #$0F      Logische Filenummer 15
0DB9: 3097: 20 C9 FF      JSR $FFC9     Befehlskanal als Ausgabekanal
0DBC: 309A: B0 EC         BCS $3088     bei Fehler anzuspringen
0DBE: 309C: A0 06         LDY #$06      Sechs Zeichen auszugeben
0DC0: 309E: B9 BC 31      LDA $31BC,Y   Hole Zeichen aus Tabelle
0DC3: 30A1: 20 D2 FF      JSR $FFD2     Ausgabe des Zeichens auf Befehlskanal
0DC6: 30A4: 88            DEY           Nächstes Zeichen
0DC7: 30A5: D0 F7         BNE $309E     Noch weitere Zeichen auszugeben
0DC9: 30A7: 20 CC FF      JSR $FFCC     CLRCH; Ein/Ausgabe wieder normal
0DCC: 30AA: 2C 0D DC      BIT SOCOD     Teste ICR
0DCF: 30AD: AE BD 31      LDX $31BD     Anzahl der Datenblöcke
0DD2: 30B0: 20 4F 31      JSR $314F     Hole Datenbyte (Fast-Modus)
0DD5: 30B3: 29 0E         AND #$0E      Teste Bits 1-3
0DD7: 30B5: D0 01         BNE $3088     Fehler aufgetreten
0DD9: 30B7: A0 00         LDY #$00      Index auf Null
0DDB: 30B9: 20 4F 31      JSR S314F     Hole Datenbyte (Fast-Modus)
0DDE: 30BC: 91 20         STA ($20),Y   Lege Zeichen im RAM ab
0DE0: 30BE: C8            INY           erhöhe den Zeiger
0DE1: 30BF: D0 F8         BNE $3089     Noch nicht alle Zeichen
0DE3: 30C1: E6 21         INC $21       Erhöhe Hi-Byte des Pointers
0DE5: 30C3: CA            DEX           Erniedrige den Blockzähler
0DE6: 30C4: D0 EA         BNE $3080     Weitere Blocks
0DE8: 30C6: AD 00 DD      LDA SDD00     PRA CIA2 holen
0DEB: 30C9: 29 EF         AND #$EF      CLK-Bit ausmaskieren
0DED: 30CB: 80 00 DD      STA $DD00     und wieder zurück
0DF0: 30CE: 60            RTS           Ende der Routine

0DF1: 30CF: A9 0F         LDA #$0F      Logische Filenummer 15
0DF3: 30D1: 80 06 FD      STA SFD06     in $FD06 ablegen
```

```
00F6: 3004: 4C 08 30     JMP $3008     Z-80 einschalten

00F9: 3007: A9 0F        LDA #$0F      Logische Filenummer
0DFB: 30D9: 18           CLC           Lösche Carry als Flag
0DFC: 30DA: 20 C3 FF     JSR $FFC3     Schließen des Kanals
0DFF: 30DD: A9 0F        LDA #$0F      Logische Filenummer
0E01: 30DF: 80 08 FD     STA $FD08     Logische Filenummer merken
0E03: 30E2: A2 08        LDX #$08      Geräteadresse
0E06: 30E4: A8           TAY           15 als Sekundäradresse
0E07: 30E5: 20 BA FF     JSR $FFBA     SETLFS; Setzen der log. Fileparameter
0E0A: 30E8: A9 00        LDA #$00      Schnellmodus der
0E0C: 30EA: 80 1C 0A     STA $0A1C     Floppy ausschalten
0E0F: 30ED: AA           TAX           Beide Konfigurationsindize
0E10: 30EE: 20 68 FF     JSR $FF68     für SETBNK auf null setzen
0E13: 30F1: A9 04        LDA #$04      Länge Filename
0E15: 30F3: A2 B8        LDX #$B8      Adresse Lo des Filenamens
0E17: 30F5: A0 31        LDY #$31      Adresse Hi des Filenamens
0E19: 30F7: 20 BD FF     JSR $FFBD     SETNAM; Filenamenparameter setzen
0E1C: 30FA: 20 C0 FF     JSR $FFC0     OPEN der Datei
0E1F: 30FD: B0 D0        BCS S30CF     Fehler aufgetreten, dann Sprung
0E21: 30FF: 20 B7 FF     JSR SFFB7     Statusbyte I/O holen
0E24: 3102: 2A           ROL A         Bit 7 ins Carry shiften
0E25: 3103: B0 CA        BCS $30CF     Fehler aufgetreten, dann Sprung
0E27: 3105: 2C 1C 0A     BIT $0A1C     Teste das Fast-Serial-Bit
0E2A: 3108: 70 26        BVS $3130     ist gesetzt, dann Sprung
0E2C: 310A: A9 0B        LDA #$0B      Logische Filenummer 11
0E2E: 310C: 18           CLC           Lösche Carry als Flag
0E2F: 310D: 20 C3 FF     JSR $FFC3     Schließe Datei 11
0E32: 3110: A9 0B        LDA #$0B      Logische Filenummer 11
0E34: 3112: A2 08        LDX #$08      Gerätenummer 8
0E36: 3114: A0 08        LDY #$08      Sekundäradresse 8
0E38: 3116: 20 BA FF     JSR $FFBA     SETLFS; Filedaten speichern
0E3B: 3119: A9 00        LDA #$00      LFN als nicht geöffnet
0E3D: 311B: 80 08 FD     STA $FD08     an $FD08 löschen
0E40: 311E: AA           TAX           Beide Konfigurationsindizes
0E41: 311F: 20 68 FF     JSR $FF68     sind null; SETBNK
```

```
0E44: 3122: A9 01       LOA #$01     Länge des Filenamens
0E46: 3124: A2 BC       LDX #$BC     Lo-Byte des Filenamens
0E48: 3126: A0 31       LOY #$31     Hi-Byte des Filenamens
0E4A: 3128: 20 BD FF    JSR $FFBD    SETNAM; Adresse Filename setzen
0E4D: 312B: 20 C0 FF    JSR $FFC0    OPEN 11,8,8,"#"; öffnen der Datei
0E50: 312E: B0 9F       BCS $30CF    Fehler aufgetreten
0E52: 3130: 60          RTS          sonst Ende der Routine

****************************************** Track/Sektor lesen

0E53: 3131: 20 A4 31    JSR $31A4    Befehlskanal auf Ausgabe legen
0E56: 3134: A0 0D       LDY #$0D     13 Zeichen sind auszugeben "U1:8 0 tt ss<CR>"
0E58: 3136: B9 AB 31    LDA $31AB,Y  Hole Zeichen aus der Tabelle
0E5B: 3139: 20 D2 FF    JSR $FFD2    und ausgeben
0E5E: 313C: 88          DEY          Zähler erniedrigen
0E5F: 3130: D0 F7       BNE $3136    und Sprung, wenn weitere Zeichen
0E61: 313F: 20 CC FF    JSR $FFCC    CLRCH; Ein/Ausgabe wieder normal
0E64: 3142: 20 89 31    JSR $3189    Befehlskanal zum Lesen Offnen
0E67: 3145: F0 05       BEQ $314C    kein Fehler aufgetreten
0E69: 3147: A9 0D       LDA #$0D     Merker für Fehler
0E6B: 3149: 80 06 FD    STA $FD06    setzen
0E6E: 314C: 4C CC FF    JMP $FFCC    CLRCH; Ein/Ausgabe wieder normal

****************************************** Warte, bis SDR (Serial Data Register) fertig und
                                          Hole dann Datenbyte

0E71: 314F: 78          SEI          Interrupt verhindern
0E72: 3150: AD 00 DD    LDA #$0000   Hole PRA CIA2
0E75: 3153: 49 10       EOR #$10     Clock-Leitung negieren
0E77: 3155: 8D 00 DD    STA $0000    Negiert zurückliefern
0E7A: 3158: A9 08       LDA #$08     SDR voll/leer-Bit abtesten
0E7C: 315A: 2C 0D DC    BIT $DC0D    SDR ist noch nicht fertig
0E7F: 3150: F0 FB       BEQ $315A    Warte, bis Timer Unterlauf hat
0E81: 315F: AD 0C DC    LDA $DC0C    Hole SDR Serial Data Register
0E84: 3162: 60          RTS          Ende der Routine
```

```
0E85: 3163: A0 0D DC     LDA SDC0D     Lösche Interruptregister im CIA1
0E88: 3166: A0 0D DD     LDA SDD0D     Lösche Interruptregister im CIA2
0E88: 3169: A9 0F        LDA #$0F      Lösche Interruptregister
0E80: 316B: 80 19 D0     STA $0019     des VIC-Chip
0E90: 316E: 68           PLA           Konfiguration
0E91: 316F: 80 00 FF     STA $FF00     wiederherstellen
0E94: 3172: 68           PLA           Y-Register
0E95: 3173: A8           TAY           wiederherstellen
0E96: 3174: 68           PLA           X-Register
0E97: 3175: AA           TAX           wiederherstellen
0E98: 3176: 68           PLA           Akku wiederherstellen
0E99: 3177: 40           RTI           Ende der Interrupt-Routine

****************************************** macht aus <Akku> zwei ASCII-Codes

0E9A: 3178: D8           CLD           Lösche Dezimalflag
0E9B: 3179: A2 30        LDX #$30      Zehnerstelle ist zunächst einmal "0"
0E9D: 317B: 38           SEC           Setze Carry-Flag für Subtraktion
0E9E: 317C: E9 0A        SBC #$0A      Zehn subtrahieren (probeweise)
0EA0: 317E: 90 03        BCC $3183     Zuviel subtrahiert
0EA2: 3180: E8           INX           sonst Zehnerstelle erhöhen
0EA3: 3181: B0 F9        BCS $317C     und weitermachen
0EA5: 3183: 69 3A        ADC #$3A      Fehler korrigieren und ASCII-Basis addieren
0EA7: 3185: 60           RTS           Ende der Routine

