Schmidt · Szczepanowski

COMMODORE 128

für Einsteiger

EIN DATA BECKER BUCH

Schmidt · Szczepanowski

COMMODORE 128

für Einsteiger

EIN DATA BECKER BUCH

ISBN 3-89011-099-1

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.

VORWORT

Der COMMODORE 128 ist zur Zeit einer der leistungsfähigsten Homecomputer auf dem Markt. Zum Einen hat der Rechner selbst hervorragende Eigenschaften, wie 80-Zeichen, hochauflösende Grafik, Synthesizer, CP/M-fähig, voll C 64 kompatibel, BASIC das kaum noch Wünsche offenläßt. Zum Anderen tritt das wesentliche Problem jedes neuen Rechners, daß keine Programme für diesen erhältlich sind, überhaupt nicht auf. Durch die 100 %ige Kompatibilität zum C 64 ist sämtliche Software für diesen Rechner auf dem COMMODORE 128 lauffähig. Zusätzlich erschließt sich durch die CP/M-Fähigkeit ein weiteres riesiges Softwareangebot. Sie können also sicher sein die richtige Wahl getroffen zu haben.

Gerade soviel Leistungsfähigkeit kann den Einsteiger allerdings eher verunsichern anstatt ihn von seinem neuen Rechner zu begeistern. Was ist denn nun eigentlich zu tun, wenn der Rechner gerade ausgepackt ist?
Genau hier setzt dieses Buch an. Vom Auspacken des Rechners über den Anschluß aller Kabel bis zur Programmierung von Sprites werden Sie keinen Augenblick alleingelassen. Jeder Handgriff, jede Taste ist genau beschrieben und wenn nötig auch im Bild gezeigt.
Damit tauchen die sonst zwangsläufig auftretenden Einsteigerprobleme erst gar nicht auf. Sie werden langsam aber sicher zum "alten Hasen" an Ihrem Rechner ohne Gefahr zu laufen "den Kram hinzuschmeißen" weil "ewig" alles nicht klappt.

Und nun viel Spaß bei Ihren ersten Schritten mit dem COMMODORE 128!

DATA BECKER GmbH

INHALTSVERZEICHNIS

Der erste Befehl

Das erste Programm

BASIC sinnvoll eingesetzt

Die Zusatzgeräte

ANWENDUNGSBEISPIELE

ANHANG

EINLEITUNG

Durch seine Konzeption nimmt der COMMODORE 128 eine besondere Stellung unter den Homecomputern ein. Es wäre durchaus berechtigt beim 128er von drei Computern zu sprechen. Nämlich:

1. C 64
2. CP/M fähiger Rechner
3. COMMODORE 128

Das ist keineswegs nur eine werbewirksame Aussage. Die Commodore Entwickler haben tatsächlich diese drei Rechner in ein Gehäuse gepackt. Es gibt deshalb auch keinerlei Einschränkungen für eine der drei Betriebsarten.
Ihnen als Anwender bringt diese Konzeption den Vorteil für nur einen angeschafften Rechner das Programmangebot von drei Rechnern nutzen zu können. Wenn Sie den 128er zum beruflichen Einstieg in die EDV verwenden wollen, so lernen Sie für den Preis eines Rechners gleich drei unterschiedliche Systeme kennen.
Für Sie als Einsteiger empfiehlt es sich allerdings nicht, gleich alles auf einmal kennenlernen zu wollen. Deshalb besprechen wir in diesem Buch den COMMODORE 128 nur in seinem eigenen Modus. Grundlagen, die Sie dabei kennenlernen, werden Ihnen auch beim späteren Einstieg in CP/M eine nützliche Hilfe sein. Der C 64 Modus ist ohnehin nur für den Ablauf fertiger Software interessant. Dennoch sei hier kurz erklärt, wie Sie sofort nach dem Einschalten in einen der drei Modi gelangen:

C 64 Modus - während dem Einschalten die linke untere Taste(mitCOMMODORESymbol)gedrückthalten
CP/M Modus - vor Einschalten des Rechners Floppy einschalten und CP/M Diskette einlegen
128 Modus - erst Rechner, dann Floppy einschalten

Diese Beschreibung ist zum Teil ein Vorgriff auf noch folgende Kapitel. Sie sollten nur grundsätzlich im Kopf behalten, das es andere Modi als den 128er Modus gibt und wie diese gestartet werden. Wenn Sie dann versehentlich

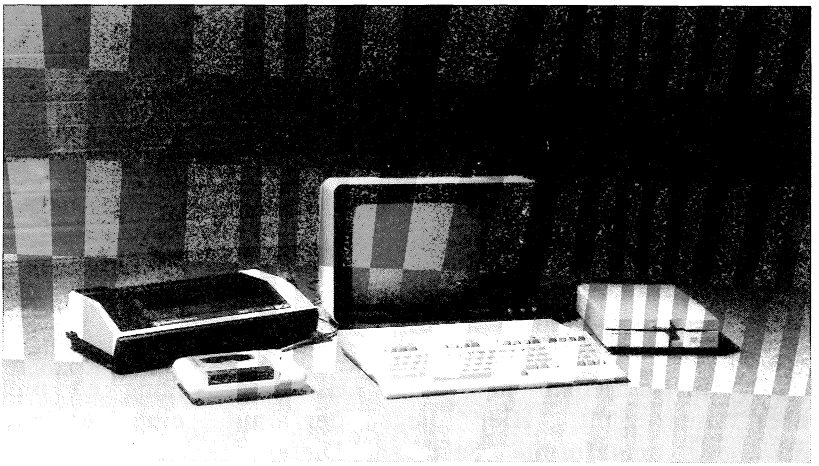
Ihren Rechner so starten, daß ein anderer als der 128er Modus auftritt, können Sie leichter nachvollziehen woran das lag und den Startvorgang richtig wiederholen.

Bleibt noch eine Vereinbarung zu treffen: Im 128er Modus haben Sie die Möglichkeit 80- oder 40- Zeichen pro Zeile auf dem Bildschirm darzustellen. Die 80-Zeichen Darstellung ist nur auf einem speziellen Monitor möglich. Wir gehen davon aus, daß die meisten Einsteiger ein Fernsehgerät benutzen. Auf einem Fernsehgerät ist nur die Darstellung von 40 Zeichen pro Zeile möglich. Deshalb beziehen sich alle Angaben und Abbildungen auf den 40-Zeichen Bildschirm.
Die Umschaltung von 40 auf 80 Zeichen erfolgt mittels einer Taste in der obersten Tastaturreihe mit der Aufschrift: '40/80 DISPLAY'. Achten Sie also bitte darauf, daß diese Taste V O R dem Einschalten Ihres 128ers ausgerastet, also nicht gedrückt, ist.
An der Funktion der BASIC Befehle ändert sich durch die Darstellung in 40 oder 80 Zeichen nichts. Lediglich die Grafikbefehle beziehen sich ausschließlich auf den 40-Zeichen Bildschirm.

NACH DEM AUSPACKEN

Dieses Kapitel richtet sich an alle diejenigen, die seit wenigen Stunden stolze Besitzer eines COMMODORE 128 und diverser Zusatzgeräte sind. Wenn der Verkäufer das Anschließen der Geräte auch noch so ausführlich erklärt hat, so sind die Gedanken in der Aufregung meist ganz woanders. Kommt man dann nach Hause und packt die Geräte aus, ist der ein oder andere meist ratlos. Überlegungen wie "wird das Netzkabel jetzt hinten oder an der Seite des 128ers angeschlossen" können dann schlimme Folgen haben. Kunden, die durch falschen Anschluß der Geräte dessen Innereien völlig zerstörten, sind keine Seltenheit. Bevor Sie sich auf Abenteuer einlassen, studieren Sie besser das folgende, von zahlreichen dokumentierenden Bildern begleitete Verkabelungs-Kapitel.

Wenn Sie alle Geräte ausgepackt und übersichtlich angeordnet haben, so stehen Sie hier vor einem Berg von Geräten und da vor einem Kabelwirrwarr.



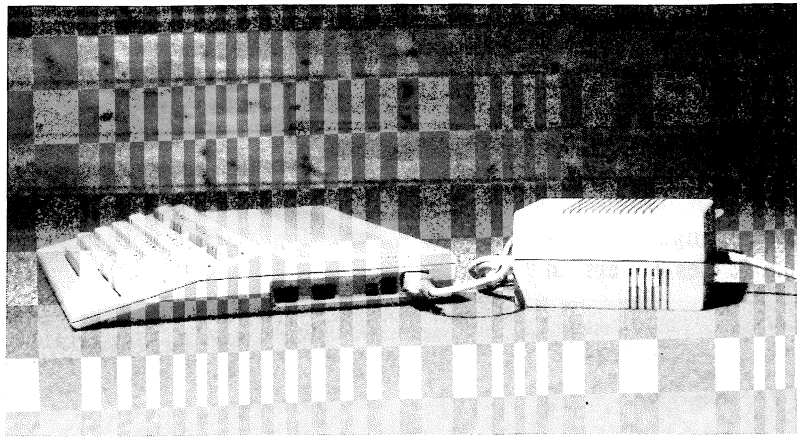
Diese Geräte sind zwar schön anzusehen, jedoch völlig funktionsuntüchtig. Erst die Stromversorgung und die nötigen Verbindungen der Geräte machen aus diesem "lahmen Haufen" geballte Computertechnik. Dazu benötigen Sie

jedoch Kabel, die bei allen Geräten mitgeliefert werden. Alle Kabel zusammen geben einen erschreckenden Anblick:

Das Netzgerät

Mit dem 128er wird ein externes Netzgerät geliefert, das ihn mit dem nötigen Strom versorgt. Sicherlich ist dieses Gerät aus gutem Grund nicht in den 128er eingebaut worden. Erstens wäre die Temperaturentwicklung unangenehm für den 128er, zweitens würde er ein größeres Gehäuse benötigen.

Wie und wo schließt man dieses Netzgerät an? Nun, das Netzgerät ist mit zwei Kabeln ausgerüstet. Das Ende des einen Kabels - Ihnen nicht mehr unbekannnt - ist ein Netzstecker. Daß dieser nicht in den 128er eingesteckt wird, dürfte jedem klar sein. Das andere Ende ist nun aber interessant. Hier liegt der auf eine geringe Spannung umgeformte "Saft" an. Dieser Stecker kommt in die dafür vorgesehene, an der rechten Seite des 128ers angeordnete Buchse. Das folgende Bild macht dies deutlich:



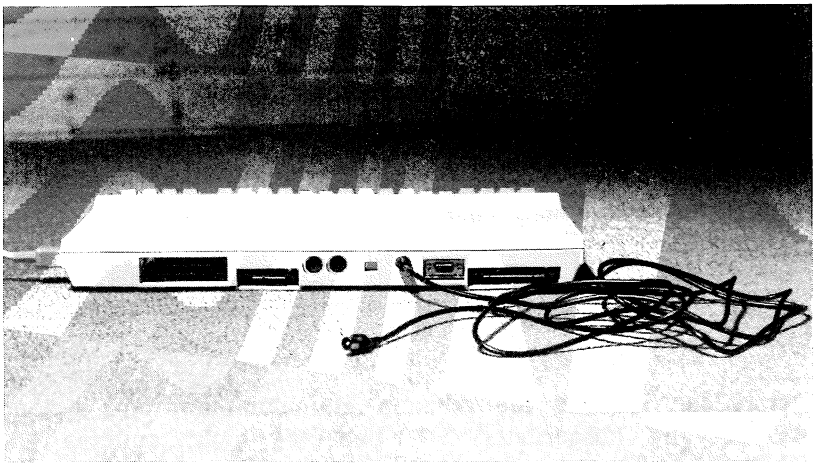
Alles klar? Bitte stecken Sie den Netzstecker noch nicht in die Steckdose, das machen wir zum Schluß mit allen Geräten gemeinsam.

Der Fernseher

Der 128er ist wie die meisten Heimcomputer mit einem HF-Ausgang ausgerüstet. HF bedeutet nicht anderes als Hochfrequenz, was bedeutet, daß Sie den 128er im 40 Zeichenmodus an jeden gewöhnlichen Fernseher anschließen können. Monitore oder sogar Farbmonitore sind nicht für jeden erschwinglich, einen Fernseher jedoch findet man in fast jeder Familie. Problematisch wird es nur, wenn die Mutter den Western mit John Wayne sehen will, der Vater das auf dem Videorecorder aufgezeichnetete Fußballspiel nicht missen und dann noch der Sohnemann das Spiel-Programm aus der letzten Computerzeitschrift abtippen möchte. Ein Familienstreit ist dann meist die Folge. In solchen Situationen ist entweder ein zweiter Fernseher oder aber ein Monitor, den man übrigens auch an den Videorecorder anschließen kann, sehr nützlich. Außerdem können Sie nur mit dem Commodore 1902 oder einem dazu kompatiblen Monitor den 128 im 80 Zeichen Modus betreiben. Wir gehen im folgenden immer davon aus, daß Sie Ihren 128er im 40 Zeichenmodus an einem Fernseher oder Monitor betreiben. Das heißt, daß die Taste mit der Aufschrift "40/80 DISPLAY" vor dem Einschalten nicht

eingerastet ist. Achten Sie bitte darauf, da andernfalls auf dem 40 Zeichen Bildschirm keine Zeichen erscheinen werden.

Doch bleiben wir beim Fernseher. Dem 128er liegt ein Antennenkabel bei, mit dem Sie ihn mit der "Glotze" verbinden können. Dazu schließen Sie das Ende mit dem etwas längeren Stecker hinten an die dafür vorgesehene Buchse des 128ers an:

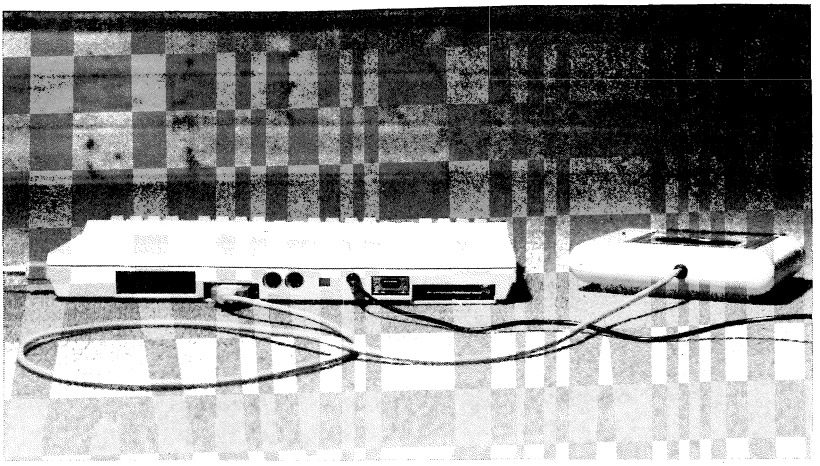


Nun wird das andere Ende in den Antenneneingang des Fernsehers eingesteckt. Das soll zunächst reichen. Das Bild stellen wir später nach dem Einschalten aller Geräte ein.

Die Datassette

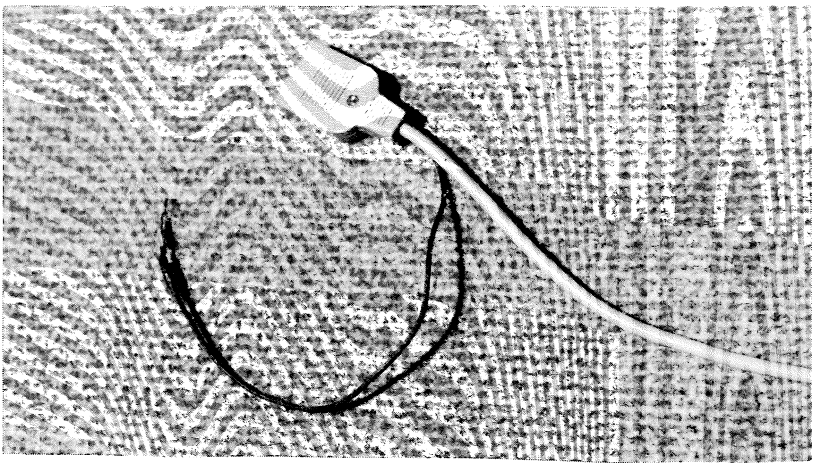
Die Datassette ist ein Kassettenrecorder, der zu einem äußerst günstigen Preis die externe Speicherung von Programmen und Daten ermöglicht. Ein 128er ohne externes Speichermedium ist wie ein Auto ohne Benzin. Sie können zwar ohne Datassette oder Diskettenlaufwerk programmieren, jedoch sind diese mühevoll erstellten Programme nach dem Ausschalten der Rechners wieder verloren. Auch können Sie keine gekauften Programme einsetzen, da diese nur über Kassette oder Diskette eingeladen werden können.

Wenn Sie die Datassette ausgepackt haben, so stellt sich die Frage, wo dieses kleine Gerät denn angeschlossen wird. Nun, hinten am 128er ist bereits ein Anschluß für ein Kassettenlaufwerk vorhanden. An diesen sogenannten Kassettenport wird das Kabel des Recorders mit dem breiten Stecker angeschlossen.



Aber woher bekommt der Recorder die notwendige Betriebsspannung? Er besitzt ja keinen Netzanschluß. Sicher haben Sie erkannt, daß die Datassette vom 128er über das Anschlußkabel mit Strom versorgt wird.

Haben Sie das kleine Kabel am Stecker des Kassettenrecorders bemerkt? Es gibt Leute, die grübeln oft stundenlang nach, wo dieses kleine Kabel denn nun angeschlossen wird.



Wohin denn nun mit diesem Kabel? Wenn die Datassette am 128er angeschlossen wird, so ist dieses Kabel überflüssig. Es ist ein Massekabel, daß nur an COMMODORE-Rechnern mit Metallgehäuse angeschlossen wird,

wie z.B. die ältere Ausführung des CBM 8032. Wenn Sie die Datassette nie an eines dieser Geräte anschließen möchten, so schneiden Sie es einfach ab.

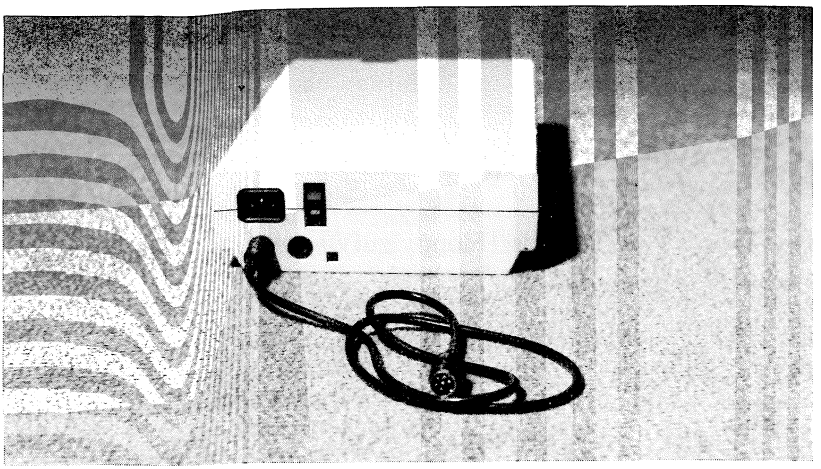
Das Diskettenlaufwerk

Die Diskettenlaufwerke (Floppylaufwerke) des Typs VC-1571 sind für viele Anwender nicht mehr wegzudenken. Wer einmal mit diesen externen Speichergeräten gearbeitet hat, wird sie nicht mehr missen können. Natürlich, ihr Preis übertrifft den Preis einer Datassette um das Vielfache, jedoch kommt ein ernsthafter Anwender des 128ers nicht um diese Laufwerke herum. Fast alle Programme, die nicht gerade dem Abschießen von UFOs dienen, nutzen die enormen Fähigkeiten der VC-1571 aus und sind somit nur auf Diskette erhältlich. Wer also Zugriff auf den gesamten Softwaremarkt des 128ers haben möchte, sollte sich gleich mit dem 128er zum Kauf dieses Laufwerkes entscheiden.

Für alle diejenigen, die ein solches Gerät besitzen, folgt nun die Anschlußbeschreibung. Die VC-1571 ist mit zwei Kabeln und drei Anschlußbuchsen versehen. Wozu denn nun zwei Kabel? Nun, da die Mechanik des Diskettenlaufwerkes sehr aufwendig ist und somit auch viel Strom benötigt, kann der 128er die Stromversorgung nicht übernehmen. Die VC-1571 ist deshalb mit einem eigenem Netzgerät versehen und wird an das 220 Volt-Netz angeschlossen. Dies übernimmt das beigefügte Netzkabel.

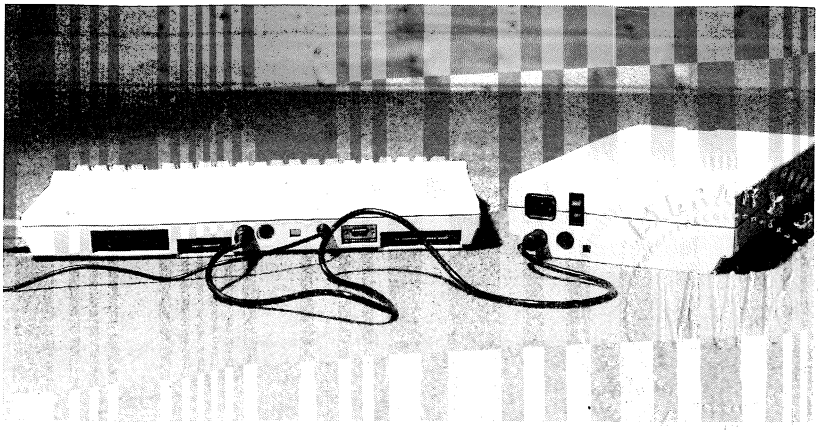
Das zweite Kabel, das einem Anschlußkabel für eine Stereoanlage sehr ähnlich sieht, verbindet das Laufwerk mit dem 128er. Alle Daten, die zwischen diesen beiden Geräten ausgetauscht werden, laufen hierüber.

Stellen wir zunächst die Verbindung 128er - VC-1571 her. Dazu stecken Sie eines der beiden Enden des schwarzen Kabels in eine der beiden dafür vorgesehenen Buchsen an der Rückseite des Floppylaufwerkes.

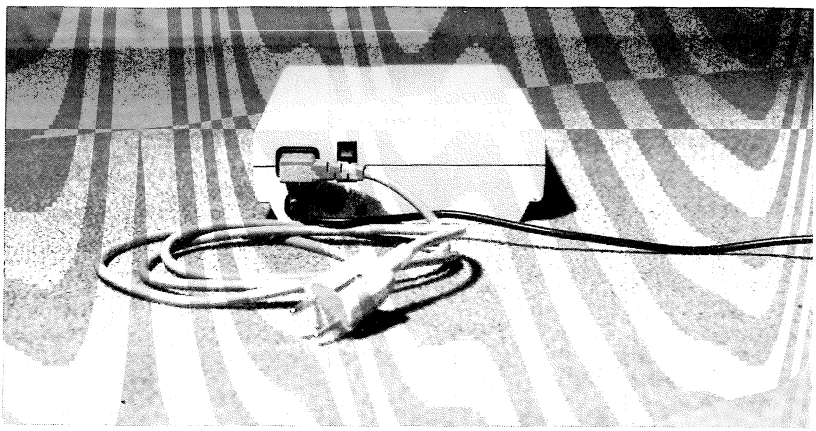


Die beiden Buchsen sind parallel geschaltet, das heißt, es ist ganz egal, welche der beiden Sie verwenden. Wozu die andere Buchse benötigt wird, werden Sie spätestens dann feststellen, wenn ein Drucker angeschlossen werden soll.

Das andere Ende des Floppykabels wird nun an den 128er angeschlossen. Dazu befindet sich eine entsprechende Buchse an der Rückseite Ihres Rechners. Hier ist jedoch Vorsicht geboten, da zwei ähnliche Buchsen nebeneinander angeordnet sind. Die andere Buchse ist für den Anschluß eines Monitors vorgesehen. Damit Sie nicht Ihre Floppy zum Monitor machen, passt das Floppykabel nicht in die andere Buchse. Wo das Kabel nun endgültig angeschlossen wird, entnehmen Sie dem folgenden Bild.



Das beigefügte, meist graufarbene Netzkabel schließen Sie hinten an der VC-1571 an. Die entsprechende Buchse ist unübersehbar.



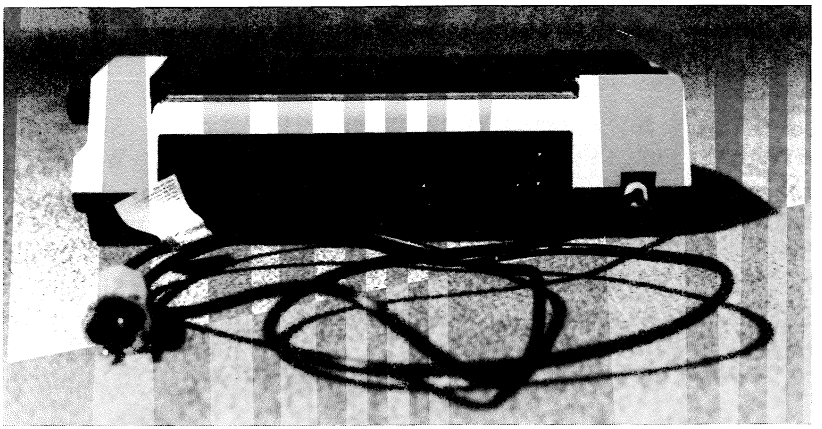
Übrigens sei hier noch vermerkt, daß Sie die Floppystation 1541 für den C 64 in derselben Weise an Ihren 128er anschließen können. Im Vergleich zur 1571 ist die 1541 wesentlich langsamer (7 - 8 mal), die Speicherkapazität ist nur halb so groß und die mechanische Ausführung ist schlechter. Wenn Sie aber Umsteiger vom C 64 sind und Ihre alte 1541 weiter benutzen wollen, so ist das ohne weiteres möglich.

Der Drucker

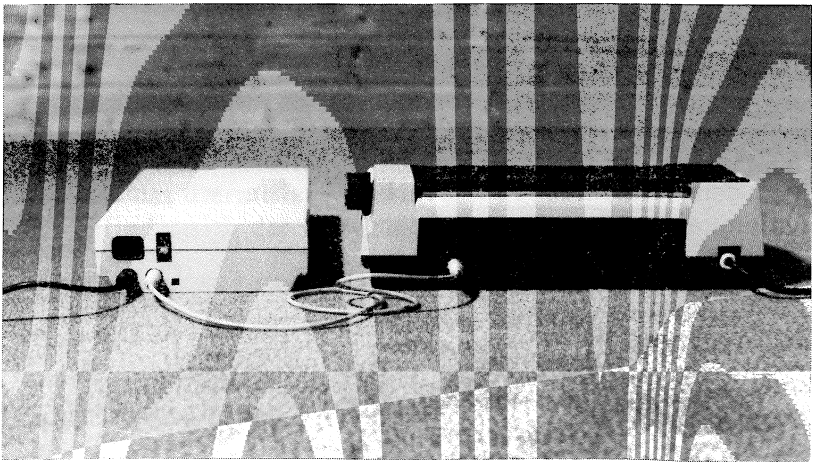
Zu einem professionellen Computersystem gehört unbedingt ein Drucker. Er ist z.B. ein nützliches Hilfsmittel während der Programmentwicklung. So sind ausgedruckte Programme wesentlich übersichtlicher als auf dem Bildschirm. Doch auch der Anwender, der den Computer sinnvoll einsetzen möchte, wie z.B. zur Textverarbeitung oder zur Dateiverwaltung, kann unmöglich auf einen Drucker verzichten.

COMMODORE bietet einen preislich auf den 128er zugeschnittenen Drucker, den MPS 801, an. Im folgenden wollen wir diesen Drucker an den Rechner anschließen. Mit einem Drucker werden wie bei der Floppystation meist zwei Kabel geliefert. Eins zur Stromversorgung und eins zur Datenübertragung vom Rechner zum Drucker. Dieses Kabel ist mit dem Kabel der Diskettenstation identisch. Es ist somit unerheblich, wenn Sie diese beiden Kabel vertauschen. Bei dem Drucker MPS 801 ist das Netzkabel am Drucker fest angeschlossen. Bei anderen Druckern sieht das Netzkabel oft genauso wie das zur Floppy aus.

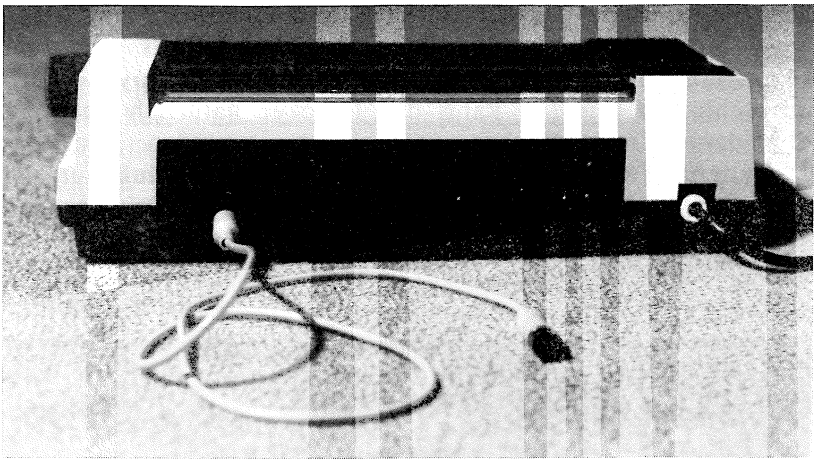
Schließen wir nun dieses Verbindungskabel an den Drucker an, der dafür hinten zwei Buchsen vorgesehen hat. Da auch diese beiden Anschlüsse wie bei der Floppystation parallel geschaltet sind, ist es egal, in welche Buchse das Kabel eingesteckt wird.



Das andere Ende des Kabels kann nun aber nicht am 128er angeschlossen werden, da dieser Anschluß bereits von dem Floppylaufwerk belegt ist. Sollte kein Floppylaufwerk angeschlossen sein, so kann der Drucker natürlich auch an die dann freie Buchse angeschlossen werden. Da das Diskettenlaufwerk aber noch einen freien Anschluß für ein weiteres Gerät hat, schließen wir hier den Drucker an. Übrigens, hier kann auch eine zweite Floppystation angeschlossen werden.