****************************************** Testet, ob auf Befehlskanal Fehler vorliegt

0EA8: 3186: 20 D7 30     JSR $3007     Fehler bei Kanal 15 aufgetreten
0EAB: 3189: A2 0F        LDX #$0F      Befehlskanal zum
0EAD: 318B: 20 C6 FF     JSR $FFC6     Lesen öffnen; CHKIN
0EB0: 318E: B0 F6        BCS $3186     Fehler aufgetreten
0EB2: 3190: 20 CF FF     JSR $FFCF     BASIN; Zeichen holen
0EB5: 3193: C9 30        CMP #$30      Akku mit "0" vergleichen (dann OK)
0EB7: 3195: 60           RTS           Ende der Routine

****************************************** Kanal 11 (#) zum Lesen vorbereiten
```

```
0EB8: 3196: 20 0A 31      JSR $310A     Fehler bei Kanal 11 aufgetreten
0EBB: 3199: A2 0B         LDX #$0B      Kanal 11 auf Eingabe
0EBD: 319B: 20 C6 FF      JSR $FFC6     legen; CHKIN
0EC0: 319E: B0 F6         BCS $3196     Fehler aufgetreten
0EC2: 31A0: 60            RTS           Ende der Routine

****************************************** Kanal 15 (Befehlskanal) auf Ausgabe

0EC3: 31A1: 20 D7 30      JSR $3007     Fehler bei Kanal 15 aufgetreten
0EC6: 31A4: A2 0F         LDX #$0F      Kanal 15 als Ausgabekanal
0EC8: 31A6: 20 C9 FF      JSR $FFC9     durch CKOUT-Routine definieren
0ECB: 31A9: B0 F6         BCS $31A1     Fehler aufgetreten
0ECD: 31AB: 60            RTS           Ende der Routine

0ECE: 31AC: 0D 73 73 20 74 74 20 30     .ss tt 0
0ED6: 31B4: 20 38 3A 31 55 30 4C 00      8:1U0L.
0EDE: 31BC: 23 01 00 00 00 30 55        #....0U

****************************************** Wird nach $FFD0 kopiert (8502-Code)

0EE5: ($FFD0) 78          SEI           Interrupt verhindern
0EE6: ($FFD1) A9 3E       LDA #$3E      Konfigurationsbyte auf $3E setzen
0EE8: ($FFD3) 8D 00 FF    STA $FF00     RAM und System I/O
0EEB: ($FFD6) A9 80       LDA #$80      Z-80 einschalten
0EED: ($FFD8) 8D 05 D5    STA $D505     im MCR
0EF0: ($FFDB) EA          NOP           Wartewirkung
0EF1: ($FFDC) 4C 00 30    JMP $3000     Sprung in nächsten Routinenteil
0EF4: ($FFDF) EA          NOP           Buffer

****************************************** Wird nach $FFE0 kopiert

0EF5: ($FFE0) F3          DI                  Interrupts unterbinden
0EF6: ($FFE1) 3E 3E       LD    A,$3E         Konfigurationsbyte
0EF8: ($FFE3) 32 00 FF    LD    ($FF00),A     setzen
0EFB: ($FFE6) 01 05 D5    LD    BC,$D505      Mode-Config.-Register
```

```
0EFE: (SFFE9) 3E B1        LD    A,$B1     und 8502 einschalten
0F00: (SFFEB) ED 79        OUT   (C),A     durch Bit-0-setzen
0F02: (SFFED) 00           NOP             Bufferwirkung
0F03: (SFFEE) CF           RST   $08       Sprung in Z-80-Teil

****************************************** Allerlei Tabellen

0F04:  9E FF BD 58 6F CE/00 0F          ab / angepaßte Farbtabelle
0F0C:  08 07 0B 04 02 00 0A 0C          für 40-Zeichen-Schirm
0F14:  09 06 01 05 03 0E/00 60
0F1C:  30 18 0C 06 03 00 18 3C
0F24:  66 00 00 00 00 00 00 00
0F2C:  00 00 00 00 7F 00 60 30
0F34:  18 00 00 00 00 00 1C 30
0F3C:  30 60 30 30 1C 00 18 18
0F44:  18 18 18 18 18 00 38 0C
0F4C:  0C 06 0C 0C 38 00 00 18
0F54:  2A 66 00 00 00 00 00 00
0F5D:  00 00 00 41 7F 00 00 F2
0F64:  5B 39 01 4E 65 37 06 03
0F6C:  1E 07 0B 68 4B 34 17 01
0F74:  44 62 2D 18 12 0B 63 59
0F7C:  31 17 00 0B 59 72 2B 18
0F84:  0F 63 00 4F 2B 05 4C 68
0F8C:  2D 17 16 69 49 25 17 13
0F94:  45 68 29 18 17 07 0C 68
0F9C:  48 34 13 0F 05 48 70 31
0FA4:  31 0D 0D 08 08 6C/00 3F          ab / MMU-Registerbelegungen
0FAC:  7F 3E 7E B0 0B 00 00 01
0FB4:  00/00 05 0A 0F 14 04 09          angepaßte Sektornummern nach
0FBC:  0E 13 03 08 0D 12 02 07          verschiedenen Sektorgrößenbezirken
0FC4:  0C 11 01 06 0B 10 00 05          aufgeteilt.
0FCC:  0A 0F 01 06 0B 10 02 07          Durch dieses Verfahren wird der
0FD4:  0C 11 03 08 0D 12 04 09          Zugriff auf Diskette zeitlich
0FDC:  0E 00 05 0A 0F 02 07 0C          noch optimiert
0FE4:  11 04 09 0E 01 06 0B 10
```

```
0FEC:  03 08 0D 00 05 0A 0F 03
0FF4:  08 0D 01 06 0B 10 04 09
0FFC:  0E 02 07 0C
```

X.1 COPYSYS

COPYSYS.COM

Kopiert normalerweise die Systemspuren und CP/M3.SYS auf eine Diskette

Eingabeformat:

COPYSYS

Beschreibung:

Um von einer Diskette CP/M 3.0 zu starten, müssen beide CP/M-Teile vorhanden sein: Der erste Teil auf den Systemspuren, der zweite als Datei CPM+.SYS. Disketten, die nicht zum Booten verwendet werden, brauchen beides nicht. Beim Commodore 128 ist die eigentliche Form des COPYSYS-Kommandos allerdings leicht abgeändert, Sie können ja das System durch einfaches Kopieren mittels PIP übertragen. Sollten Sie dennoch - etwa aus Gewohnheit - das Kommando COPYSYS aufrufen, so wird Ihnen auf dem Bildschirm angezeigt, daß es dieses Kommando praktisch nicht mehr gibt.

X.2 DATE

DATE.COM

```
Zeigt Datum und Zeit und ermöglicht deren Eingabe
```

Eingabeformat:

DATE
DATE CONTINUOUS
DATE SET
DATE MM/DD/YY HH:MM:SS

Beschreibung:

Mit diesem Programm können Sie ein Datum und eine Uhrzeit in Ihr System eingeben oder beide Informationen abrufen.

Anwendung:

Geben Sie nur den Befehl DATE ein, bekommen Sie das Datum und die Uhrzeit angezeigt

Tue 05/06/85 23:49:17

Dies ist eine stehende Anzeige. Möchten Sie die Zeit mit einer anderen Uhr vergleichen, geben Sie den Befehl DATE mit der Option "C" ein. Dann sehen Sie die laufende Uhr, können aber in dieser Zeit kein anderes Programm laufen lassen oder irgendwelche Befehle eingeben. Drücken Sie eine beliebige Taste, um das Programm abzubrechen.

Sie können Datum und Zeit auf zwei Wegen eingeben: einmal mit der Option SET, dann erfragt das Programm die Werte hintereinander und Sie können eine Eingabe mit ENTER überspringen. Die andere Möglichkeit sieht so aus:

DATE 05/06/85 23:49:17

Geben Sie eine Zeit ein, die etwa ein bis zwei Minuten hinter der aktuellen liegt. Wenn der von Ihnen eingegebene Zeitpunkt kommt, drücken Sie nur noch die Leertaste, und die Uhr läuft.

Besitzt Ihr Computer keine batteriegepufferte Uhr, und das ist beim Commodore 128 der Fall, so müssen Sie die Eingabe jedesmal nach dem Starten eingeben. Sie können sich das erleichtern, indem Sie den Befehl DATE SET in die Datei PROFILE.SUB schreiben.

Durch Eingabe von DATEC können Sie sich die Eingabe des DATE-Komamndos mit der C-Option ersparen. Interessant vielleicht noch die Tatsache, daß sich die Quelldatei für das DATE-Programm auf der Systemdiskette unter dem Namen DATE.ASM befindet. Anschauen ist hier sicherlich sehr lehrreich.

X.3 DEVICE

DEVICE.COM

```
Zeigt und verändert die Wege zu den Peripherie-Geräten
```

Eingabeformat:

DEVICE NAMES
DEVICE VALUES
DEVICE LST:=XXXXXX [NOXON,1200]
DEVICE CONOUT:=XXX,XXX
DEVICE CON:[PAGES]
DEVICE CON:[COLUMNS=nn, LINES=mm]

Beschreibung:

Dieses Programm wird benutzt, um die Ein- und Ausgabekanäle des Betriebssystems anzuzeigen und zu verändern. Die drei logischen Kanäle heißen: CON:, LST: und AUX:. Man kann auch einen logischen Kanal auf mehrere Peripheriegeräte lenken. Zum Beispiel können die Daten aus dem Computer sowohl auf den Drucker, wie auch gleichzeitig auf den Bildschirm geschickt werden. Mit DEVICE können Sie auch die Zahl der Zeilen und Spalten auf dem Bildschirm bestimmen.

Anwendung:

Die Namen der Ausgabeeinheiten werden von den Herstellern festgelegt. Geben Sie DEVICE mit der Option NAMES ein, bekommen Sie eine Liste mit den Namen und die Übertragungsraten der einzelnen Geräte. Geben Sie DEVICE VALUES ein, bekommen Sie etwa so eine Anzeige:

Current Assignments:

CONIN: = CRT
CONOUT: = CRT
AUXIN: = LPT
AUXOUT: = LPT
LST: = CEN

CONIN: und CONOUT: bedeuten CONsol INput, Tastatur-Eingabe und CONsole OUTput, Tastatur-Ausgabe. Steht die Bezeichnung CON: allein, bezieht sie sich auf beides. LST: bedeutet LIST DEVICE und spricht den Drucker an.

Mit der PAGE-Option können Sie feststellen, wieviele Zeilen und Spalten zur Zeit auf dem Bildschirm angezeigt werden. Geben Sie in der Form [nnn] eine bestimmte Baudrate ein, wird der Datenverkehr mit dem dazugehörenden Gerät mit dieser Geschwindigkeit abgewickelt. Auf den Befehl DEVICE hin, bekommen Sie die Anzeige von DEVICE NAMES und VALUES zusammen angezeigt.

Beachten Sie, daß das Kommando DEVICE unter CP/M 2.0 durch das Kommando STAT DEV: ersetzt wird (richtiger müßte man sagen, daß man das Kommando STAT unter CP/M 2.2 gesplittet hat, da es zu komplex ist.)

X.4 DIR

DIR.COM (transient auf Diskette) und resident