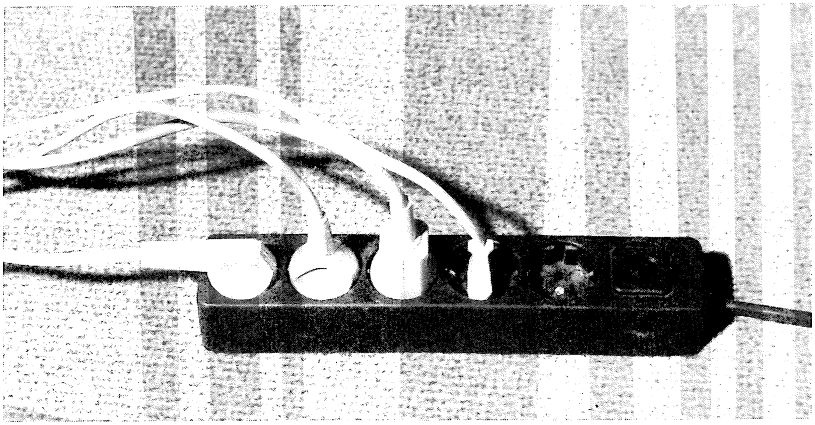
Auch der Drucker benötigt seine eigene Stromversorgung. Das dafür vorgesehene Netzkabel, das dem der Diskettenstation ähnlich ist, wird hinten am Drucker angeschlossen.



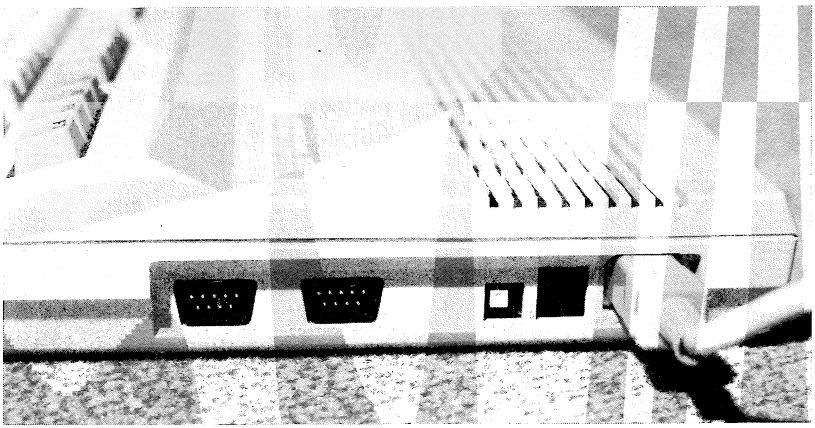
Mit dem Anschluß des Druckers haben wir unser recht umfangreiches System vervollständigt.

Die Inbetriebnahme

Nachdem alle Geräte angeschlossen sind, müssen diese zunächst mit Strom versorgt werden. Ärgerlich ist meist, wenn Sie erst jetzt bemerken, daß diese vielen Netzstecker nicht an eine Steckdose angeschlossen werden können. Gar in Verzweiflung gerät man, wenn keine Mehrfachsteckdose im Hause ist. Eine solche Mehrfachsteckdose ist unbedingt erforderlich. Kaufen Sie keine mit drei Abzweigungen, da diese meist nicht ausreicht. Sehr praktisch sind solche, auf denen ein Schalter angebracht ist, mit dem man die gesamten Geräte mit einem Knopfdruck ein- bzw. ausschalten kann. Diese Vielfachsteckdosen erhält man im Elektrofachhandel.



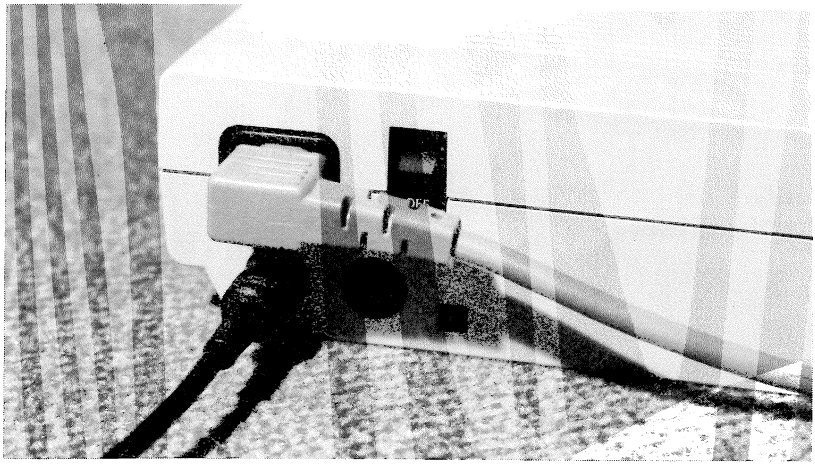
Wenn nun alle Geräte mit Strom versorgt sind, kann es losgehen. Wir schalten alle Geräte nacheinander ein. Zunächst das Herz des Systems, den 128er. Auf der rechten Seite befindet sich der Netzschalter.



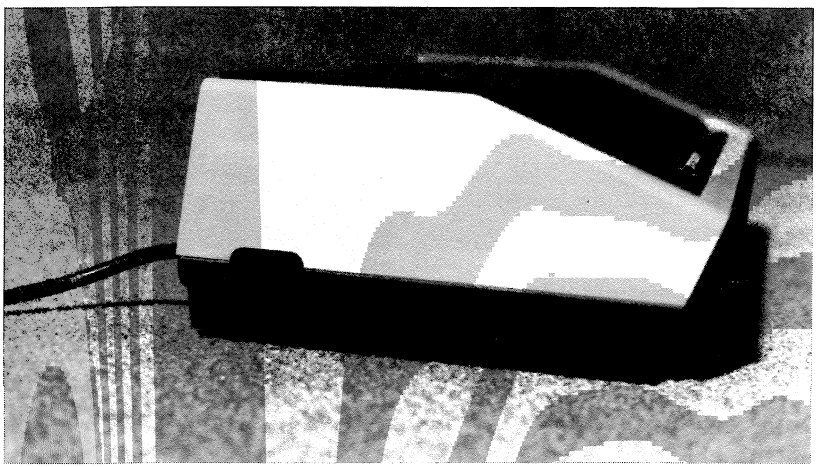
Nach Einschalten des 128ers muß die rote Leuchtdiode oben links auf dem Gerät die Betriebsbereitschaft signalisieren.

Nun erwecken wir die Diskettenstation zum Leben. Der entsprechende Schalter befindet sich auf der Rückseite des Geräts. Die grüne Leuchtdiode vorne am Gerät zeigt

die Betriebsbereitschaft an.



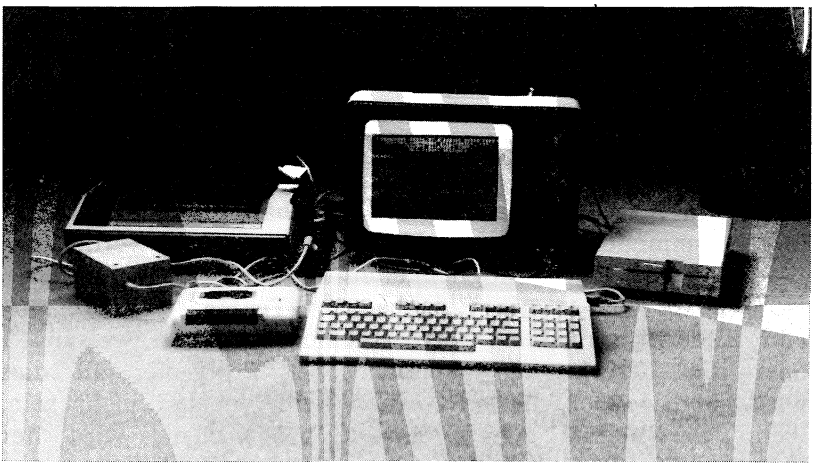
Schließlich muß nur noch der Drucker aktiviert werden. Der Netzschalter hierzu befindet sich auf der rechten Seite des Gerätes. Nach dem Einschalten bewegt sich der Druckkopf kurz hin und her. Danach leuchtet vorne am Gerät ein rotes Licht. Achtung! Wenn Sie den Drucker gerade ausgepackt haben, müssen Sie zunächst die Druckkopfsperre lösen (siehe Handbuch).



Um endgültig mit den ersten Experimenten beginnen zu können, muß noch der Fernseher eingeschaltet und

abgeglichen werden. Sie stellen den Fernseher auf den 128er so ein, als würden Sie einen Fernsehsender suchen. Es ist der Kanal 36 auf UHF. Falls gerade auf diesem Kanal in Ihrer Nähe ein Sender liegt, so können Sie von diesem Kanal abweichen. Dazu befindet sich hinten neben dem Fernsehkabelanschluß ein kleiner Trimmer, wo mit einem Schraubendreher die gewünschte Abweichung eingestellt wird.

Wenn Sie nun alle vorhandenen Geräte angeschlossen und eingeschaltet haben, ergibt sich etwa folgendes Bild:



Wer soweit gekommen ist, dem öffnet sich nun die Tür zu der faszinierenden Welt des Computers.

DIE TASTATUR

In diesem Kapitel lernen Sie die Handhabung der 92 Tasten Ihres Commodore 128. Mit Hilfe dieser Tasten werden Sie alle verborgenen Möglichkeiten Ihres Heimcomputers erschließen. Sei es die Eingabe von Texten, das Erstellen von Bildern (Grafiken), das Abschicken von Anweisungen oder die Berechnung mathematischer Probleme, all dies wird für Sie nach diesem Kapitel kein Neuland mehr sein. Wenn Sie einmal ALLE Tasten beherrschen, wird der Commodore 128 Ihnen ein folgsamer Partner bei der Lösung vieler Probleme sein.

Das Beherrschen der Tastatur dient nicht nur Computer-Freaks, die einmal jedes Bit des Rechners mit dem Vornamen kennen möchten, sondern auch denen, die ihren Computer nur mit den reichhaltig angebotenen Fertigprogrammen "füttern" wollen. Jemand, der sich zu den zuletzt genannten Anwendern zählt, darf bei einer Meldung eines Standardprogramms wie z.B. "Druecken sie die CLEAR/HOME-Taste zum Ausdruck des Bildschirminhalts" nicht verzweifelt in seinem Handbuch wühlen.

Kurzum sollte jeder, der auch nur gelegentlich an dem Commodore 128 arbeitet, mit der Tastatur vertraut sein.

Der Vater eines computerfaszinierten Sohnes z.B. sollte zumindest wissen, wie das beste Spielprogramm des Sohnemanns geladen wird, wenn der gerade Fußball spielt.

Doch keine Angst, Sie müssen sich nicht vorher zu einem Schreibmaschinenkursus an der Volkshochschule anmelden. Die meisten auch noch so erfahrenen Freizeitprogrammierer arbeiten mit dem "Zweifinger-Suchsystem". Wer eine gewisse Zeit mit der Tastatur vertraut ist, wird sich bald darüber wundern, wie flink er über die Tasten saust.

In diesem Kapitel werden die oben beschriebenen Hilfstasten besonders ausführlich beschrieben. Diese Tasten sind sehr wichtig, da damit Farben bestimmt, Texte eingegeben, Programme unterbrochen und Befehle an den Rechner übermittelt werden können.

Allgemeines zur Tastatur

Auf dem ersten Blick erweckt die Tastatur des Commodore 128 den Eindruck einer üblichen Schreibmaschinentastatur. Doch wenn Sie genauer hinsehen, werden Sie leichte Abweichungen feststellen:

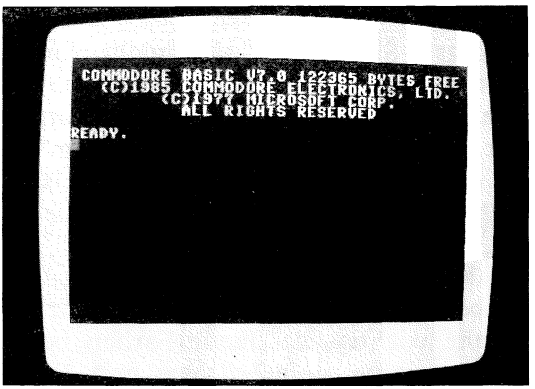
- Die Buchstaben 'Y' und 'Z' sind gemäß der amerikanischen ASCII-Norm vertauscht.

- Auf einigen Tasten der Tastatur sind die Umlaute (ö,ä,ü), das scharfe 's' (ß) und einige andere Zeichen farblich abgesetzt neben anderen Zeichen aufgedruckt.

- Es gibt viele zusätzliche Tasten (Hilfstasten), deren Funktionen später erläutert werden.

Sie sollten keine Experimente mit der Tastatur machen, bevor Sie nicht mit deren Funktion vertraut sind. Zwar können durch Fehlbedienungen keine Rauchwolken aus dem Rechner aufsteigen, doch Sie können durch überraschende Mißerfolge verblüfft werden. Warten Sie also geduldig auf die praktische Einweisung der nächsten Seiten.

Los geht's



Nun wird es Zeit, die schlafende Technik Ihres CBM 128 zum Leben zu erwecken. Schalten Sie dazu den Rechner ein. Nach jedem Einschalten erscheint eine Meldung, die besagt, daß der Rechner nun bereit ist, Ihren Kommandos zu folgen. Die erste Zeile der Meldung weist auf die interne BASIC-Version hin. Der COMMODORE 128 ist mit dem COMMODORE BASIC Version 7 ausgerüstet. Das ist die leistungsfähigste Version die bisher auf einem COMMODORE-Rechner zu finden ist.

Die erste Zeile der Einschaltmeldung informiert Sie auch darüber, daß Ihnen 122365 Bytes zur BASIC Programmierung zur Verfügung stehen. Das ist eine ganze Menge, die von einem Programm allein meist nicht ausgenutzt wird. Oft wird diese enorme Speicherkapazität zur rechnerinternen Datenspeicherung benutzt. D.h., die vom Programm zu verarbeitenden Daten befinden sich komplett im Speicher.

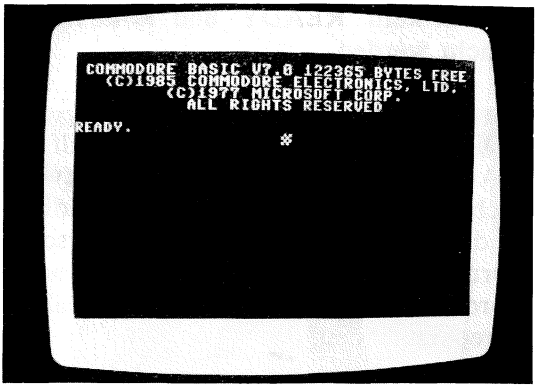
Die folgenden Zeilen informieren Sie darüber, daß die Rechte für Betriebssystem und BASIC bei COMMODORE und MICROSOFT liegen. In der letzten Zeile besagt die Meldung 'READY', daß das Betriebssystem nun Kommandos erwartet. Diese READY-Meldung signalisiert, daß Ihnen der Rechner zur Verfügung steht. Während des Programmablaufs

erscheint demnach kein 'READY' und es können somit keine Eingaben gemacht werden.

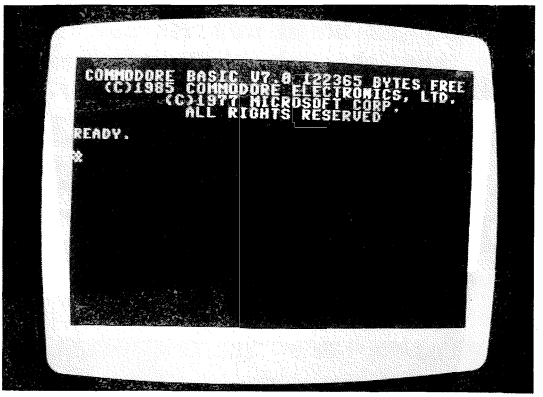
Nun zu dem kleinen blinkenden Quadrat unterhalb des 'READY'. Dies ist eine Orientierunghilfe auf dem Bildschirm. Angenommen, Sie möchten etwas auf dem Bildschirm schreiben, dann ist es notwendig, genau zu wissen, wo das einzugebende Zeichen erscheinen wird. Diese Markierung, die man in der Fachsprache CURSOR ("körsa" gesprochen) nennt, hilft Ihnen also, sich am Bildschirm zurechtzufinden.

CURSOR LINKS/RECHTS - Taste

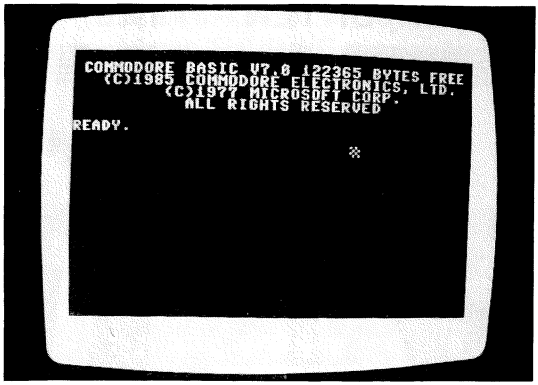
Diese Taste befindet sich unten rechts auf der Tastatur. Sie ist beschriftet mit der Abkürzung für Cursor (CRSR) und den Pfeilen nach rechts und links. Mit dieser Taste können Sie nun den CURSOR auf dem Bildschirm nach rechts oder links verschieben. Betätigen Sie nun diese Taste 20 mal, so wandert der Cursor 20 Stellen nach rechts und blinkt wieder geduldig unterhalb dem 'S' von 'RIGHTS'.



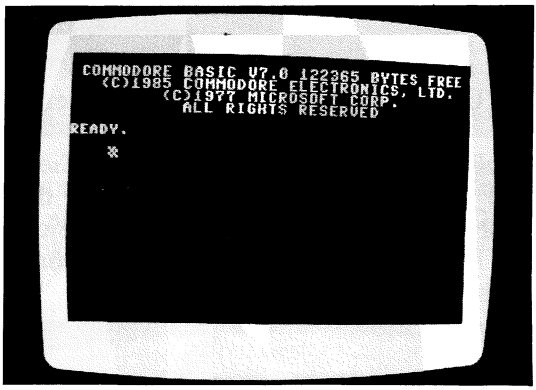
Was aber, wenn der Cursor den rechten Rand dieser Zeile überschreitet? Probieren Sie dies selbst aus und drücken dazu nochmals 20 mal die CURSOR LINKS/RECHTS - Taste. Nach dem 20. Druck auf diese Taste erscheint der Cursor wieder auf der 1. Spalte der folgenden Zeile.



Es ist doch recht umständlich, 20 mal auf die Cursor-Taste zu hämmern, um in die Bildschirmmitte zu gelangen. Diese Prozedur können Sie etwas vereinfachen. Wenn Sie die Taste gedrückt halten, so wandert der Cursor selbständig mit 15 Zeichen je Sekunde nach rechts. Probieren Sie dies aus, und halten Sie dazu die CURSOR LINKS/RECHTS - Taste gedrückt. Beobachten Sie, wie der Cursor zum rechten Bildschirmrand sprintet.



Sicher haben Sie sich gefragt, wie der Cursor denn nun in die andere Richtung (nach links) zu bewegen ist. Dazu benutzen Sie die Taste 'SHIFT'. Diese Taste schaltet die Funktion der Cursor-Taste um auf CURSOR LINKS. Halten Sie nun die SHIFT-Taste gedrückt und betätigen die CURSOR LINKS/RECHTS - Taste. Der Cursor bewegt sich also nun in die andere Richtung.



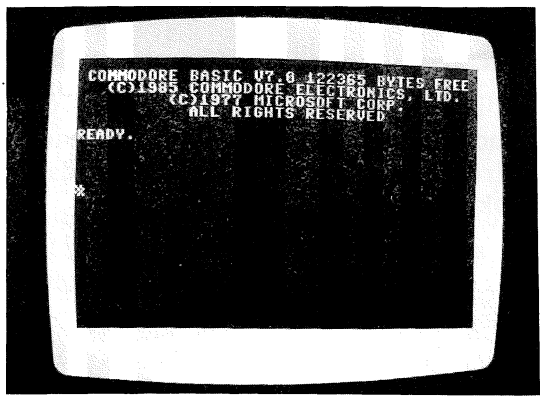
Selbstverständlich bewegt sich der Cursor auch selbständig nach links, ohne daß immer wieder einzeln auf die Taste gedrückt werden muß. Halten Sie dazu die beiden Tasten gedrückt.

CURSOR OBEN/UNTEN - Taste

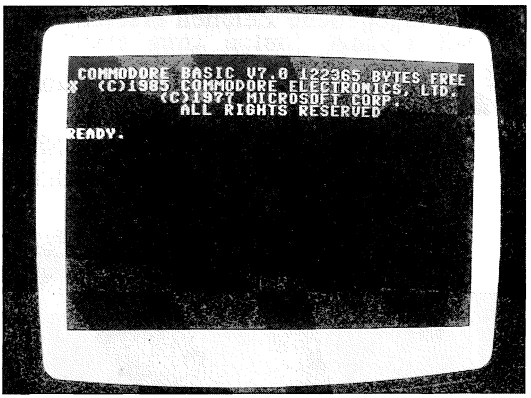
Wie kann man nun den Cursor wieder in die Ausgangsstellung (unterhalb 'READY') bringen? Dazu könnten Sie z.B. solange die CURSOR LINKS - Taste gedrückt halten, bis der Cursor Zeile für Zeile nach oben wandert. Doch dies ist nicht im Sinne des Erfinders. Die CURSOR OBEN/UNTEN - Taste ermöglicht Ihnen die ebenso schnelle Bewegung des Cursors nach oben und unten wie mit der CURSOR LINKS/RECHTS - Taste nach links und rechts.

Bewegen Sie nun den Cursor zum linken Bildschirmrand (SHIFT und CURSOR LINKS/RECHTS drücken).

Der Cursor blinkt wiederum geduldig, bis Sie ihn an eine andere Stelle bewegen. Peilen Sie nun die CURSOR OBEN/UNTEN - Taste an und drücken Sie diese Taste dreimal. Sie haben bemerkt, daß der Cursor drei Zeilen nach unten gesprungen ist.



Versuchen Sie nun, den Cursor in die 3. Bildschirmzeile zu bewegen. Dazu müssen Sie den Cursor nach oben bewegen. Wenn Sie aber auf die CURSOR OBEN/UNTEN - Taste drücken, so bewegt sich der Cursor nach unten! Sicher haben Sie bereits geschaltet und Rückschlüsse aus der zuvor beschriebenen Taste gezogen. Hier wurde mit Hilfe der SHIFT - Taste der Cursor nach links bewegt. Genauso können Sie mit der SHIFT - Taste und der CURSOR OBEN/UNTEN - Taste den Cursor nach oben bewegen. Versuchen Sie es nun! Drücken Sie solange SHIFT und CURSOR OBEN/UNTEN, bis der Cursor in der dritten Bildschirmzeile blinkt.



Auch hier wandert der Cursor selbständig Zeile für Zeile nach oben oder nach unten, wenn Sie die CURSOR OBEN/UNTEN - Taste gedrückt halten.

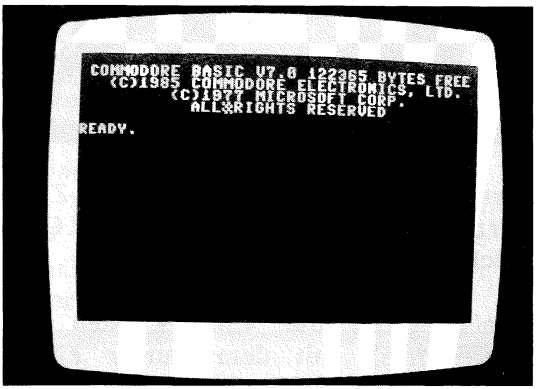
Um ein Gefühl für die Cursor-Tasten zu erlangen, versuchen Sie einmal, mit dem Cursor ein unsichtbares Quadrat zu zeichnen. D.h. Sie bewegen den Cursor rechts / unten / links / oben oder anders herum links / unten / rechts / oben. Das Beherrschen der Cursor- Tasten ist erstrebenswert, da Sie später z.B. Korrekturen in einem Programm blitzschnell vornehmen können. Doch Übung macht den Meister, und der ist bekanntlich noch nicht vom Himmel gefallen.
Anzumerken bleibt noch, daß die vier Tasten mit jeweils einem Pfeil der obersten Tastaturreihe dieselbe Funktion haben, wie die 'CRSR'-Tasten. Probieren Sie das aus.

Editieren mit den Cursor-Tasten

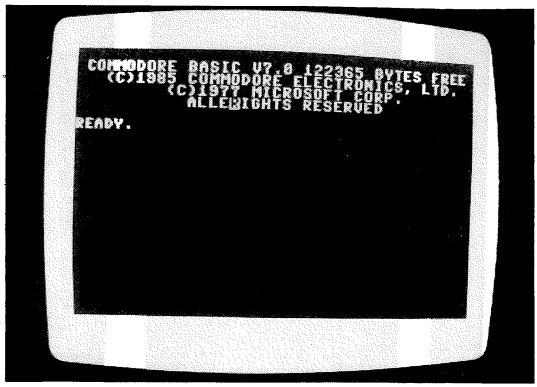
Hier taucht ein Fremdwort aus der Datenverarbeitung auf, dessen Bedeutung dem einen oder anderen Leser, besonders dem Anfänger, sicher nicht bekannt ist. Editieren bedeutet Bearbeiten von Texten, d.h., Texte erstellen und ändern. Man spricht bereits vom Editieren, wenn Sie ein paar Worte auf dem Bildschirm schreiben.

Mit den Cursortasten können Sie einen bereits auf dem Bildschirm enthaltenen Text ändern. Wenn Sie den Cursor auf ein Zeichen des Bildschirminhalts positionieren und eine beliebige Taste drücken, so wird dieses Zeichen durch das zuletzt eingegebene Zeichen ersetzt.

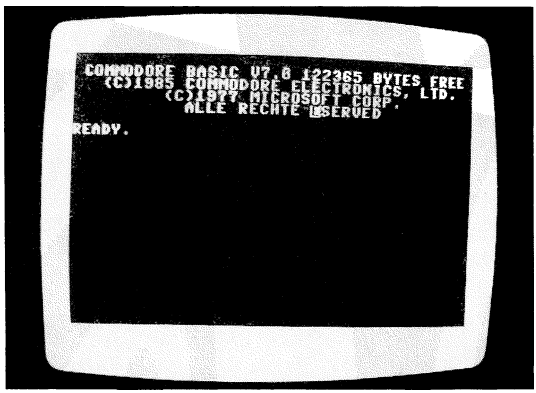
Nun können wir z.B. die Einschaltmeldung verändern. Ändern wir die vierte Zeile der Einschaltmeldung um in "ALLE RECHTE VORBEHALTEN". Bringen Sie zunächst den Cursor auf den ersten zu ändernden Buchstaben (hinter das letzte 'L' von 'ALL').



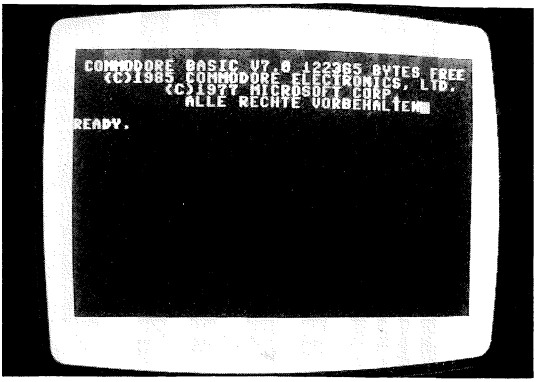
An dieser Stelle soll nun ein 'E' angefügt werden. Nichts leichter als das! Sie drücken einfach die Taste 'E', und aus 'ALL' ist 'ALLE' geworden. Der Cursor ist selbständig ein Zeichen nach rechts auf das 'R' von 'RIGHTS' gerückt. Das ist ja auch klar, sonst müssten Sie bei der Texteingabe nach jedem Zeichen den Cursor manuell weiterstellen.



An die Stelle des 'R' schreiben Sie ein Leerzeichen (die unübersehbare Taste ganz unten). Ändern Sie nun auch die fünf letzten Buchstaben dieses Wortes ('IGHTS') in 'RECHTE' um. Dazu geben Sie einfach unmittelbar hintereinander diese sechs Buchstaben ein. Nun haben wir das Wort 'RIGHTS' in 'RECHTE' umgewandelt.

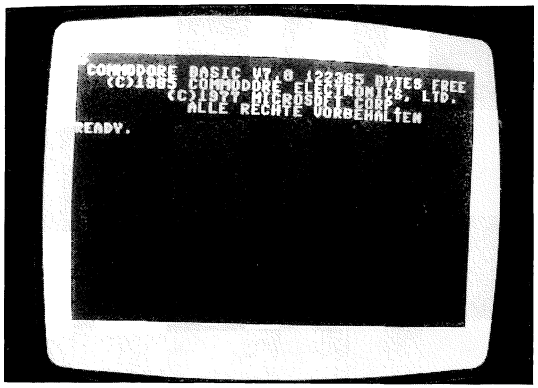


Der Abschluß dieser kleinen Übung ist für Sie sicher kein Rätsel mehr. Sie überschreiben den Rest der Zeile rücksichtslos mit dem Wort 'VORBEHALTEN'.

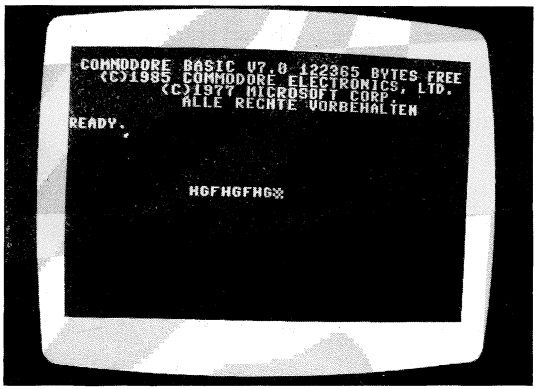


CLR/HOME - Taste

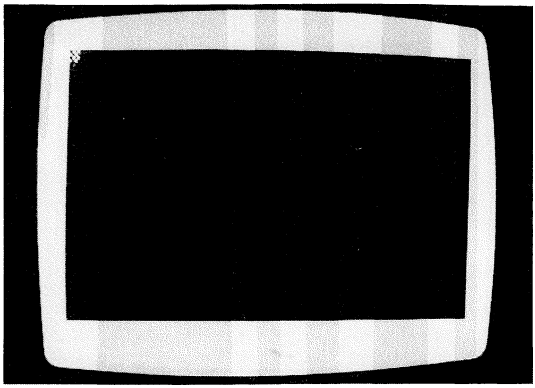
Wir haben uns bereits damit beschäftigt, den Cursor so schnell wie möglich zum oberen Bildschrmrand zu manövrieren. Dazu haben wir die CURSOR OBEN/UNTEN - Taste benutzt. Aber es geht noch schneller: Die CLR/HOME - Taste hat zwei Funktionen, eine SHIFT- und eine Normalfunktion. Drücken Sie diese Taste ohne SHIFT, so wird die HOME-Funktion aktiv. Der Cursor wird dann in die obere, linke Ecke positioniert. Diese Ecke ist das "Zuhause" des Cursors. Überzeugen Sie sich selbst davon und drücken nun die CLR/HOME - Taste.