```
Zeigt  bestimmte Dateinamen und das Inhaltsverzeichnis
```

Eingabeformat:

Eingebauter Befehl:

DIR
DIR A:
DIR DATEI.XXX
DIR AB*.*

Transienter Befehl:

DIR[OPTION]
DIR DATEI.XXX[OPTION]
DIR AB*.*[OPTION]

Beschreibung:

DIR ist ein sehr hilfreiches Programm, um Dateien auf einer Diskette oder erst recht auf einer Festplatte wiederzufinden. (Noch ist kein Anbieter für Festplatten für einen der Schneider-Rechner bekannt - leider) Die eingebaute Version arbeitet von jedem Laufwerk und aus jedem Benutzerbereich. Die "Jokerzeichen" "*" und "?" können benutzt werden. Wird keine Option mit eingegeben, werden alle Nicht-System-Dateien des entsprechenden Benutzerbereiches angezeigt.

DIR als transienter Befehl kann erheblich mehr: Dateien mit Datum-Stempel anzeigen, in alphabetischer Reihenfolge sortieren usw. Eine Aufzählung der verschiedenen Optionen,

die in eckigen Klammern eingegeben werden müssen, folgt jetzt. Stehen auch System-Dateien im Inhaltsverzeichnis, erscheint die Meldung:

SYSTEM FILE(S) EXIST

OPTIONEN:

Option	Funktion
ATT	Gib benutzerdefinierte Datei-Attribute aus
DATE	Zeige Dateien mit Zeit und Datum
DIR	Zeige Nicht-System-Dateien
DRIVE=ALL	Zeige Dateien aller Laufwerke
DRIVE=A	Zeige Dateien auf Laufwerk A:
DRIVE=(A,B)	Zeige Dateien in den angegebenen Laufwerken
EXCLUDE	Zeige alle Dateien, außer dem eingegebenen
FF	Seitenvorschub vor Ausgabe des Verzeichnisses
FÜLL	Zeige Dateien mit vollständiger Beschreibung
LENGTH=n	Überschrift nach n Zeilen
MESSAGE	Zeige Laufwerke und USER-Bereiche fortlaufend
NOPAGE	Zeige Inhaltsverzeichnis ohne Stopp
NOSORT	Zeige Dateien unsortiert
RO	Zeige Read Only Dateien
RW	Zeige Read/Write Dateien
SIZE	Größe jeder Datei mit angeben
SYS	Zeige nur System-Dateien
USER=ALL	Zeige Dateien aller USER-Bereiche
USER=5	Zeige Dateien des USER-Bereiches
USER=(3,5)	Zeige Dateien der USER-Bereiche

X.5 DIRSYS

Eingebauter Befehl

```
Zeigt die Dateien an, die zu SYSTEM-Dateien erklärt wurden
```

Eingabeformat:

DIRSYS
DIRSYS B:
DIRSYS DATEI.XXX
DIRSYS AB?*.*

Beschreibung:

Mit DIRSYS werden alle SYSTEM-Dateien angezeigt. Dies kann zwar auch mit DIR und Optionen erreicht werden, aber DIRSYS ist schneller, da es ein eingebauter Befehl ist. Sie können den Befehl zu:

DIRS

abkürzen. Ferner erhalten Sie beim DIRSYS-Kommando noch Auskunft darüber, ob es neben den System-Dateien noch "normale", CP/M nennt diese Dateien Nicht-System-Dateien existieren. Existieren keine System-Dateien, so erscheint auf dem Monitor die Meldung:

```
No File
```

X.6 DUMP

DUMP.COM

Zeigt Nicht-Text-Dateien in hexadezimal und ASCII

Eingabeformat:

DUMP
DUMP DATEI.XXX

Beschreibung:

DUMP ist besonders für Programmierer interessant, die mit Assembler arbeiten. Textdateien werden mit TYPE auf den Bildschirm oder Drucker gebracht, Dateien in der Art von COM, REL oder OVR mit DUMP. Dabei werden auf dem Bildschirm die Hexadezimalwerte und, sofern möglich, die entsprechenden ASCII-Werte dargestellt. Ein nicht darstellbares Zeichen wird durch einen Punkt (.) auf dem Bildschirm angezeigt. Hat man den Drucker durch <CTRL>-P aktiviert, so wird die Ausgabe selbstverständlich auch auf den Drucker umgeleitet.

Unter Kapitel VIII. finden Sie detaillierte Beschreibungen zu den COM-Dateien MAC, SID und DUMP.

X.7 ED

ED.COM

Unter CP/M integrierter Editor

Eingabeformat:

ED <Dateiname>(.<Dateikennung>)

Beschreibung: Der eingebaute Editor ED ist für sehr kurze Texte vielleicht geeignet, allerdings hat er einen recht schlechten Ruf, und das hat seinen Grund. Komplizierte Bedienung (selbst für "Profis") tut hier seinen Teil dazu. Es ist mir kaum jemand bekannt, der ED freiwillig benutzen würde.

Bei Eingabe eines Dateinamen wird von ED übberprüft, ob diese Datei bereits existiert. Ist dies der Fall, so wird der Dateiinhalt in den ED-Buffer geladen und eine Backup-Datei erstellt. Sollte die Datei noch nicht existieren, so wird dies von ED erkannt, und es erscheint die Meldung

NEW FILE

auf dem Bildschirm. Sie verlassen den Editor inklusive Abspeichern der Datei, indem Sie das Kommando "E" eingeben.

X.8 ERA(SE)

Eingebauter Befehl und ERASE.COM

```
Löscht eine, mehrere oder alle Dateien von der Diskette
```

Eingabeformat:

Eingebauter Befehl:

ERASE
ERASE DATEI.XXX
ERASE AB?*.*

Transienter Befehl:

ERASE DATEI.XXX[CONFIRM]

Beschreibung:

Auf den Befehl ERASE hin, wird die angegebene Datei gelöscht, wenn sie nicht "Read Only" oder schreibgeschützt ist. Löschen können Sie nur innerhalb des USER-Bereiches. Benutzen Sie "*" oder "?", wiederholt der Befehl die Löschanweisung, und fragt nach, ob wirklich gelöscht werden soll.

Anwendung:

Ohne Optionen eingegeben, kann ERASE von jedem Laufwerk aus arbeiten. Bei Eingaben mit Optionen, ist das Programm ERASE.COM auf dem Laufwerk notwendig. ERASE kann zu ERA abgekürzt werden. Die Option "CONFIRM" kann zu "C" abgekürzt werden.

X.9 FORMAT

FORMAT.COM

Formatiert ein Diskette neu

Eingabeformat:

FORMAT

Beschreibung:

Jede Diskette, die Sie in Ihrem Computer verwenden, muß entsprechend formatiert sein. Das Programm zerstört beim Formatieren alle Daten auf einer Diskette. FORMAT ist kein Standard-CP/M-Programm. Deshalb kann es ein, daß es auf Ihrer System-Diskette nicht vorhanden ist. Suchen Sie dann nach dem Programm COPY.COM und sehen Sie nach, ob Sie damit eine Diskette formatieren können.

Anwendung:

FORMAT arbeitet solange, bis Sie es mit Control C (^C) abbrechen oder bis Sie auf die Frage Format another disk mit N antworten. So können Sie mehrere Disketten hintereinander formatieren.

Anmerkung:

Beim Commodore 128 sind verschiedene Disketten-Formate möglich. Entsprechend bekommen Sie beim Formatieren auch angeboten, ob Sie einseitig oder zweiseitig formatieren etc. Sie können beim FORMAT-Kommando ausschließlich Disketten im A-Laufwerk formatieren.

X.10 GENCOM

GENCOM.COM

Erstellt eine spezielle Version von CP/M 3.0

Eingabeformat:

GENCOM

Beschreibung:

Wird von Programmierern gebraucht, um eine CP/M-Version anzupassen oder zusätzliche Programmteile in CP/M einzuflechten.

X.11 GET

GET.COM

```
Holt Daten aus einer Datei, statt von der Tastatur
```

Eingabeformat:

GET CONSOLE INPUT FROM FILE DATEI.XXX
GET CONSOLE INPUT FROM CONSOLE
GET CONSOLE INPUT FROM FILE DATEI.XXX[SYSTEM]
GET FILE DATEI.XXX[NOECHO]

Beschreibung:

Mit dem GET-Befehl können Sie Eingaben oder Befehle statt über die Tastatur einzugeben, aus einer beliebigen Datei kommen lassen.

Anwendung:

Die erste Zeile oben befiehlt CP/M, den nächsten Befehl oder das nächste Programm zu verarbeiten, der oder das über die Tastatur eingegeben wird. Sobald eine Eingabe nötig wird, holt sich CP/M diese aus der Datei DATEI.XXX. Ist das Programm zu Ende oder kein weiterer Befehl mehr in der Datei, kehrt CP/M wieder zur Tastatur-Eingabe zurück.

Der Befehl in der zweiten Zeile weist CP/M an, wieder Befehle von der Tastatur entgegenzunehmen. Dieser Befehl kann in der Datei DATEI.XXX stehen oder vom Benutzer eingegeben werden.

Die dritte Form des Befehls leitet die Tastatur-Eingabe sofort zu der benannten Datei um, damit diese die Tastatur-Befehle lesen und verarbeiten kann. Geben Sie als Option (NOECHO) mit an, werden die gerade ausgeführten Befehle nicht auf dem Bildschirm angezeigt.

Dieser Befehl ähnelt dem SUBMIT-Kommando, bei dem ja auch Befehle praktisch von einer Datei geordert werden können. Für geschickte Programmierer stehen hier eine Menge Möglichkeiten offen. Commodore selbst beispielsweise hat durch ein GET-Kommando eine Flut von Kommandos auf einer Update-Diskette in Gang gebracht, die einen Fehler in einer alten CCP.COM-Datei korrigiert.

X.12 HELP

HELP.COM und HELP.HLP