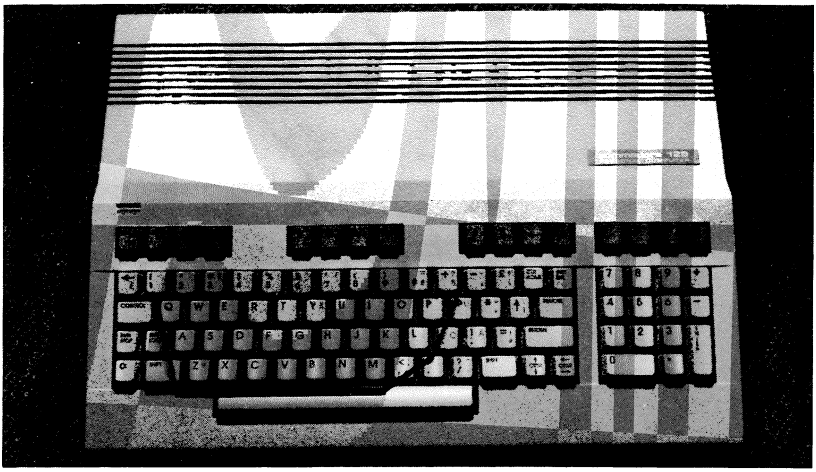
Damit Sie nicht aus der Übung kommen: Schreiben Sie ein paar Buchstaben in die Bildschirmmitte.



Wenn Sie nun denken, daß das, was sich nun auf dem Bildschirm befindet, eigentlich alles unbrauchbares Kauderwelsch ist, so wischen Sie einfach alles weg. HALT!! nicht mit dem Fensterleder, damit können Sie vielleicht den Staub vom Bildschirm wischen, nicht jedoch die eingetippten und somit im Rechner gespeicherten Daten des Bildschrims. Schließlich ist es für einen Rechner, der ca. 1 Million Befehle pro Sekunde abarbeiten kann, ein Kinderspiel, "mal eben" die 1000 Zeichen auf dem Bildschirm auf Tastendruck zu löschen. Drücken Sie nun die Tasten SHIFT und CLR/HOME gemeinsam, so wird der Bildschirm blitzschnell gelöscht. Der Cursor blinkt dann ganz verlassen "zuhause", in der oberen, linken Ecke des Bildschirms.



Die Tasten mit Buchstaben

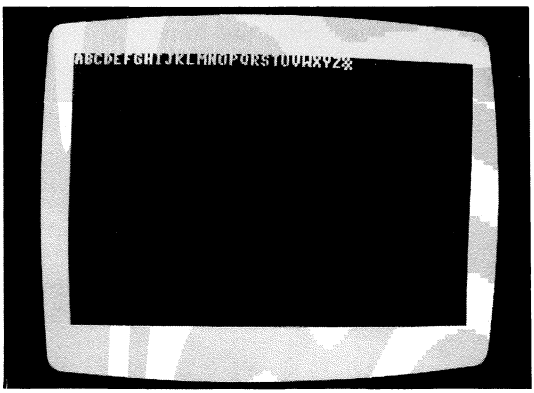


Wie bereits erwähnt, gleicht die Anordnung dieser Tasten bis auf kleine Abweichungen der einer Schreibmaschine. Sicher sind Ihnen die jeweils zwei Grafiksymbole an der Vorderseite jeder dieser Tasten sofort aufgefallen. Doch was es damit auf sich hat, stellen wir zunächst zurück.

Konzentrieren wir uns auf die Buchstaben. Wenn Sie irgendeine dieser Tasten drücken, so erscheint der entsprechende Großbuchstabe an der Cursor-Position auf dem Bildschirm. Bevor wir diese Tasten einsetzen, löschen Sie bitte den Bildschirm mit den Tasten SHIFT und CLR/HOME.

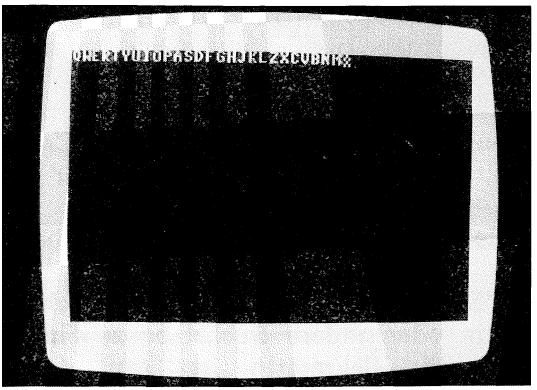
Nun haben wir einen "sauberen" Bildschirm, auf dem wir editieren können. Bitte geben Sie nun keine Unmengen von Texten ein in der Hoffnung, daß Ihr Computer alles, zu jeder Zeit abrufbereit, in seinen Speicher aufnimmt. Ganz so einfach ist es nicht. Das, was Sie eingeben, wird lediglich im flüchtigen Bildschirmspeicher festgehalten. Wie man eingegebene Texte abspeichern kann, erfahren Sie in unserer BASIC-Einführung.

Vorerst gehen wir davon aus, daß die 'CAPS LOCK / ASCII DIN'- Taste nicht gedrückt ist. Wir kommen auf diese Taste noch zu sprechen.
Schreiben Sie jetzt einmal alle Buchstaben auf dem Bildschirm, vielleicht sogar in alphabetischer Reihenfolge. Sie werden feststellen, wie schwer doch zu Anfang der ein oder andere Buchstabe zu finden ist, wenn man nicht gerade Sekretärin ist. Auf dem Bildschirm befinden sich nun alle verfügbaren Großbuchstaben.

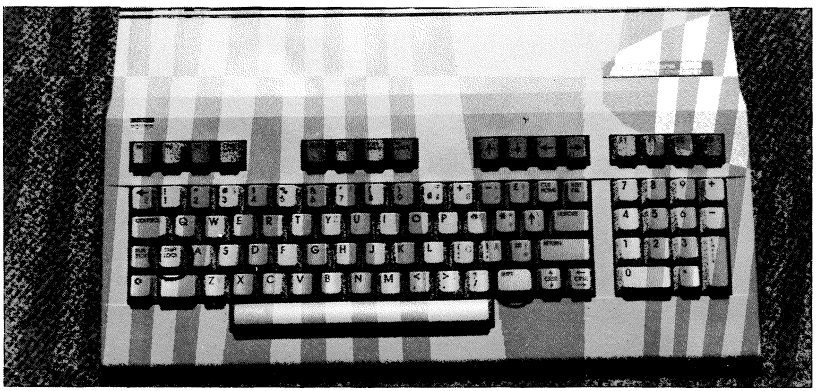


Löschen Sie wieder den Bildschirm und geben nochmals alle Buchstaben ein. Doch nun nicht in alphabetischer

Reihenfolge, sondern wie sie auf der Tastatur angeordnet sind. Also zunächst die obere Reihe (Q bis P), dann die mittlere Reihe (A bis L) und schließlich die untere Reihe (Z bis M). Sicher fällt Ihnen diese Reihenfolge viel leichter.



Die SHIFT- und SHIFT LOCK - Tasten

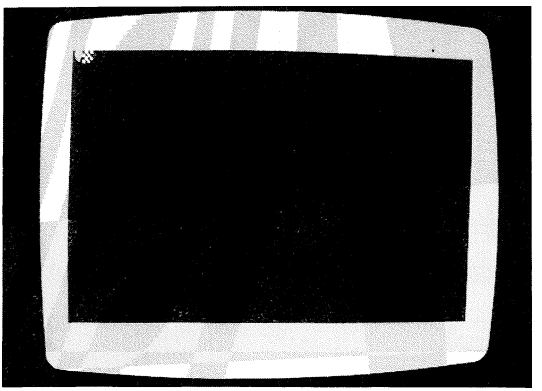


Die SHIFT-Taste haben Sie bereits kennengelernt. Sie benutzten diese Taste zum Umschalten der mit zwei Funktionen belegten Tasten. Zusätzlich ist der COMMODORE 128 mit einer zweiten SHIFT-Taste ausgerüstet (SHIFT LOCK), die bei Betätigung einrastet. Somit ist es überflüssig, die SHIFT-Taste gedrückt zu halten. Betätigen Sie diese Taste ein zweites Mal, so wird sie wieder entriegelt.

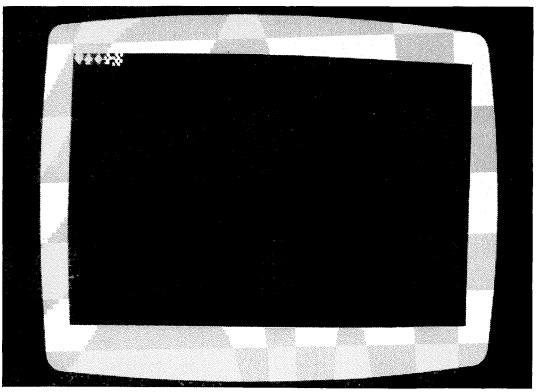
Fast jede Taste ist mit mehr als einem Zeichen oder einer Funktion belegt. Mit der SHIFT-Taste erreichen Sie die

zweite Funktion bzw. das zweite Zeichen einer mehrfach belegten Taste. Bei den Tasten, die zusätzlich mit zwei Grafiksymbolen beschriftet sind, wählen Sie mit der SHIFT-Taste das jeweils rechte Grafikzeichen aus.

Versuchen Sie nun, die vier Spielkartensymbole der Tasten A,S,Z und X auf den Bildschirm zu bekommen. Löschen Sie zuvor den Bildschirm mit den beiden Tasten SHIFT und CLR/HOME. Das Herz z.B. erscheint auf dem Bildschirm, wenn Sie die Taste SHIFT gedrückt halten und dann die Taste 'S' betätigen.

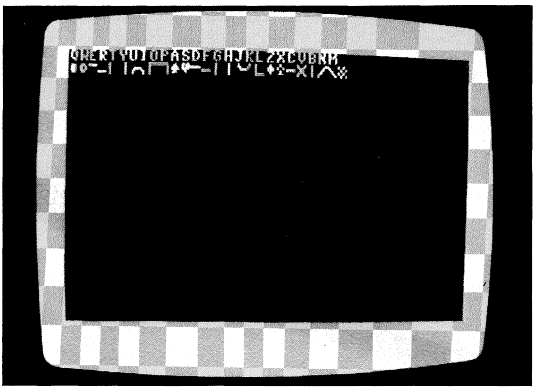


Sicher ist es für Sie nun ein Kinderspiel, die anderen drei Symbole direkt neben dem einsamen Herzen auf den Bildschirm zu zaubern.



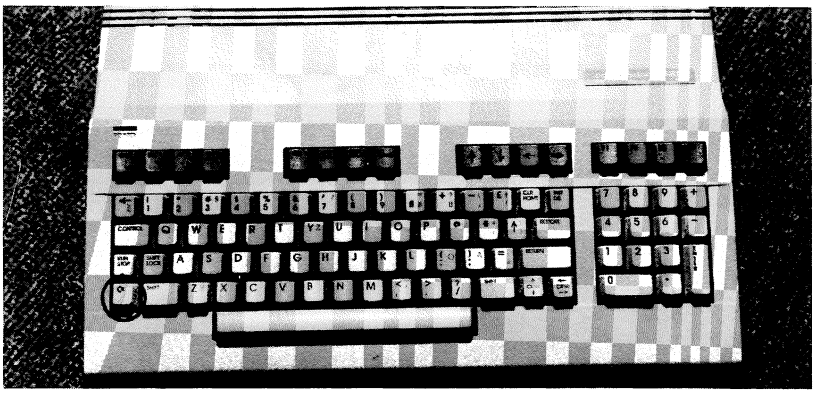
Dies war noch relativ einfach. Aber nun eine etwas schwierigere Aufgabe: Stellen Sie nochmals alle

Buchstaben wie sie auf der Tastatur angeordnet sind dar (vorher Bildschirm löschen). Dann positionieren Sie den Cursor auf den Anfang der nächsten Zeile und schreiben unter jeden Buchstaben das jeweils dazugehörige Grafikzeichen. Ein Tip: Wenn Sie die SHIFT LOCK-Taste einrasten, haben Sie bei der Eingabe der Grafikzeichen eine Hand frei, womit Sie z.B. zur Kaffeetasse greifen können.



Entriegeln Sie bitte die SHIFT LOCK-Taste, wenn sie verwendet wurde.

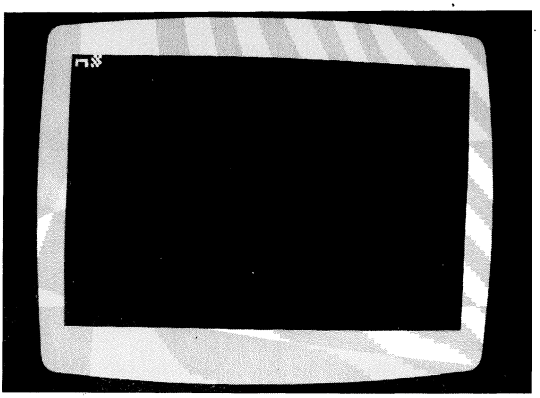
Die COMMODORE - Taste 'C='



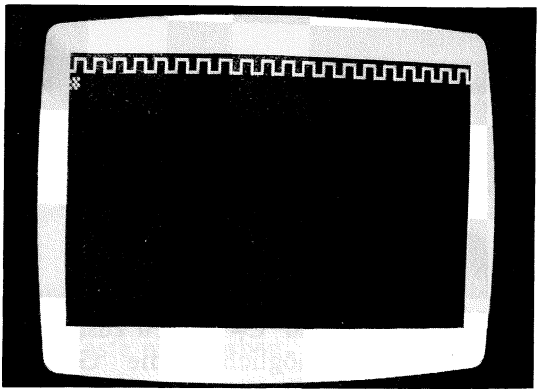
Mehrfachbelegte Tasten sind für Sie nicht mehr unbekannt. Sie haben mit SHIFT z.B. das rechte Grafiksymbol eines Buchstabens zum Vorschein gebracht. Sie wissen sicher, worauf hier angespielt wird. Da die Buchstaben - Tasten zwei Grafiksymbole darstellen können, muß natürlich auch

eine Möglichkeit gegeben werden, das zweite Grafiksymbol anzusprechen. Dazu wurde diese Taste installiert. Halten Sie die Taste 'C=' gedrückt, wenn Sie eines der links an der Vorderseite der Tasten dargestellten Grafiksymbole auswählen möchten.

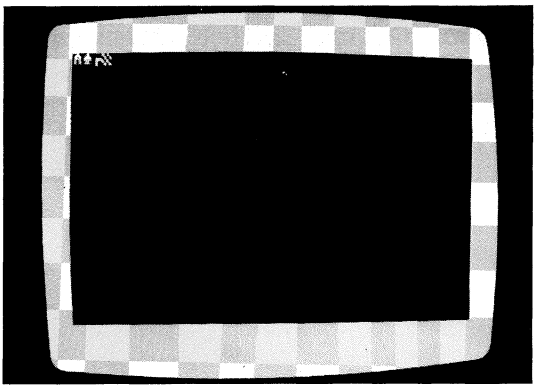
Nun aufgepasst! Links neben den schon bekannten Spielkarten - Symbolen sind Grafikzeichen angebracht, die jeweils einen Winkel darstellen. Halten Sie nun die Taste 'C=' gedrückt und stellen die zwei Winkel der Tasten 'A' und 'S' auf dem Bildschirm dar. Bitte löschen Sie auch hier wieder vorher den Bildschirm.



Nun eine etwas schwierigere Aufgabe: Versuchen Sie das in Bild 2 dargestellte Muster nachzuzeichnen. Nein, nicht mit Bleistift und Papier, sondern mit den Tasten Ihres Rechners auf dem Bildschirm. Ein Tip: Dieses Muster besteht aus zwei Zeilen!



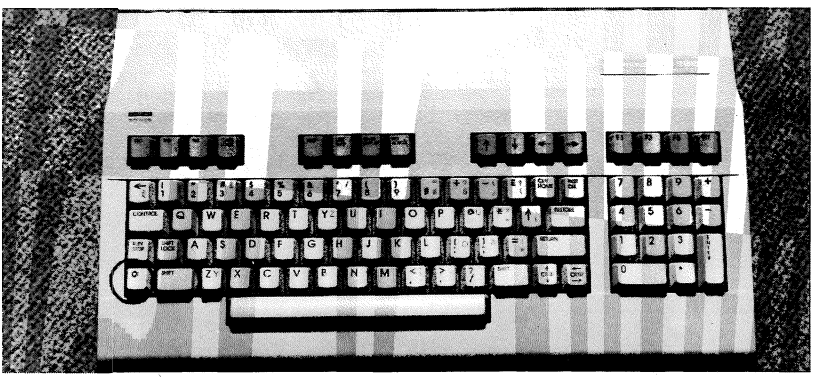
Schreiben Sie nun die drei mit der Taste 'A' darstellbaren Zeichen nebeneinander auf dem Bildschirm. Zuerst den Buchstaben, dann das rechte und schließlich das linke Grafikzeichen. Löschen Sie bitte vorher den Bildschirm.



Mit einiger Übung können Sie allein mit den Grafikzeichen hübsche Bilder auf den Bildschirm zaubern. Vielleicht starten Sie in Ihrer Familie einen Wettbewerb nach dem Motto "wer tastet das schönste Bild auf den Bildschirm?".

Diese einfarbigen Bilder können Sie auch noch in 16 Farben darstellen, doch dazu erfahren Sie an späterer Stelle mehr.

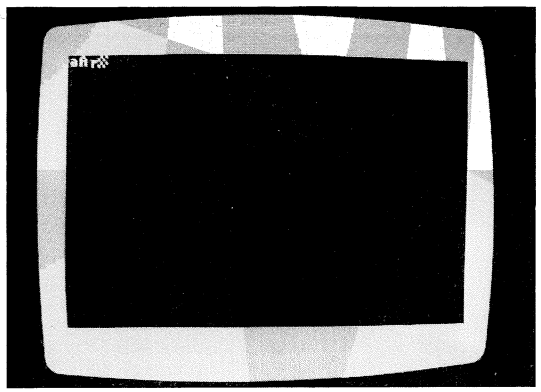
Der Textmodus



Die beiden Tasten SHIFT und 'C=' haben zusammen noch eine weitere Aufgabe: Sie ermöglichen die Groß/Kleinschrei-

bung. In diesem Modus gleicht die Tastatur noch mehr einer Schreibmaschine. Bei normaler Betätigung werden Kleinbuchstaben und mit Hilfe der Umschalt-Taste (bei uns die SHIFT - Taste) Großbuchstaben angezeigt. Die Grafiksymbole, die normalerweise mit SHIFT angesprochen werden, stehen nun nicht mehr zur Verfügung.

Sie schalten den Textmodus ein, indem Sie bei gedrückter SHIFT- Taste einmal die 'C='- Taste gleichzeitig drücken. Wollen Sie wieder im Normalmodus arbeiten, so drücken Sie nochmals diese beiden Tasten. Schalten Sie nun den Textmodus ein und beobachten, was aus den drei Zeichen der letzten Übung geworden ist. Sie werden sofort dem Textmodus angepasst. Aus dem großem 'A' wurde ein kleines 'A', aus dem PIK wurde ein großes 'A' und das zweite Sonderzeichen ist erhalten geblieben.

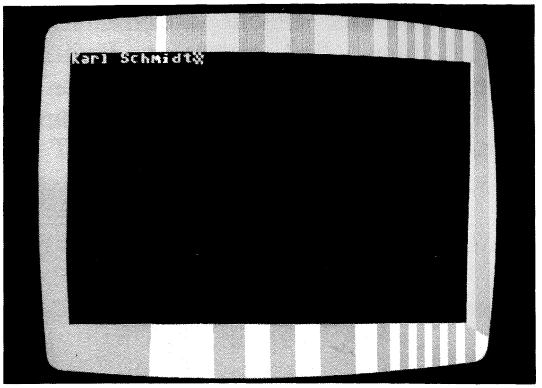


Es gibt also zwei verschiedene Möglichkeiten, die Tastatur zu nutzen:

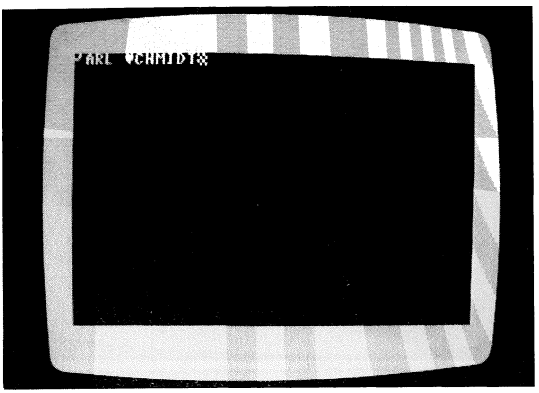
1. Den Groß/Grafik-Modus
2. Den Groß/Klein-Modus

Wollen Sie Kleinbuchstaben darstellen, so müssen Sie auf die Grafikzeichen, die mit SHIFT angesprochen werden, verzichten und umgekehrt.

Schreiben Sie nun einmal im Textmodus den Namen "Karl Schmidt" auf dem Bildschirm. Die Großbuchstaben werden mit der SHIFT-Taste eingegeben.



Was meinen Sie, tut sich auf dem Bildschirm, wenn Sie nun den Textmodus wieder ausschalten? Überzeugen Sie sich selbst. Schalten Sie den Textmodus wieder aus, indem Sie wiederum SHIFT und 'C=' gleichzeitig drücken.



Aus den Großbuchstaben wurden wieder Grafikzeichen und aus den Kleinbuchstaben wurden Großbuchstaben.

In der Praxis hat sich erwiesen, daß man im Textmodus besser arbeiten kann. Der Text erscheint in Kleinbuchstaben nicht so gedrängt wie in Großbuchstaben. Programmlisten z.B. sind wesentlich übersichtlicher, wenn sie im Textmodus auf dem Bildschirm angezeigt werden. Ein ständig eingeschalteter Textmodus ist selbstverständlich nur geeignet, wenn ohne Grafikzeichen gearbeitet wird.

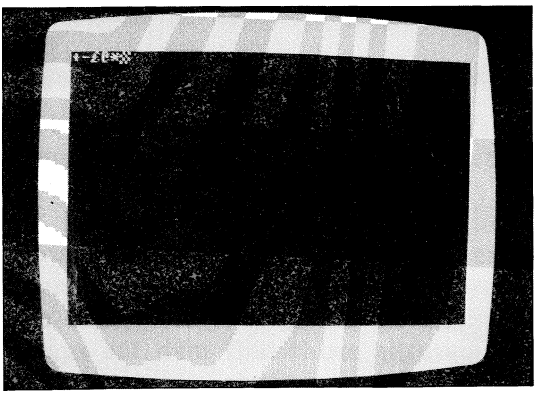
Die ASCII DIN- Taste

Ist die 'CAPS LOCK / ASCII DIN'- Taste gedrückt, so erscheint bei betätigen einer Taste, soweit vorhanden, das graue Zeichen. Drücken Sie also die Taste mit dem schwarzen '+' Zeichen, so erscheint auf dem Bildschirm ein 'ß' so, wie es auf der Taste grau abgebildet ist.
Durch die gedrückte 'CAPS LOCK / ASCII DIN'- Taste läßt sich die Tastatur exakt der deutschen DIN Norm einer Schreibmaschinentastatur anpassen. Beachten Sie dabei auch das mit dem 'Z' vertauschte 'Y'.
Sie können durch Betätigen der 'CAPS LOCK / ASCII DIN'- Taste beliebig zwischen den Zeichensätzen hin- und herschalten. Probieren Sie daß einmal aus.
Wir gehen im folgenden allerdings weiter davon aus, daß die 'CAPS LOCK / ASCII DIN'- Taste nicht gedrückt ist.

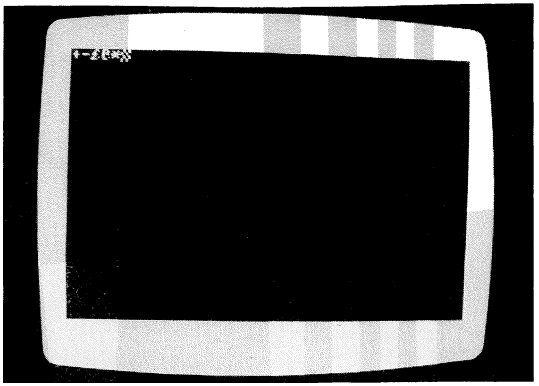
Weitere Grafiktasten +, -, , und *

Außer den vielen Grafikzeichen auf den Buchstaben - Tasten finden Sie weitere Grafikzeichen auf diesen fünf Tasten. Sie werden entsprechend mit den Tasten SHIFT und 'C=' angesprochen. Die Tasten +, -, , und * haben jedoch eine Besonderheit: Sie sind sowohl im Textmodus als auch im Normalmodus identisch.

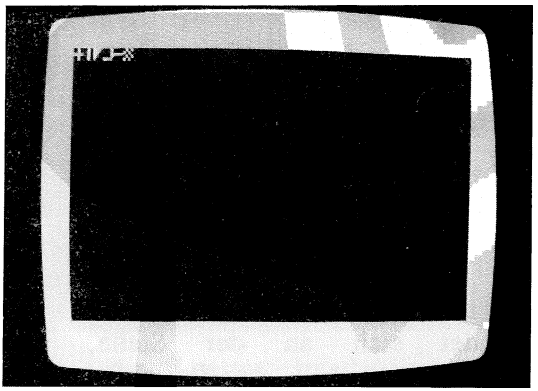
Überzeugen Sie sich davon und geben Sie die Zeichenfolge '+- *' ein, nachdem Sie den Bildschirm gelöscht haben.



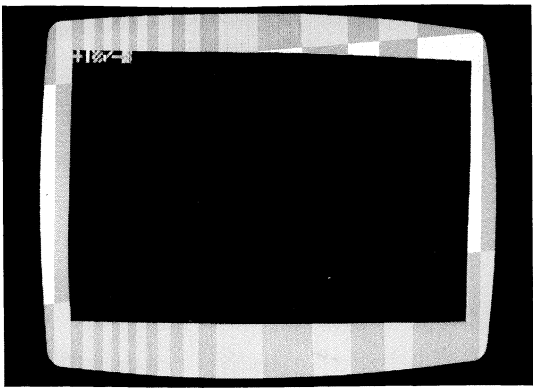
Schalten Sie nun den Textmodus ein, so werden Sie feststellen, daß die Zeichen unverändert geblieben sind.



Verlassen Sie wieder den Textmodus (wiederum mit SHIFT und 'C='). Im Textmodus werden bekanntlich aus den SHIFT-Grafikzeichen Großbuchstaben. Da auch die hier behandelten fünf Tasten SHIFT-Grafikzeichen enthalten, geben wir diese Zeichen ein, um festzustellen, ob der Textmodus hier etwas verändert. Löschen Sie also nochmals den Bildschirm und betätigen diese fünf Tasten, diesmal zusammen mit SHIFT.



Wenn Sie nun den Textmodus einschalten, werden Sie feststellen, daß zwei Tasten im Textmodus andere Grafikzeichen zum Vorschein bringen. Dies sind zwei Zeichen, die Sie auf der Tastatur zwar nicht finden, die aber im Textmodus plötzlich erscheinen.



Schalten Sie den Textmodus bitte aus, bevor Sie fortfahren.

Die Leertaste

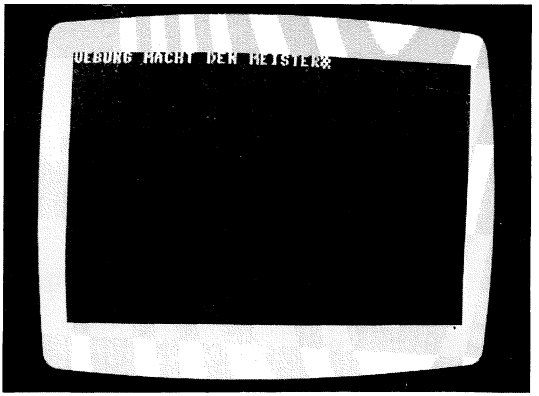
Auch diese Taste finden Sie an jeder Schreibmaschine wieder. Bei Ihrem Computer brauchen Sie ebenfalls nicht darauf zu verzichten, Leerzeichen zwischen die Wörter zu setzen. Befindet sich an der Stelle, an der ein Leerzeichen erscheinen soll, bereits ein anderes Zeichen, so wird dieses natürlich gelöscht.

Löschen Sie den Bildschirm und geben den folgenden Satz ein:

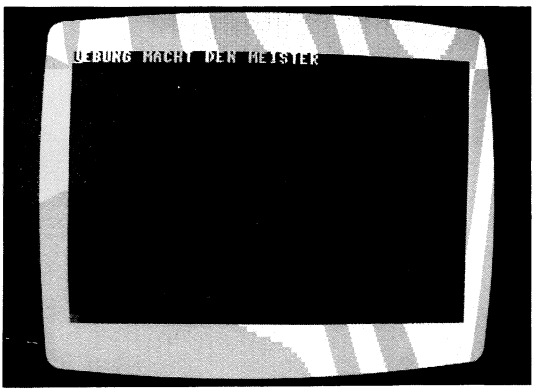
UEBUNG MACHT DEN MEISTER

Hier müssen Sie zwischen den einzelnen Wörtern die Leertaste drücken. Sie können auch den Cursor um eine

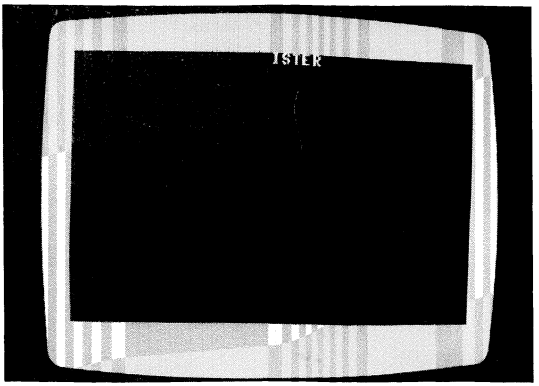
Stelle nach rechts rücken, doch dies ist sehr umständlich und somit nicht angebracht.