```
Gibt Hilfestellung zu den einzelnen CP/M-Kommandos, mit Beispielen
```

Eingabeformat:

HELP
HELP Stichwort
HELP Stichwort Unterbegriff
HELP Stichwort[Option]
HELP .Unterbegriff
HELP [Option]

Beschreibung:

Mit dem Programm HELP.COM bekommen Sie Hilfe und Informationen über die CP/M-Befehle und -Programme. Sie können HELP nicht aufrufen, wenn Sie gerade mit einem anderen Programm arbeiten.

Anwendung:

Wenn Sie HELP aufrufen, bekommen Sie eine Reihe von Stichwörtern angeboten. Sie können sich das für Sie interessante aussuchen und eingeben. Dabei dürfen Sie das Stichwort bis auf zwei Zeichen abkürzen. Ist die Information länger als 23 Zeilen, stoppt die Anzeige, sobald der Bildschirm gefüllt ist und Sie kommen mit der Leertaste zum nächsten Bildschirm. Geben Sie vor Anwahl der HELP-Funktion ein Control P (^P) ein, werden alle Informationen auf dem Drucker ausgedruckt.

Wollen Sie den Text in der Datei HELP.HLP verändern, müssen Sie zuerst HELP mit der Option EXTRACT eingeben, können dann

Ihren Text einfügen und erstellen die neue, von CP/M lesbare Datei mit der Option CREATE.

Das Kommando HELP ist besonders für Anfänger sehr hilfreich, da man das Handbuch zu CP/M nicht immer parat haben muß. Auch in anderen bekannten Software-Paketen wurde diesem Beispiel von Digital Research nachgeeifert - wieviele Programme haben mittlerweile die sogenannten HELP-SCREENS integriert, die einem die Arbeit mit diesen Programmpaketen erleichtern sollen.

Besonders hilfreich sind die HELP-Möglichkeiten auch bei Systemen, vor denen Sie zum ersten mal sitzen. Nicht selten sind einzelne Kommandos ein wenig oder vollkommen verschieden, weil systemabhängig. Hier ist es empfehlenswert, sich über die HELP-Texte Information zu verschaffen, was zumeist auch viel schneller geht als das Manual irgendwo herauszukramen.

X.13 HEXCOM (Additional Utilities)

HEXCOM.COM

```
Erstellt  eine direkt ausführbare Datei mit der Kennung COM
```

Eingabeformat:

HEXCOM <Dateiname>
HEXCOM

Beschreibung:

Dieses Programm brauchen Sie, wenn Sie in Assembler programmieren. HEXCOM wandelt mit dem Programm MAC.COM erstellte Dateien im INTEL-Hex-Format in ausführbare Dateien mit der Kennung COM um. HEXCOM ist also praktisch der Nachfolger von LOAD (CP/M 2.2). Auch bei HEXCOM brauchen Sie keine Dateikennung anzugeben, HEXCOM hängt automatisch die Kennung ".HEX" an den <Dateinamen> an.

Beachten Sie aber, daß HEXCOM nicht .HEX-Dateien in COM-Dateien umwandelt, die von RMAC, also den relokatiblen Code erstellenden Assembler, erzeugt worden sind. Hierfür müssen Sie sich des Programmes LINK bedienen.

X.14 INITDIR

INITDIR.COM

Bereitet ein Disketten-Inhaltsverzeichnis darauf vor, mit Zeit- und Datumsstempel versehen zu werden

Eingabeformat:

INITDIR B:

Beschreibung:

CP/M 3.0 bietet die Möglichkeit, Dateien mit einem Zeit- und Datumsstempel zu versehen. Die entsprechenden Informationen werden in einem separaten Teil des Directorys gespeichert. Deshalb muß das Inhaltsverzeichnis jeder Diskette entsprechend vorbereitet werden, indem man das Programm INITDIR aufruft. Mit dem Programm SET wählen Sie die Art der Kennzeichnung der Dateien aus.

Anwendung:

Geben Sie INITDIR am Besten bei neuen Disketten ein, da die Einträge im Inhaltsverzeichnis Platz verbrauchen. Benutzen Sie INITDIR mit einer bereits beschriebenen Diskette, machen Sie vorher eine Sicherheitskopie. Wird nämlich das Programm INITDIR während der Arbeit, zum Beispiel durch einen Stromausfall, unterbrochen, verlieren Sie sämtliche Einträge.

Sie sehen, daß es sich nicht nur bei der Systemdiskette von CP/M 3.0 lohnt, eine Sicherheitskopie anzufertigen.

X.15 KEYFIG

KEYFIG.COM

Mittels KEYFIG sind Sie in der Lage, Ihre Tastatur frei umzubelegen. Ferner können Sie auf einzelne Tasten ganze Zeichenketten legen.

Eingabeformat

KEYFIG

Beschreibung:

KEYFIG ist vollkommen menüorientiert. Man kann durch die <CTRL>-<RSHFT>-Tastenkombinationen zumeist dasselbe ohne KEYFIG erreichen, allerdings ist diese Anwendung etwas komplizierter und unübersichtlicher. Für den geübten Anwender erspart diese Methode allerdings auch Zeit. KEYFIG hält auch HELP-Texte (Hilfen) für den Anwender bereit. Mittels KEYFIG sind Sie Herr über die Tastatur: Sie können jeder Taste, ob mit <Shift> oder mit <CTRL>, beliebige ASCII-Werte zuodnen. Ferner haben Sie die Möglichkeit, die Bildschirmfarben zu definieren und den Tasten Spezialfunktionen zuzuordnen (wie beispielsweise das Rebooten des Systems o.ä.). Als Krönung kann man dann den Tasten noch ganz Zeichenketten zuordnen, so wie es beispielsweise bei allen Funktionstasten und der HELP-Taste der Fall ist. Alles in allem: Eine sehr nützliche Sache. Allerdings müssen Sie dieses Kommando mit dem Vorbehalt benutzen, daß Sie ein ähnliches Programm auf einem anderen CP/M-Rechner nicht unbedingt erwarten dürfen.

X.16 LIB (Additional Utilities)

LIB.COM

```
Erstellt und verändert eine "Bibliothek" von Unterprogrammen
```

Eingabeformat:

LIB DATEI.XXX[Optionen]

Beschreibung:

Viele Compiler und die Assembler RMAC und MACRO-80 benutzen die Methode der separaten Unterprogramme, meistens mit der Kennung REL oder IRL versehen. Eine Zusammenstellung der wichtigsten Unterprogramme in einer "Bibliothek" können Sie mit LIB erreichen.

X.17 LINK (Additional Utilities)

LINK.COM