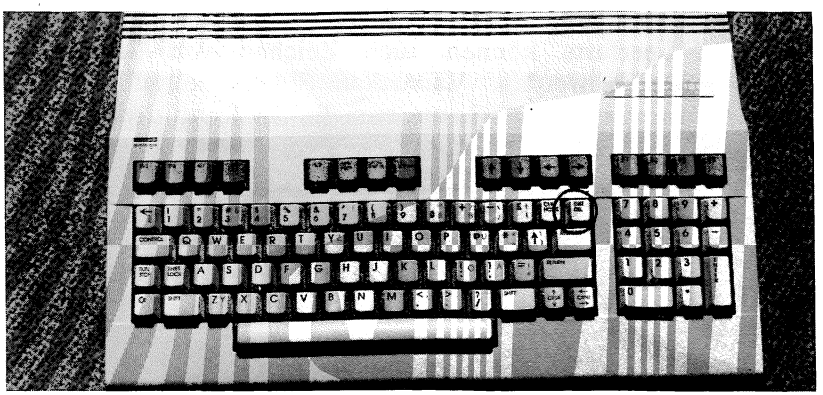
Mit der Leertaste können auch Zeichen von links nach rechts gelöscht werden. Versuchen Sie es selbst: Drücken Sie die Taste CLR/HOME, um zum linken Rand der obersten Bildschirmzeile zu gelangen.



Halten Sie nun die Leertaste gedrückt, so werden alle eingegebenen Zeichen durch Leerzeichen ersetzt, also gelöscht.



Die INST/DEL - Taste



Die Abkürzungen dieser Taste sagen bereits viel über die Funktion aus. DEL bedeuted DELETE (löschen) und INST bedeuted INSERT (einsetzen). Sie verfügen somit über zwei komfortable Hilfen zum Editieren von Texten.

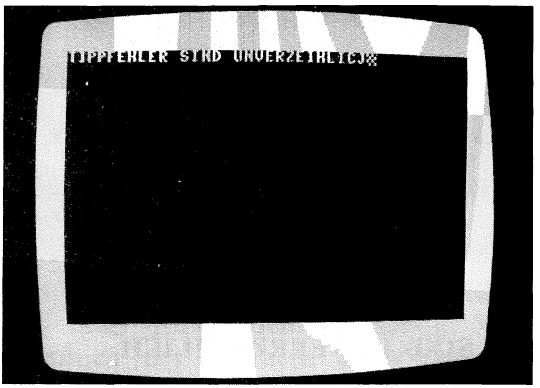
DEL - LÖSCHEN

Ohne SHIFT ist die Funktion DEL aktiv. Im letzten Beispiel haben wir bereits mit der Leertaste Zeichen auf dem Bildschirm gelöscht. Doch dies war nur in eine Richtung - nach rechts - möglich. Mit DEL können nun Zeichen links vom Cursor gelöscht werden. Dies ist besonders hilfreich beim Korrigieren von Tippfehlern. Ein Druck auf die Taste INST/DEL und das falsch eingetippte Zeichen ist wieder gelöscht.

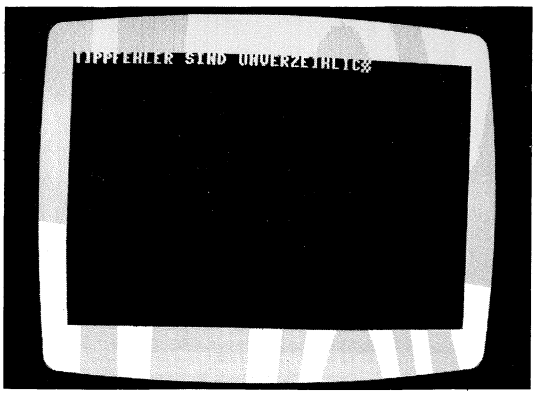
Löschen Sie nun den Bildschirm und tippen den folgenden Satz ab:

TIPPFEHLER SIND UNVERZEIHLICJ

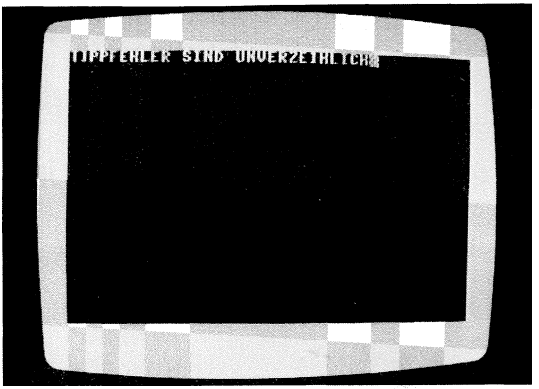
Bei der Eingabe des letzten Buchstabens ist ein Fehler unterlaufen.



Tipp-Ex ist hier nicht notwendig. Auf Tastendruck wird der zuletzt eingegebene Buchstabe gelöscht. Drücken Sie dazu die Taste INST/DEL



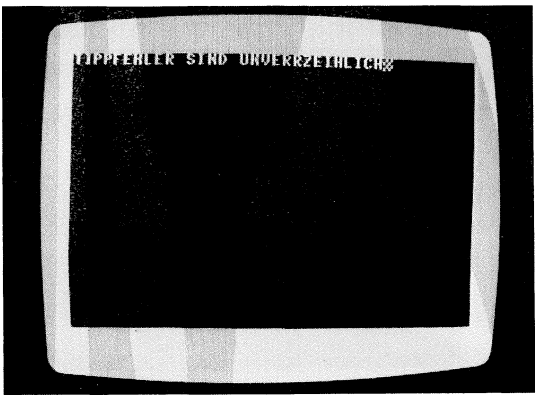
Nun können Sie hier den richtigen Buchstaben einsetzen, und der Tippfehler ist vergessen.



Doch dieses Beispiel zeigt nicht den überragenden Vorteil dieser Taste. Spielen wir ein weiteres Beispiel durch. Geben Sie, nachdem Sie den Bildschirm gelöscht haben, den folgenden Satz ein:

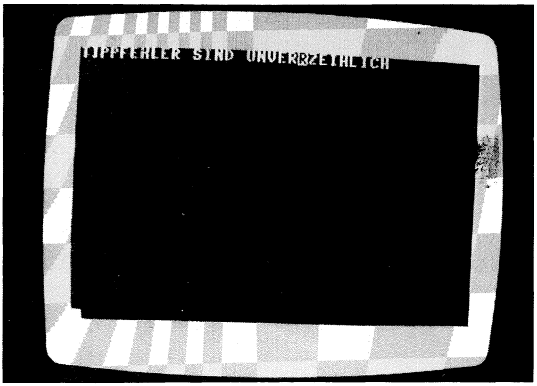
TIPPFEHLER SIND UNVERRZEIHLICH

Auch hier hat sich ein Tippfehler eingeschlichen: Zwei 'R' sind hier nicht angebracht.

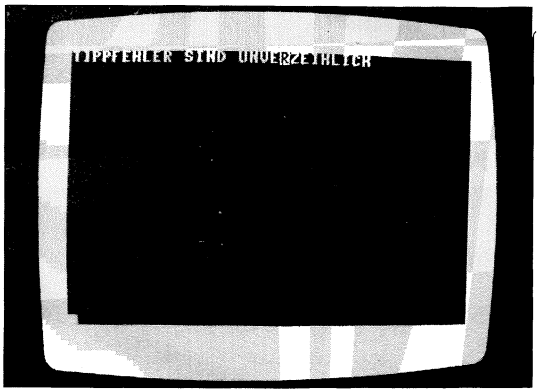


Es ist nicht notwendig, das letzte Wort ab dem Tippfehler nochmal zu schreiben. Wird nämlich ein Zeichen links vom Cursor mit der Taste INST/DEL gelöscht, so werden automatisch alle Zeichen in dieser Zeile, die sich rechts vom Cursor befinden, nachgerückt. Das ist nicht einfach zu erklären, sehen Sie darum selbst:

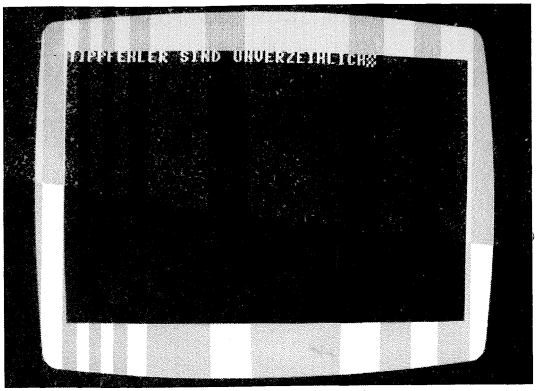
Bewegen Sie den Cursor auf das zu löschende 'R' des fehlerhaften Satzes. Dazu drücken Sie die Tasten SHIFT und CURSOR LINKS/RECHTS gleichzeitig. 1:48-*



Wenn Sie nun die Taste INST/DEL (ohne SHIFT) drücken, wird das 'R' links neben dem Cursor gelöscht und der Rest, einschließlich des Zeichens unter dem Cursor, nach links aufgerückt.



Wenn Sie nun den Cursor zum Ende dieses Satzes bringen, können Sie mit dem Schreiben fortfahren. Halten Sie dazu solange die CURSOR LINKS/RECHTS - Taste gedrückt, bis der richtige Punkt erreicht ist.

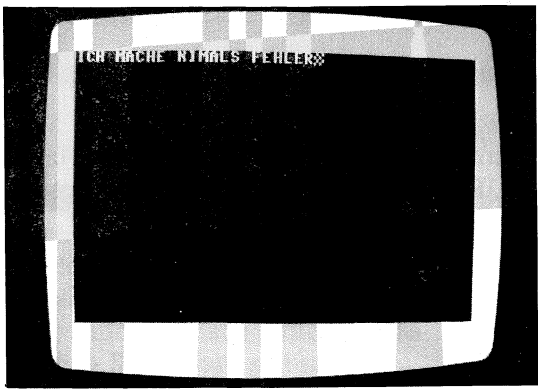


INST - einfügen

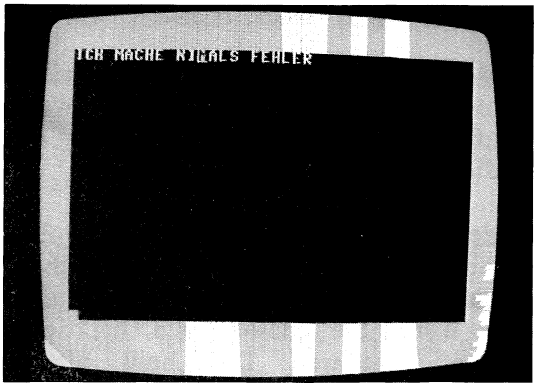
Die INST/DEL - Taste hat wie gesagt zwei Funktionen. Einmal - wie bereits beschrieben - die DEL-Funktion, mit der Sie Zeichen links vom Cursor löschen können. Die zweite Funktion wird wie üblich mit der SHIFT-Taste aktiviert. Die Einfügefunktion Ihres COMMODORE 128 ist sehr beliebt, doch überzeugen Sie sich selbst an dem folgenden Beispiel.
Geben Sie zunächst den folgenden Satz ein, nachdem Sie den Bildschirm gelöscht haben:

ICH MACHE NIMALS FEHLER

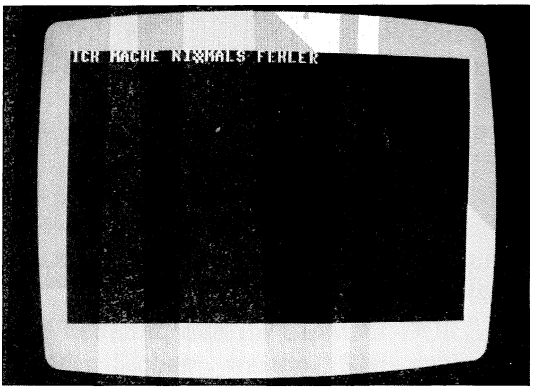
Hierin ist ein Fehler enthalten, der mit der INST/DEL - Taste wieder korrigiert werden kann.



Jedem, der in der Schule ein wenig aufgepaßt hat, wird dieser Fehler sicher sofort ins Auge gefallen sein. Es wäre nun sehr umständlich, wenn man alle Zeichen bis zum Fehler löschen und neu eingeben müßte. Da gerade diese Einfügefunktion sehr oft benötigt wird, ist Sie mit den Tasten SHIFT und INST/DEL sofort abrufbereit. Bewegen Sie dazu den Cursor hinter das Zeichen, nach dem weitere Zeichen eingefügt werden sollen, in unserem Fall hinter das 'I', also auf das 'M'.



Beobachten Sie nun was passiert, wenn Sie die SHIFT-Taste gedrückt halten und die INST/DEL-Taste einmal betätigen. Es wird Platz für das einzufügende Zeichen geschafft, wobei der Cursor auf diesem Platz verbleibt.



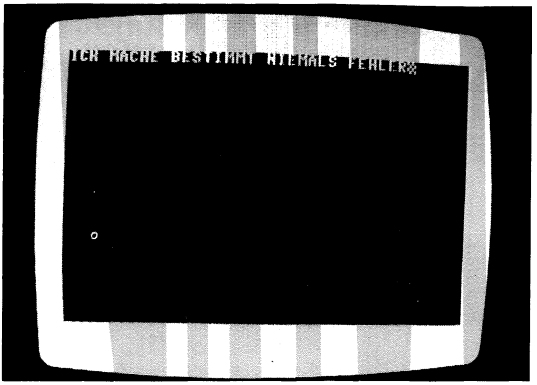
Nun geben Sie das gewünschte Zeichen ein, in unserem Beispiel also das fehlende 'E'. Wenn Sie den Cursor nun

nach rechts zum Ausgangspunkt bewegen, können Sie die Texteingabe fortsetzen.

Selbstverständlich können Sie auch mehrere Zeichen einfügen. Dazu drücken Sie an der entsprechenden Postition mehrmals die INST/DEL- mit der SHIFT-Taste. Wenn Sie beide Tasten gedrückt halten, tritt eine automatische Wiederholung in Kraft. Die Taste INST/DEL wird dann selbständig solange betätigt, bis Sie beide Tasten wieder loslassen.

Gerade die Bedienung der Taste INST/DEL bereitet am Anfang Schwierigkeiten. Doch Sie werden selbst feststellen, daß Sie mit der Zeit sehr viel Routine beim Umgang mit dieser Taste entwickeln werden.

Weil es gerade gut an diese Stelle paßt, wollen wir jetzt auch unsere erste ESCAPE-Funktion besprechen. Die linke Taste der oberen Tastaturreihe hat die Aufschrift 'ESC' für ESCAPE. Wenn Sie diese Taste betätigen, geschieht zunächst nichts sichtbares. Das heiß allerdings nicht, daß Ihr 128er diesen Tastendruck nicht bemerkt hätte. Vielmehr "weiß" Ihr Computer nun, daß die nächste gedrückte Taste für ein Kommando und nicht für ein auf dem Bildschirm abzubildendes Zeichen steht.
Betätigen Sie, nachdem Sie einmal die ESC-Taste betätigt haben, die Taste mit dem Buchstaben 'A'. Es geschieht wieder nichts sichtbares. Auch ist das 'A' nicht an der aktuellen Cursorposition wie gewohnt erschienen. Was ist los - reagiert der 128er jetzt nicht mehr auf die Tastatur? Keine Sorge, es wird sich sofort alles aufklären.
Fahren Sie jetzt bitte mit dem Cursor auf das 'N' von 'NIEMALS' und geben hintereinander das Wort 'BESTIMMT' ein. Wenn Sie alles richtig gemacht haben, steht jetzt auf Ihrem Bildschirm:



Die Tastenfolge 'ESC' und 'A' hat also offensichtlich bewirkt, daß neue Zeichen in eine Zeile eingefügt werden ohne die alten Zeichen wie bisher einfach zu überschreiben.
Mit der Tastenfolge 'ESC' und 'C' können Sie diesen Einfügemodus wieder ausschalten.
Der COMMODORE 128 kennt noch viele solcher ESCAPE-Funktionen. Im folgenden werden wir noch einige davon besprechen. Eine Übersicht über alle ESCAPE-Funktionen finden Sie im Hanbuch zu Ihrem COMMODORE 128 auf Seite 4-4.