```
Erstellt eine ausführbare Datei aus Unterprogramm-Modulen
```

Eingabeformat:

LINK DATEI, DATEI2, DATEI3,..[Optionen]

Beschreibung:

Viele Compiler und die Assembler RMAC und MACRO-80 benutzen die Methode der separaten Unterprogramme, meistens mit der Kennung REL oder IRL versehen. Mit LINK können diese Dateien zu einer COM-Datei zusammengefasst werden.

X.18 MAC (Additional Utilities)

MAC.COM

Ein Macroassembler für Programme in 8080-Mnemonik und (neu) bedingt auch für Z-80- und 6502-Mnemonik.

Eingabeformat:

MAC DATEI
MAC DATEI $Optionen

Beschreibung:

Drei Dateien werden von dem Macroassembler MAC erstellt, dazu eine Macro-Bibliothek. Das Quellprogramm hat die Kennung ASM und die "Bibliothek" LIB. Der assemblierte Programmcode (Objektcode) steht in der Datei mit der Kennung HEX, die Symbol-Tabelle in der SYM-Datei und das ausdruckbare Listing in der PRN-Datei. Statt mit den üblichen Klammern, werden Optionen mit dem Dollarzeichen ($) gekennzeichnet.

Wenn Sie mehr tiefergreifende Informationen über den Assembler MAC haben wollen, so finden Sie diese im Kapitel VII.

X.19 PATCH

PATCH.COM

Installiert Änderungen in CP/M 3.0 oder Programmen

Eingabeformat:

PATCH DATEI
PATCH DATEI n

Beschreibung.

Mit dem Programm PATCH können Sie Änderungen am CP/M-System vornehmen und in CP/M oder eines der transienten Programme einfügen. Diesen Prozess nennt man patchen. Mit PATCH werden die Veränderungen automatisch in das jeweilige Programm eingefügt, bei mehreren Patches bezieht man sich mit Referenznummern auf den entsprechenden Patch. Die Referenznummern beginnen mit 1.

X.20 PIP

PIP.COM

Kopiert Dateien und transportiert Dateien zwischen Peripheriegeräten

Eingabeformat:

PIP
PIP Zieldatei=Quelldatei[Option]
PIP B:=Datei.XXX
PIP Alles.txt=TEXT1.TXT,TEXT2.TXT...
PIP AB1?*.*=AB2?*.*[Option]

Beschreibung:

PIP ist wohl das leistungsstärkste Programm bei CP/M. Sie können damit Dateien kopieren und zwar von einer Diskette zur anderen, aber auch zwischen verschiedenen Benutzerbereichen, können aus mehreren Dateien eine machen lassen, Ihre zuletzt bearbeiteten Dateien automatisch sichern, Zeilen durchnummerieren, alles in Großbuchstaben umwandeln und das achte Bit auf Null setzten lassen. Wenn Sie genauere Informationen brauchen, sehen Sie bitte in Kapitel VI nach.

Anwendung:

PIP kann mit oder ohne Optionen verwendet werden. Wenn PIP zur Ausführung von Befehlen bereit ist, zeigt es sein Bereitschaftszeichen, den Stern (*). Es erstellt die Zieldatei als genaue Kopie der Quelldatei, außer wenn bestimmte Optionen mit angegeben werden. Die eckigen Klammern müssen direkt, ohne Leerzeichen, hinter dem Dateinamen stehen. Für Buchstaben der Dateinamen, die Sie nicht wissen oder wenn Sie mehrere Dateien einer bestimmten Kategorie kopieren

wollen, können Sie auch die Zeichen "?" für unbekannte Zeichen im Namen oder "*" für alle Zeichen verwenden. Hier ein paar Beispiele:

Ihr Standardlaufwerk ist A:.

PIP B:=DATEI.XXX[V]

Mit diesem Befehl kopieren Sie die Datei DATEI.XXX von A: nach B: und lassen PIP die Korrektheit der neuen Datei überprüfen.

PIP B:=DATEI?.*[V]

So kopieren Sie eine Reihe von Dateien mit dem Namen DATEI, gleich welche Kennung sie haben, von A: nach B:.

PIP B:=*.*[V]

Damit werden alle Dateien eines Laufwerks auf das andere Laufwerk überspielt und das korrekte Kopieren überprüft.

PIP DATEINEU.XXX=DATEIALT.XXX

Mit diesem Befehl kopieren Sie die Datei DATEIALT.XXX in die neu erstellte Datei DATEINEU.XXX auf dem gleichen Laufwerk. Beide Dateien sind anschließend gleichzeitig auf dieser Diskette vorhanden.

PIP ALLES.TXT=TEXT1.TXT,TEXT2.TXT,...

So bringen Sie mehrere Dateien in einer Datei unter, die dann alle anderen enthält.

PIP B:[G] =TEXT.TXT

Die Datei TEXT.TXT wird auf das Laufwerk B: und in den Benutzerbereich "5" kopiert.

Optionen

A — Archiv-Funktion. Kopiert nur Dateien, die seit der letzten Archivierung erstellt oder bearbeitet wurden. Die Funktion Zeit- und Datumstempel muß gesetzt sein.

C — Confirm. CP/M fragt bei jeder Datei nach, ob diese kopiert werden darf.

Dn — Lösche nach n Spalten. PIP löscht alle Zeichen, die nach der n-ten Spalte in der Datei stehen. Nur für Textdateien benutzen.

E — Zeige (echo) Text auf dem Bildschirm. Der Inhalt gerade kopierter Dateien wird auf dem Bildschirm angezeigt. Nicht mit dem Parameter "N" zusammen benutzen. Nur für Textdateien einsetzen.

F — Beseitige das Zeichen für "nächste Seite" (Form Feed). Einige Drucker benötigen ein (^L), um die nächste Seite anzusteuern. Wenn Ihr Drucker das nicht braucht, können Sie die entsprechenden Steuerzeichen mit "F" beseitigen.

Gn — Hole oder bringe eine Datei von oder zu Benutzerbereich "n". Diese Angabe muß direkt hinter dem ersten Dateinamen stehen und ist die einzige, die dort stehen darf. Beispiel:

```
PIP B:[G5]=TEXT.TXT[G1]
```

Kopiert die Datei TEXT.TXT von Benutzerbereich "1" nach Benutzerbereich "5".

H Übertragung von hexadezimalen Daten. Diese Option sollten Sie immer eingeben, wenn Sie HEX-Dateien übertragen wollen. PIP überprüft dann den Inhalt daraufhin, ob die Daten in einwandfreiem INTEL-Format geschrieben sind.

I Ignoriere das Datei-Ende-Zeichen in HEX-Dateien. Geben Sie diese Option für jede Datei ein, außer der letzten. Mit der Option "I" wird gleichzeitig die Option "H" von PIP gesetzt. Verwenden Sie für die letzte Datei den Parameter "H":

```
PIP DATEI.HEX=PROG1.HEX[I],PROG2.HEX[H]
```

K Unterdrücke die Anzeige der Dateinamen während des Kopierens.

L Übersetze alle Großbuchstaben in Kleinbuchstaben. Benutzen Sie diese Option nur für Text-Dateien und verwenden Sie zusätzlich den Parameter "Z" für WordStar-Dateien.

N Nummeriert die Zeilen einer Datei fortlaufend. Die Zahlen beginnen mit 1 in der ersten Zeile und werden pro Zeile und eins erhöht. Die Ziffern können maximal sechstellig sein, nicht gebrauchte Stellen erscheinen als Leerzeichen. Ein Doppelpunkt und ein Leerzeichen stehen hinter den Ziffern. Nicht mit den Optionen "E" oder "N" zusammen benutzen. Für WordStar-Dateien die Optionen "Z" mit eingeben.

N2 Nummeriert die Zeilen für ein BASIC-Programm.

O Übertragung von Objekt-Dateien. Wird benutzt, um Nicht-Text- und Nicht-COM-Dateien zu übertragen. Nicht für Text-Dateien verwenden.

Pn Setzt die Seitenlänge. Voreinstellung ist 60 Zeilen pro Seite. Die "F"-Option sollten Sie mit eingeben, damit vorhandene Zeichen für das Seitenende (^L) gelöscht werden. Nur für Text-Dateien verwenden.

Qxxxx^Z Ende des Kopierens nach dieser Zeichenkette. PIP kopiert eine Datei bis und einschließlich der eingegebenen Zeichenkette. Benutzten Sie den Befehl mit zwei Befehlszeilen, damit die Zeichenkette nicht in Großbuchstaben übersetzt wird:

```
PIP
CON:=TEXT.TXT[Weiler]
```

Schreiben Sie alles in die erste Zeile hinter PIP, erscheint die Eingabe in Großbuchstaben.

R Kopiere SYSTEM-Dateien. Diese Dateien werden mit DIR nicht angezeigt und von PIP normalerweise nicht kopiert. Mit "R" kann diese Einschränkung aufgehoben werden.

Sxxxx^Z Beginne bei dieser Zeichenkette. PIP beginnt mit dem Kopieren an dieser Zeichenkette. Schreiben Sie den Befehl in zwei Zeilen, sonst wird die Zeichenkette in Großbuchstaben übersetzt. Nur für Text-Dateien verwenden.

Tn Schrittweite des Tabulators einstellen. Normalerweise arbeitet CP/M mit einer Schrittweite von acht Zeichen. Beim Drucken einer Datei funktio-

niert dies jedoch nicht und die Schrittweite muß definiert werden. "n" gibt die Anzahl der Leerzeichen für einen Tabulator-Schritt ein. Nur für Text-Dateien benutzen.

U Wandle in Großbuchstaben um. Wandelt alle Kleinbuchstaben in Großbuchstaben um. Nur mit Text-Dateien verwenden. Für WordStar-Dateien zusätzlich die "Z"-Option setzen.

V Überprüfen. Dieser Parameter sorgt dafür, daß PIP die kopierte Datei genau mit der Ursprungsdatei vergleicht.

W Dateien mit dem Attribut Read Only (RO) überschreiben. Diese Dateien können normalerweise nur gelesen, aber nicht verändert oder gelöscht werden. Mit der "W"-Option werden diese Dateien ohne Rückfrage gelöscht.

Z Das Paritäts-Bit (8.Bit) löschen. Der ASCII-Zeichensatz benutzt nur sieben Bits. Das achte Bit wird von verschiedenen Programmen für verschiedene Aufgabe eingesetzt. Die "Z"-Option ist nicht notwendig beim Kopieren nach ASCII-Einheiten, wie CON: oder LST:.

X.21 PUT

PUT.COM

Schreibt die Daten für den Drucker oder den Bildschirm in eine Datei

Eingabeformat:

PUT CONSOLE OUTPUT TO FILE DATEI.XXX[OPTION]
PUT PRINTER OUTPUT TO FILE DATEI.XXX[OPTION]
PUT CONSOLE OUTPUT TO CONSOLE
PUT PRINTER OUTPUT TO PRINTER

Beschreibung:

Normalerweise schickt CP/M die Daten für den Bildschirm zum Bildschirm und die Daten für den Drucker zum Drucker. Mit dem PUT-Befehl können die Daten zusätzlich in eine Datei geschrieben werden.

Anwendung:

Die ersten beiden Zeilen oben sagen CP/M, es soll eine Datei mit dem Namen DATEI.XXX eröffnen und alles, was für den Bildschirm oder den Drucker bestimmt ist, dort hineinschreiben. Ist die Operation beendet, kehrt das Programm wieder zum normalen CP/M-Modus zurück. Die beiden letzten Zeilen brechen das PUT-Programm ab. Diese können Sie verwenden, wenn Sie beispielsweise eine SUBMIT-Datei haben, die den PUT-Befehl benutzt und anschließend wieder in den Normal-Modus finden soll.

Mit der Option NO ECHO verhindern Sie die Anzeige der übertragenen Daten auf dem Bildschirm. Mit der Option FILTER verwandeln Sie jeden Control Charakter in ein druckbares Zeichen.

Mit der Option SYSTEM gehen nicht nur die Daten, sondern auch die Kommandozeile in die Datei.

X.22 REN(AME)

RENAME eingebauter Befehl oder RENAME.COM

Eine Datei umbenennen

Eingabeformat:

RENAME
RENAME DATEI2.XXX=DATEI1.XXX
RENAME AB?*2=AB?*1

Beschreibung:

Mit RENAME ändern Sie den Namen einer Datei - und nur den Namen. Der Inhalt wird nicht verändert.

Anwendung:

Wenn Die Datei nicht auf dem Standard-Laufwerk ist, kann die Laufwerksbezeichnung mit eingegeben werden. Geben Sie RENAME ohne Optionen ein, fragt das Programm danach.

Sollte sich beim Umbennen herausstellen, daß eine bestehende Datei überschrieben würde, so fragt RENAME nach, ob die alte Datei gelöscht werden soll. Beantworten Sie diese Frage mit Nein, so wird das RENAME-Kommando abgebrochen.

X.23 RMAC (Additional Utilities)

RMAC.COM

Macroassembler für die Programmierung in Assembler. Erstellt verschieblichen Objektcode, der mit LINK nachbearbeitet werden muß.

Eingabeformat:

RMAC DATEI
RMAC DATEI SOPTIONEN

Beschreibung:

Drei Dateien werden von dem Macroassembler RMAC erstellt, dazu eine Macro-Bibliothek. Das Quellprogramm hat die Kennung ASM und die "Bibliothek" LIB. Der assemblierte Programmcode steht in der Datei mit der Kennung REL, die Symbol-Tabelle in der SYM-Datei und das ausdruckbare Listing in der PRN-Datei. Mit dem Programm LINK wird ein ausführbares Programm aus der REL-Datei erstellt.

Anwendung:

Statt mit den üblichen Klammern, werden Optionen mit dem Dollarzeichen ($) gekennzeichnet. Vor dem Dollarzeichen muß ein Leerzeichen stehen.

X.24 SAVE

SAVE.COM

Speichert den Inhalt des Arbeitsspeichers in eine Datei

Eingabeformat:

SAVE

Beschreibung:

Der SAVE-Befehl erstellt eine Datei mit dem Inhalt eines bestimmten Speicherbereiches. Vor dem Abspeichern muß ein anderes Programm etwas in den Speicher hineingeschrieben haben.

Anwendung:

Sie müssen SAVE aufrufen, BEVOR Sie ein anderes Programm in den Arbeitsspeicher laden. SAVE plaziert sich automatisch am obersten Ende von CP/M und übergibt dann die Kontrolle wieder an CP/M. Dann lassen Sie das Programm laufen. Sobald es beendet ist, übernimmt SAVE automatisch wieder die Kontrolle. Sie werden dann nach dem Dateinamen, der Start- und Endadresse innerhalb des Speichers gefragt.

X.25 SET

SET.COM

Setzt Datei-Attribute, ermöglicht die Eingabe eines Schutzwortes, vergibt Namen für Disketten und wählt die Art der Zeit- und Datumsstempel

Eingabeformat:

SET DATEI.XXX[Option]
SET AB?*.*[Option]
SET B:[Option]
SET[Option]

Beschreibung:

Das SET-Programm wird für mehrere Dienste eingesetzt. Zu den wichtigsten Diensten gehören: Das setzten des Archiv-Modus und die Möglichkeit, Dateien und ganze Laufwerke zu Read Only-Dateien oder -Laufwerken zu erklären.

Für jede Diskette kann ein eigener Name eingegeben werden, zusätzlich noch ein Schutzwort für jede Diskette und jede Datei. Die Disketten-Namen können mit SHOW angezeigt werden.

Als dritte Möglichkeit können Sie die Dateien mit Stemplen für Zeit und Datum versehen.

Lesen Sie Kapitel V , wenn Sie nähere Informationen brauchen.

X.26 SETDEF

SETDEF.COM

Zeigt oder definiert den Suchpfad und schaltet das bildschirmweise Anzeigen aus oder ein

Eingabeformat:

SETDEF
SETDEF B:
SETDEF[Option]

Beschreibung:

Normalerweise starten Sie ein transientes Programm, indem Sie den Namen eintippen und RETURN hinterher. Wollen Sie das Programm von einem anderen Laufwerk aus aufrufen, geben Sie die Laufwerksbezeichnung mit ein.

Arbeiten Sie zum Beispiel immer auf dem Laufwerk B:, aber Ihre CP/M-Dateien sind auf Laufwerk A:, können Sie CP/M vorschreiben, wo es zuerst nach den Dateien suchen soll. Das nennt man einen Suchpfad vorgeben. Damit könnten Sie CP/M sagen, daß es zuerst auf Laufwerk A:, dann auf Laufwerk C: und erst dann auf dem Standard-Laufwerk suchen soll.

X.27 SHOW

SHOW.COM

```
Zeigt die Optionen einer Diskette an
```

Eingabeformat:

SHOW[Option]
SHOW B:[Option]

Beschreibung:

Mit SHOW können Sie sich sozusagen die "technischen Daten" einer Diskette anzeigen lassen. Als Information können Sie bekommen: den freien Speicherplatz auf der Diskette, die Anzahl der freien Einträge ins Inhaltsverzeichnis, die Zahl der aktiven Benutzer-Bereiche und die angewählte Benutzer-Bereichs-Nummer. Dazu noch Angaben zu der Gesamt-Speicherkapazität der Diskette, Blockgröße, Sektorengröße und Anzahl der Sektoren pro Track. Der Name einer Diskette kann auch angezeigt werden.

SHOW ist als sinnvolle Ergänzung zu DIR zu sehen und auch so zu verwenden. Mittels SHOW können Sie beispielsweise auch erfahren, ob einzelne Dateien oder gar die gesamte Diskette mit einem Schutzwort versehen worden ist.

X.28 SID (Additional Utilities)

SID.COM

Ein Debugger-Programm, um Assembler-Programm zu laden, verändern und zu testen

Eingabeformat:

SID
SID DATEI.XXX
SID DATEI.XXX TEXT.SYM

Beschreibung:

Mit SID können Sie COM- oder HEX-Dateien in den Arbeitsspeicher laden, dort ansehen und verändern. Wenn Sie eine SYM-Datei mitladen, können Sie das Programm unter genauer Kontrolle laufen lassen. SID kann auch ablauffähige Programme in INTEL 8080 Mnemonics umwandeln.

Leider befindet sich auch dieses Programm ausschließlich auf der Additional Utility Diskette und steht somit nur Käufern dieses Paketes zur Verfügung. SID ist übrigens eine Weiterentwicklung des DDT-Programmes unter CP/M 2.2.

X.29 SUBMIT

SUBMIT.COM

Ermöglicht die Befehlseingabe aus einer Datei, statt von der Tastatur

Eingabeformat:

SUBMIT
SUBMIT DATEI.SUB
SUBMIT DATEI DATEI1 DATEI 2 DATEI3...

Beschreibung:

Befehle, die Sie sonst über die Tastatur eingeben, können sie auch in eine SUBMIT-Datei schreiben und dann automatisch abarbeiten lassen. Steht kein Befehl mehr in der Datei, kommt die Kontrolle an CP/M zurück und Sie können normal weitermachen.

CP/M sucht bei jedem Neustart eine Datei namens PROFILE.SUB und führt die dort stehenden Befehle aus, wenn es diese Datei findet.

Anwendung:

Sie schreiben eine SUBMIT-Datei einfach mit Ihrem Textprogramm, indem Sie die gewünschten Befehle jeweils in eine Zeile schreiben. Als Kennung muß diese Datei SUB haben, sonst wird sie von SUBMIT nicht anerkannt.

X.30 USER

USER eingebauter Befehl

Verändert den aktuellen Benutzerbereich

Eingabeformat:

USER
USER n

Beschreibung:

Jede Diskette kann in 16 unterschiedliche Benutzerbereiche unterteilt werden. So hat man für unterschiedliche Arbeiten jeweils einen Benutzerbereich, in dem nur die notwendigen Dateien und Programme stehen. SYSTEM-Dateien in der Benutzerebene Null, können aus allen Benutzerbereichen heraus aufgerufen werden.

Anwendung:

Der aktuelle Benutzerbereich wird im CP/M-Prompt angezeigt (1B>) und kann durch die Eingabe von USER verändert werden. Geben Sie den neuen Bereich nicht mit an, fragt das Programm danach. Mit dem Programm SHOW können Sie sich die aktiven Benutzerbereiche anzeigen lassen.

X.31 XREF (Additional Utilities)

XREF.COM

```
Erstellt eine Referenzliste für Assembler-Programme
```

Eingabeformat:

XREF DATEI
XREF DATEI $P

Beschreibung:

Die Assembler MAC und RMAC erstellen eine alphabetische Liste aller Symbole und deren Werte in einem Programm. XREF erhöht den Komfort dadurch, daß es zu jedem Symbol noch mit angibt, in welcher Programmzeile es zu finden ist. Das Programm XREF benötigt die Dateien SYM und PRN. XREF erstellt unter dem Dateinamen <Dateiname>.XRF eine Assembler-Liste (ähnlich dem <Dateiname>.PRN) und hängt hieran noch eine ausführliche Auflistung aller vorkommenden Symbole inklusive Angabe der Vorkommnisse im Programm unter Angabe der Zeilennummer an. Man nennt so etwas auch Cross Reference List. Die Amerikaner schreiben für Cross gerne nur kurz den Buchstaben X, weshalb dieses Kommando XREF heißt.

Anhang 1

Umrechnungstabelle

Umrechnungstabelle

DEZIMAL	HEXA-DEZIMAL	BINÄR	ASCII
0	00	000 0000	NUL
1	01	000 0001	SOH
2	02	000 0010	STX
3	03	000 0011	ETX
4	04	000 0100	EOT
5	05	000 0101	ENQ
6	06	000 0110	ACU
7	07	000 0111	BEL
8	08	000 1000	ES
9	09	000 1001	HT
10	0A	000 1010	LF
11	0B	000 1011	VT
12	0C	000 1100	FF
13	0D	000 1101	CR
14	0E	000 1110	SO
15	0F	000 1111	SI
16	10	001 0000	OLE
17	11	001 0001	DC1
18	12	001 0010	DC2
19	13	001 0011	DC3
20	14	001 0100	DC4
21	15	001 0101	AAK
22	16	001 0110	SYU
23	17	001 0111	ETB
24	18	001 1000	CAN
25	19	001 1001	EM
26	1A	001 1010	SUB
27	1B	001 1011	ESC
28	1C	001 1100	FS
29	1D	001 1101	GS
30	1E	001 1110	RS

DEZIMAL	HEXA-DEZIMAL	BINÄR	ASCII
31	1F	001 1111	VS
32	20	010 0000	SP
33	21	010 0001	!
34	22	010 0010	"
35	23	010 0011	#
36	24	010 0100	$
37	25	010 0101	%
38	26	010 0110	&
39	27	010 0111	'
40	28	010 1000	(
41	29	010 1001	)
42	2A	010 1010	*
43	2B	010 1011	+
44	2C	010 1100	,
45	2D	010 1101	-
46	2E	010 1110	.
47	2F	010 1111	/
48	30	011 0000	0
49	31	011 0001	1
50	32	011 0010	2
51	33	011 0011	3
52	34	011 0100	4
53	35	011 0101	5
54	36	011 0110	6
55	37	011 0111	7
56	38	011 1000	8
57	39	011 1001	9
58	3A	011 1010	:
59	3B	011 1011	;
60	3C	011 1100	<
61	30	011 1101	=
62	3E	011 1110	>
63	3F	011 1111	?

DEZIMAL	HEXA-DEZIMAL	BINÄR	ASCII
64	40	100 0000	@
65	41	100 0001	A
66	42	100 0010	B
67	43	100 0011	C
68	44	100 0100	D
69	45	100 0101	E
70	46	100 0110	F
71	47	100 0111	G
72	48	100 1000	H
73	49	100 1001	I
74	4A	100 1010	J
75	4B	100 1011	K
76	4C	100 1100	L
77	40	100 1101	M
78	4E	100 1110	N
79	4F	100 1111	O
80	50	101 0000	P
81	51	101 0001	Q
82	52	101 0010	R
83	53	101 0011	S
84	54	101 0100	T
85	55	101 0101	U
86	56	101 0110	V
87	57	101 0111	W
88	58	101 1000	X
89	59	101 1001	Y
90	5A	101 1010	Z
91	58	101 1011	Ä
92	5C	101 1100	Ö
93	5D	101 1101	Ü
94	5E	101 1110	^
95	5F	101 1111	_
96	60	110 0000	

DEZIMAL	HEXA-DEZIMAL	BINÄR	ASCII
97	61	110 0001	a
98	62	110 0010	b
99	63	110 0011	c
100	64	110 0100	d
101	65	110 1101	e
102	66	110 0110	f
103	67	110 0111	g
104	68	110 1000	h
105	69	110 1001	i
106	6A	110 1010	j
107	6B	110 1011	k
108	6C	110 1100	l
109	60	110 1101	m
110	6E	110 1110	n
111	6F	110 1111	o
112	70	111 0000	p
113	71	111 0001	q
114	72	111 0010	r
115	73	111 0011	s
116	74	111 0100	t
117	75	111 0101	u
118	76	111 0110	v
119	77	111 0111	w
120	78	111 1000	x
121	79	111 1001	y
122	7A	111 1010	z
123	7B	111 1011	ä
124	7C	111 1100	:
125	70	111 1101	ü
126	7E	111 1110	'
127	7F	111 1111	DEL

Anhang 2

Die CP/M Control-Zeichen

Die CP/M Control-Zeichen

Befehl	Wirkung
^A	Cursor ein Zeichen nach links*
^B	Cursor an Anfang der Zeile oder an's Ende, wenn Cursor schon am Anfang steht
^C	CP/M abbrechen; Reset durchführen
^E	Cursor zur nächsten Zeile
^F	Cursor ein Zeichen rechts*
^G	Zeichen unter Cursor löschen*
^H	Lösche Zeichen links vom Cursor
^I	Cursor zur nächsten TAB-Position
^J	Ausführungsbefehl (Zeilenschaltung)
^K	Lösche von Cursor bis Zeilenende
^M	wie RETURN
^P	Drucker ein/ausschalten
^Q	Scrollen einschalten nach ^S
^R	Befehlszeile wiederholen
^S	Bildschirm-Ausgabe anhalten
^U	Alle Zeichen in Zeile löschen

^W	Alte Befehlszeile wieder anzeigen*
^X	Zeichen links vom Cursor löschen
^Z	String-Ende in PIP und ED

* bedeutet: nur bei CP/M mit "bankswitching" verfügbar

Anhang 3

Alle PIP-Parameter

PIP - Parameter

A — Archiv-Funktion. Kopiert nur Dateien, die seit der letzten Archivierung erstellt oder bearbeitet wurden. Die Funktion Zeit- und Datumsstempel muß gesetzt sein.

C — Confirm. CP/M fragt bei jeder Datei nach, ob diese kopiert werden darf.

Dn — Lösche nach n Spalten. PIP löscht alle Zeichen, die nach der n-ten Spalte in der Datei stehen. Nur für Textdateien benutzen.

E — Zeige (echo) Text auf dem Bildschirm. Der Inhalt gerade kopierter Dateien wird auf dem Bildschirm angezeigt. Nicht mit dem Parameter "N" zusammen benutzen. Nur für Textdateien einsetzen.

F — Beseitige das Zeichen für "nächste Seite" (Form Feed). Einige Drucker benötigen ein (^L), um die nächste Seite anzusteuern. Wenn Ihr Drucker das nicht braucht, können Sie die entsprechenden Steuerzeichen mit "F" beseitigen.

Gn — Hole oder bringe eine Datei von oder zu Benutzerbereich "n". Diese Angabe muß direkt hinter dem ersten Dateinamen stehen und ist die einzige, die dort stehen darf. Beispiel:

```
PIP B:[G5]=TEXT.TXT[G1]
```

Kopiert die Datei TEXT.TXT von Benutzerbereich "1" nach Benutzerbereich "5".

H — Übertragung von hexadezimalen Daten. Diesen Parameter sollten Sie immer eingeben, wenn Sie HEX-Dateien

übertragen wollen. PIP überprüft dann den Inhalt daraufhin, ob die Daten in einwandfreiem INTEL-Format geschrieben sind.

I Ignoriere das Datei-Ende-Zeichen in HEX-Dateien. Geben Sie diesen Parameter für jede Datei ein, außer der letzten. Mit der Option "I" wird gleichzeitig die Option "H" von PIP gesetzt. Verwenden Sie für die letzte Datei den Parameter "H":

```
PIP DATEI.HEX=PROG1.HEX[I],PROG2.HEX[I],PROG3.HEX[H]
```

K Unterdrücke die Anzeige der Dateinamen während des Kopierens.

L Übersetze alle Großbuchstaben in Kleinbuchstaben. Benutzen Sie diesen Parameter nur für Text-Dateien und verwenden Sie zusätzlich den Parameter "Z" für WordStar-Dateien.

N Nummeriert die Zeilen einer Datei fortlaufend. Die Zahlen beginnen mit 1 in der ersten Zeile und werden pro Zeile und eins erhöht. Die Ziffern können maximal sechstellig sein, nicht gebrauchte Stellen erscheinen als Leerzeichen. Ein Doppelpunkt und ein Leerzeichen stehen hinter den Ziffern. Nicht mit den Parametern "E" oder "N" zusammen benutzen. Für WordStar-Dateien den Parameter "Z" mit eingeben.

N2 Nummeriert die Zeilen für ein BASIC-Programm.

O Übertragung von Objekt-Dateien. Wird benutzt, um Nicht-Text- und Nicht-COM-Dateien zu übertragen. Nicht für Text-Dateien verwenden.

Pn Setzt die Seitenlänge. Voreinstellung ist 60 Zeilen

pro Seite. Den "F"-Parameter sollten Sie mit eingeben, damit vorhandene Zeichen für das Seitenende (^L) gelöscht werden. Nur für Text-Dateien verwenden.

Qxxxx^Z Ende des Kopierens nach dieser Zeichenkette. PIP kopiert eine Datei bis und einschließlich der eingegebenen Zeichenkette. Benutzten Sie den Befehl mit zwei Befehlszeilen, damit die Zeichenkette nicht in Großbuchstaben übersetzt wird:

```
PIP
CON:=TEXT.TXT[Weiler]
```

Schreiben Sie alles in die erste Zeile hinter PIP, erscheint die Eingabe in Großbuchstaben.

R Kopiere SYSTEM-Dateien. Diese Dateien werden mit DIR nicht angezeigt und von PIP normalerweise nicht kopiert. Mit "R" kann diese Einschränkung aufgehoben werden.

Sxxxx^Z Beginne bei dieser Zeichenkette. PIP beginnt mit dem Kopieren an dieser Zeichenkette. Schreiben Sie den Befehl in zwei Zeilen, sonst wird die Zeichenkette in Großbuchstaben übersetzt. Nur für Text-Dateien verwenden.

Tn Schrittweite des Tabulators einstellen. Normalerweise arbeitet CP/M mit einer Schrittweite von acht Zeichen. Beim Drucken einer Datei funktioniert dies jedoch nicht und die Schrittweite muß definiert werden. "n" gibt die Anzahl der Leerzeichen für einen Tabulator-Schritt ein. Nur für Text-Dateien benutzen.

U Wandle in Großbuchstaben um. Wandelt alle Kleinbuchstaben in Großbuchstaben um. Nur mit Text-

Dateien verwenden. Für WordStar-Dateien zusätzlich den "Z"-Parameter setzen.

V Überprüfen. Dieser Parameter sorgt dafür, daß PIP die kopierte Datei genau mit der Ursprungsdatei vergleicht.

W Datein mit dem Attribut Read Only (RO) überschreiben. Diese Dateien können normalerweise nur gelesen, aber nicht verändert oder gelöscht werden. Mit dem "W"-Parameter werden diese Dateien ohne Rückfrage gelöscht.

Z Das Paritäts-Bit (8.Bit) löschen. Der ASCII-Zeichensatz benutzt nur sieben Bits. Das achte Bit wird von verschiedenen Programmen für verschiedene Aufgabe eingesetzt. Der "Z"-Parameter ist nicht notwendig beim Kopieren nach ASCII-Einheiten, wie CON: oder LST:.

Anhang 4

Die SET-Parameter

OPTION	BEDEUTUNG
DIR	Macht eine SYSTEM-Datei wieder im normalen Inhaltsverzeichnis sichtbar
SYS	Macht eine Datei zur SYSTEM-Datei
RO	Bestimmt, daß eine Datei nur gelesen werden kann
RW	Bestimmt, daß eine Datei gelesen und verändert werden kann
ARCHIV=OFF	Setzt das ARCHIV-Attribut auf "aus". Das bedeutet, daß diese Datei noch nicht gesichert (archiviert) wurde. Das Programm PIP mit der Option [A] kann Dateien mit mit dem Attribut ARCHIV=OFF kopieren. Den PIP-Befehl geben Sie mit den Sternchen für die Dateinamen ein und PIP kopiert alle Datei, die seit dem letzten Kopieren mit PIP und der ÄAÜ-Option verändert wurden. Nachdem PIP kopiert hat, setzt es die Datei-Attribute auf ARCHIV=ON.
ARCHIV=ON	Setzt das ARCHIV-Attribut auf "ein". Das bedeutet, daß diese Datei gesichert wurde. Normalerweise ändert PIP mit der Option [A] das Attribut nach dem Sichern von Dateien.Sie können das Attribut auch selber ändern, wenn Sie den SET-Befehl verwenden.
F1=ON/OFF	Schaltet das benutzerdefinierte Datei-

Attribut F1 ein oder aus.

F2=ON/OFF — Schaltet das benutzerdefinierte Datei-Attribut F2 ein oder aus.

F3=ON/OFF — Schaltet das benutzerdefinierte Datei-Attribut F3 ein oder aus.

F4=ON/OFF — Schaltet das benutzerdefinierte Datei-Attribut F4 ein oder aus.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Bücher zum Commodore 128

Der Commodore 128 steht seinem kleinen Bruder, dem C-64, in bezug auf Grafik in nichts nach – das beweist dieses Buch. Ein Team von Grafik-Spezialisten deckt wirklich alle Geheimnisse des C-128 auf. Mit diesem Buch können Sie sofort alle Möglichkeiten Ihres Commodore-Rechners für eigene Programme nutzen!

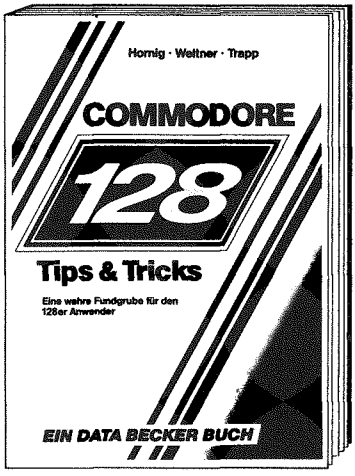
Aus dem Inhalt:

- Die 3 Modi des C-128
- Grafikbefehle im 128er-Modus
- Betriebsarten des VIC-Chips
- Verwaltung der HI-RES
- Lage des Grafikspeichers
- Farbgebung in der HRG-MC-Grafik
- Programmierung von Sprites
- IMR-IRQ-Interruptprogrammierung
- Der Lightpen
- Register des VDC-Chips
- Aufbau des Video-RAM
- Der Charactergenerator
- HI-RES-Grafik mit dem VDC
- Animationsgrafiken
- Statistische Auswertungen
- Funktionsplotter
- Einführung in CAD
- Ein-/Ausgabe von Grafiken
- Grafikprogrammierung in 6502-Assembler

Durben, Löffelmann, Plenge, Vüllers
Das große Grafik-Buch zum C128
369 Seiten, DM 39,–
ISBN 3-89011-154-8

Bücher zum Commodore 128

128 Tips & Tricks ist eine riesige Fundgrube für jeden 128er-Besitzer, der mehr mit seinem Rechner machen will. Dieses Buch enthält nicht nur viele Beispielprogramme, sondern erläutert auch leichtverständlich den Aufbau des Rechners und seine Programmierung.



Aus dem Inhalt:

- Grafik auf dem Commodore 128
- Arbeiten mit mehreren Bildschirmen
- Eigener Zeichensatz
- Sprite-Handling
- Grafik mit den eingebauten Befehlen
- Simulation mehrerer Windows
- Listing-Konverter
- Modifiziertes Input
- Software-Schutz auf dem Commodore 128
- Zeilen einfügen
- Rund um die Tastatur
- Befehlserweiterung – selbst gemacht
- Banking
- Weitere Möglichkeiten der MMU
- Autostart
- Der Speicher
- Wechseln des Betriebssystems
- Der 64er-Modus auf dem C-128
- Die 10er-Tastatur am C-64
  und vieles mehr

Hornig, Weltner, Trapp
Commodore 128 Tips & Tricks
Hardcover, 327 Seiten, DM 49,–
ISBN 3-89011-097-5

Bücher zum Commodore 128

Das große BASIC-Buch zum Commodore 128 ist eine ausführliche, didaktisch gut geschriebene Einführung in das CBM BASIC 7.0. Von den BASIC-Befehlen über die Problemanalyse bis zum fertigen Algorithmus lernt man schnell und sicher das Programmieren. Übungsaufgaben helfen, das Gelernte zu vertiefen. Gleichzeitig erhält der BASIC-Programmierer ein praxisbezogenes Nachschlagewerk.

Aus dem Inhalt:

- Datenfluß- und Programmablaufpläne
- Fortgeschrittene Programmiertechniken
- Menüerstellung
- Mehrdimensionale Felder
- Sortierroutinen
- Dateiverwaltung
- Windowprogrammierung
- BASIC intern
- Tokentabelle
- Der Monitor
  und viele nützliche Utilities

Kampow
Das große BASIC-Buch zum Commodore 128
452 Seiten, DM 39,–
ISBN 3-89011-114-9

Bücher zum Commodore 128

Das große Floppybuch zur 1570/1571 gibt Ihnen das notwendige Wissen zur Programmierung Ihres neuen Diskettenlaufwerkes. Für Anfänger, Fortgeschrittene und Profis. Dieses Buch beschreibt wirklich alle Leistungsmerkmale dieser schnellen Floppy.

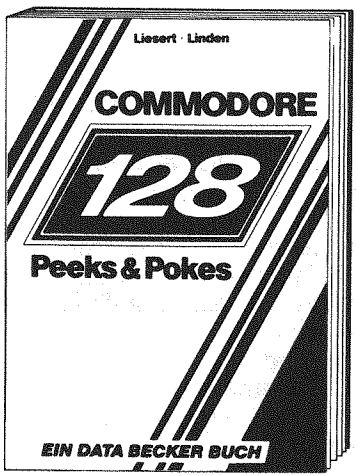
Aus dem Inhalt:

- Einführung für Einsteiger
- Die Floppy und das COMMODORE-BASIC
- Sequentielle und relative Dateien
- Fremde Diskettenformate verarbeiten
- Programmierung im DOS-Puffer
- Die CP/M-Fähigkeiten der 1570/1571
- Floppy intern: Schaltungsaufbau und Funktion
- 1571 Fast-Load
- Das DOS im Detail
- Komplettes DOS-Listing (mit Cross-Reference)

Ellinger
Das große Floppybuch zur 1570/1571
Hardcover, 554 Seiten, DM 49,–
ISBN 3-89011-124-6

Bücher zum Commodore 128

Schlagen Sie dem Betriebssystem Ihres C128 ein Schnippchen. Wie? Mit PEEKS & POKES natürlich! Dieses Buch erklärt leichtverständlich den Umgang damit. Mit einer riesigen Anzahl wichtiger POKES und ihren Anwendungsmöglichkeiten. Nebenbei wird der interne Aufbau Ihres neuen C128 prima erklärt.



Aus dem Inhalt:

- Die Arbeitsweise Ihres Rechners
- Was ist ein Betriebssystem?
- Wie arbeitet der Interpreter
- RAM-Erweiterungsbefehle
- Bankswitching
- Die Zeropage
- Pointer & Stacks
- Speicherbelegungsplan
- Massenspeicherung & Peripherie
- Der 40-/80-Zeichen-Bildschirm
- Sprites
- Grafik mit 640×200 Punkten
- Die Tastatur
- Der User-Port
- BASIC und Betriebssystem
- Grundlagen der Maschinensprache
- 8502-/Z80-Maschinensprache

Liesert, Linden
Peeks & Pokes zum Commodore 128
248 Seiten, DM 29,–
ISBN 3-89011-138-6

DFÜ für jedermann – das ist nicht nur eine ausführliche und leichtverständliche Einführung in das Gebiet der Datenfernübertragung mit dem C 64 und dem C 128, sondern gibt handfeste Informationen zur effektiven Nutzung der vorhandenen Kommunikationsnetze für Einsteiger und Profis. Einrichten einer eigenen Mailbox und das erste Terminalprogramm gehören ebenso zum Inhalt wie die aktuellen Mailboxnummern in aller Welt.

Aus dem Inhalt:

- Was ist DFÜ?
- Die Netze der Post
- Wichtige Postbestimmungen und Gebühren
- DATEX-P
- BTX
- Alles über Akustikkoppler und Modems
- Einrichtung und Benutzung von Mailboxen
- Der Zugriff auf Datenbanken
- Begriffserklärungen: Originate, Answer, Half-Duplex usw.
- Serielle Schnittstelle und ihre Belegung am C 64/C 128

Severin
DFÜ für jedermann
331 Seiten, DM 39,–
ISBN 3-89011-141-6

Computerschach

Wie funktioniert eigentlich ein Schachprogramm? Die wenigsten wissen, mit welchen Algorithmen man dem Computer das königliche Spiel beibringen kann. Dieses Buch informiert über alle Aspekte der Schachprogrammierung und enthält ein komplettes Schachprogramm in BASIC, das auf anschauliche Weise die Probleme der Programmierung und ihre Lösungen darstellt. Abgerundet wird das Buch durch eine kleine Geschichte des Computerschach und eine Menge Tips zum Thema „Wie spielt man Schach gegen den Computer?".

Aus dem Inhalt:

- Programme, Partien und Personen
- Strategiespiele auf dem Computer
- Stack und Rekursion in BASIC
- Brettdarstellung und Zuggenerierung
- Suchalgorithmen in Schachprogrammen
- Bewertungsfunktionen
- Komplettes Schachprogramm in BASIC
- Ausführliche Dokumentation zum Programm
- Testverfahren für Schachprogramme
- 7 goldene Regeln zum Spiel gegen Computer

Bartel, Kraas, Schrüfer
Das große Computerschach-Buch
462 Seiten, DM 49,–
ISBN 3-89011-117-3