Die CTRL-Taste bringt Farbe ins Spiel

Sicher ist Ihnen bekannt, daß Ihr COMMODORE 128 16 Farben darstellen kann. Sicher warten Sie auch schon lange darauf, diese selbst hervorzuzaubern. Nichts leichter als das. Jede der Zifferntasten 1 bis 8 schaltet zusammen mit der CTRL (Control) - Taste auf eine andere Zeichenfarbe

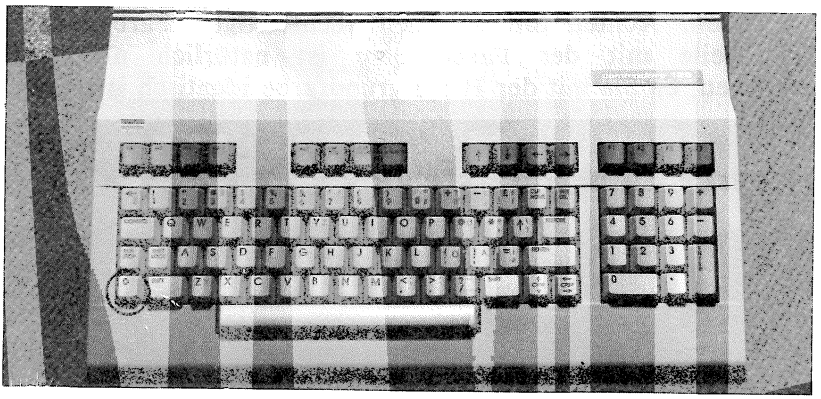
um. Halten Sie z.B. CTRL gedrückt und tippen auf die Taste '1', so schreiben Sie ab sofort in schwarzer Schrift. Die acht Tasten und die zugehörigen Farben:

Taste	Beschriftung	Farbe
1	BLK (black)	schwarz
2	WHT (white)	weiß
3	RED	rot
4	CYN (cyan)	türkis
5	PUR (purple)	violett
6	GRN (green)	grün
7	BLU (blue)	blau
8	YEL (yellow)	gelb

Wenn Sie nun alle Farben austesten, so werden Sie feststellen, daß die ein oder andere Farbe auf dem blauen Hintergrund schlecht lesbar ist. Man sagt, der Kontrast stimmt dann nicht. Doch dies ist nicht weiter tragisch, da Sie die Farbe des Hintergrunds mit einem BASIC-Befehl, den Sie an späterer Stelle noch kennenlernen werden, ändern können.

Probieren Sie nun einmal folgendes aus: Drücken Sie die CTRL-Taste zusammen mit der Taste '7'. Was passiert und wie ist dies zu erklären? Ganz einfach: Die Farbe der Taste '7' ist die Farbe blau und somit mit dem Hintergrund identisch. Der Cursor ist nun wie alle anderen Zeichen zwar vorhanden, aber unsichtbar. Man könnte nun sogar Befehle eingeben, die man aber auf dem Bildschirm nicht sieht. Innerhalb von Programmen nutzt man dies oft aus, um Meldungen des Rechners für den Benutzer unsichtbar zu machen.

Weitere Farbenpracht mit der C= -Taste



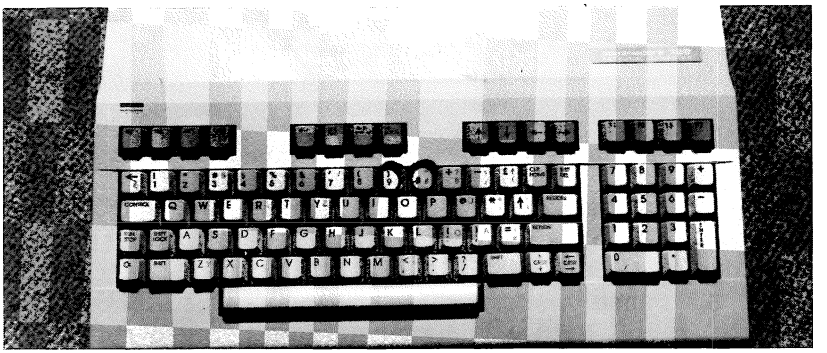
Viele vergleichbare Computer sind zumeist mit acht Farben ausgerüstet. Doch Ihr COMMODORE 128 gibt sich damit nicht zufrieden. Mit der COMMODORE-Taste und den Zifferntasten 1 bis 8 können weitere acht brillante Farben ausgewählt werden. Diese Farben sind aber auf den Tasten nicht ausgewiesen. Drücken Sie nun einmal die C= -Taste und die Taste '1' gleichzeitig. Der Cursor wechselt zur Farbe orange und alle Zeichen, die Sie nun eingeben, erscheinen ebenfalls in der Farbe orange. Der folgenden Tabelle können Sie alle 16 Farben und die dazu zu bedienenden Tasten entnehmen:

Taste	Beschriftung	CTRL	C=
1	BLK	schwarz	orange
2	WHT	weiß	braun
3	RED	rot	hellrot
4	CYN	türkis	grau 1
5	PUR	violett	grau 2
6	GRN	grün	hellgrün
7	BLU	blau	hellblau
8	YEL	gelb	grau 3

Das sind nun die 16 mit dem COMMMODORE 128 darstellbaren Farben. Mit zu den Farben gehören drei Grauabstufungen. Wenn Sie die verschiedenen Graustufen einmal auswählen und einige Zeichen damit schreiben, so werden Sie feststellen, daß GRAU 1 ein dunkeles Grau und GRAU 3 ein helles Grau ist.

Versuchen Sie nun einmal, die ersten 16 Zeilen des Bildschirms mit jeweils einer Farbe und dem Zeichen 'A' zu füllen. Achten Sie auf den rechtzeitigen Farbwechsel. Die Zeile mit der Farbe Blau ist natürlich nicht zu erkennen, da Sie mit der Hintergrundfarbe identisch ist.

Reverse Zeichendarstellung



Die deutsche Übersetzung für 'REVERSE' ("rivörs" gesprochen) ist sinngemäß 'UMGEKEHRT'. Doch was sind 'umgekehrte' Zeichen? Sicher haben Sie schon einmal Negative von Schwarz/Weiß-Fotos betrachtet und festgestellt, daß die Farben vertauscht sind. ein dunkler Hintergrund des Fotos ist auf dem Negativ hell.

Wenn Sie die Zeichen auf dem Bildschirm genau betrachten, so merken Sie, daß sie aus einzelnen Punkten zusammengesetzt sind. Alle Zeichen werden in einer Matrix von 8 mal 8 Bildschirmpunkten dargestellt. Der Buchstabe 'B' z.B. sieht dann etwa so aus:

```
 oooooo
  oo  oo
  oo  oo
  ooooo
  oo  oo
  oo  oo
 oooooo
```

Wenn dieses Zeichen nun reverse dargestellt wird, werden alle gesetzten Punkte der 8 mal 8-Matrix gelöscht und

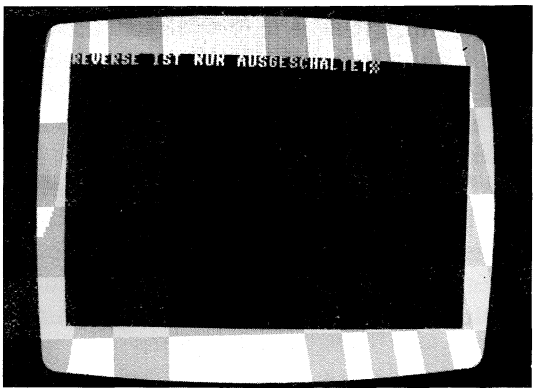
umgekehrt. Das 'B' sieht demnach dann so aus:

```
      oo
o   oo  o
o   oo  o
o       oo
o   oo  o
o   oo  o
      oo
oooooooo
```

Wenn Sie bei der Farbauswahl zufällig die Taste '9' zusammen mit CTRL betätigt haben, sind Sie schon unfreiwillig mit den reversen Zeichen konfrontiert worden. Die reverse Zeichendarstellung wird mit den Tasten 'CTRL' und '9' eingeschaltet.

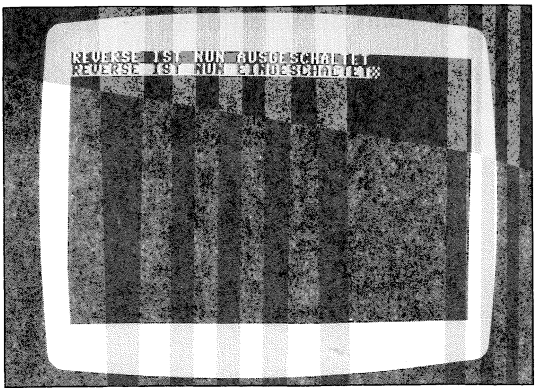
Löschen Sie nun den Bildschirm und schreiben in die erste Zeile den Text

REVERSE IST NUN AUSGESCHALTET



Nachdem Sie den Cursor zum Anfang der zweiten Zeile befördert haben, halten Sie bitte die Taste 'CTRL' gedrückt und betätigen kurz die Taste '9'. Schreiben Sie danach den folgenden Text auf den Bildschirm:

REVERSE IST NUN EINGESCHALTET



Jetzt sehen Sie bestens, was reverse Zeichendarstellung bedeutet. Natürlich muß dies auch wieder auszuschalten sein. Die Taste rechts neben der '9', also die Taste '0', schaltet den Reverse-Modus zusammen mit CTRL wieder aus. Halten Sie dazu die CTRL-Taste gedrückt und tippen kurz auf die Taste '0'. Nun können Sie in der gewohnten Art und Weise weiterschreiben.

Wozu benötigt man diese ungewohnte Zeichendarstellung? Eine berechtigte Frage, die jedoch leicht zu beantworten ist. Wenn Sie schon einmal Programme aus dem Handel gestartet und bedient haben, werden Sie sicher hier und da reverse Zeichen erkannt haben. Z.B. wird die letzte Bildschirmzeile gerne dazu benutzt, den Benutzer des Programms auf Fehler hinzuweisen. Diese Fehlermeldungen werden deshalb reverse ausgegeben, damit sie sofort ins Auge fallen.
Dies war nur eines von vielen Beispielen für reverse Zeichendarstellung.

FERTIGPROGRAME - LADEN UND STARTEN

Nicht jeder, der den COMMODORE 128 einsetzt, möchte einmal ein perfekter Programmierer werden und nach Möglichkeit alle Programme selbst schreiben. Natürlich finden viele einen Anreiz darin, den Computer zu beherrschen und mit ihm alles zu lösen, was man sich in den Kopf setzt. Doch wer den Rechner auch nur einige Stunden in Betrieb genommen hat, wird feststellen, daß er keine Wunder vollbringen kann. Ohne ein entsprechendes Programm ist der Rechner eigentlich "tot". Erst ausgeklügelte und meist in monatelanger Arbeit erstellte Programme erwecken ihn zum Leben.

Der Softwaremarkt des COMMODORE 128 ist für den C 64 Modus sehr umfangreich. Viele der angebotenen Programme sind bereits oder werden sicher in Kürze auch für den 128er Modus angeboten. Da das Angebot so reichhaltig ist, kann sich jeder Anwender ein Stückchen von dem riesigen Kuchen des Softwareangebots abschneiden.

Alles gut und schön. Sie haben nun ein Programm erworben und möchten es natürlich so schnell wie möglich zum Laufen bringen. Betriebssystem und BASIC-Interpreter sind für Sie Fremdwörter und sollen es auch bleiben. Dies ist durchaus verständlich. Warum soll der Anwender, der mit dem COMMODORE 128 lediglich eine Kartei (oder besser Datei) seiner 230 Kühe führen möchte, erst 6 Monate BASIC im Do-It-Yourself-Verfahren lernen. Immerhin sind die Anwendungsprogramme so gestaltet, daß sie von einem Nicht-Programmierer ohne Probleme bedient werden können.

Das folgende Kapitel gibt Aufschluß darüber, wie solche Standardprogramme geladen und gestartet werden.

Laden von Diskette

Die meisten Programme werden nur als Diskettenversion auf den Markt gebracht. Dafür gibt es viele Gründe. Z.B. ist es für einen Anwender bequem, wenn ein Programm schnell geladen und somit einsatzbereit ist.

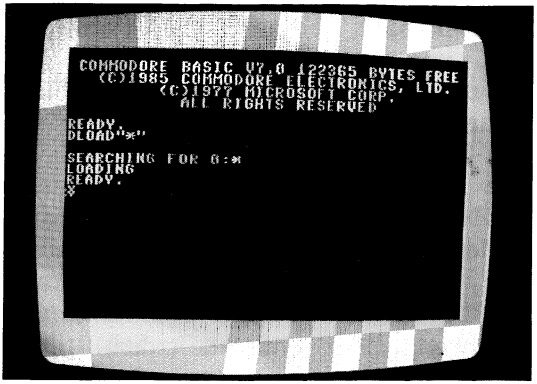
Darüber hinaus ist die Datenverwaltung mit einem Diskettenlaufwerk wesentlich komfortabler und schneller. Die Effektivität des Datenzugriffs nimmt großen Einfluß auf den gesamten Programmablauf. Wer möchte schon Minuten warten, bis eine Adresse in die entsprechende Datei aufgenommen wird?

Es gibt aber auch Programme, die einfach nicht mit einem Kassettenrecorder eingesetzt werden können. Kommerzielle Programme wie z.B. Dateiverwaltungen oder Finanzbuchhaltungen sind so umfangreich, daß sie nicht komplett im Hauptspeicher aufgenommen werden können. Sie bestehen aus mehreren Teilprogrammen, die bei Bedarf nachgeladen werden. Es wird also ständig direkt auf Diskette zugegriffen, was bei einer Kassettenspeicherung nicht möglich ist. Kassettenrekorder sind zwar zum Laden und Speichern von Spielen geeignet - ignoriert man die lange Wartezeit -, in der Praxis haben sich jedoch Diskettenlaufwerke als unverzichtbar erwiesen.

Wie laden Sie nun Programme von der Diskettenstation? Schalten Sie zunächst den Rechner und die Diskettenstation ein (evtl. noch den Fernseher bzw. Monitor). Die Reihenfolge ist hier egal. Übrigens: Eine Vielfachsteckdose mit Hauptschalter erspart Ihnen das mehrmalige Einschalten der vielen Geräte.

Nachdem alle Geräte eingeschaltet wurden, legen Sie die Programmdiskette ins Laufwerk und schließen die Klappe. Nun müssen Sie dem Rechner befehlen, das Programm zu laden. Bis auf wenige Ausnahmen werden alle Standardprogramme durch betätigen der 'RUN/STOP'-Taste bei gedrückter 'SHIFT'-Taste von Diskette geladen und gestartet.

Ein korrekter Ladevorgang sieht wie folgt aus:



Nach Eingabe des Ladebefehls meldet der Rechner mit "SEARCHING FOR *", daß er nach dem ersten Programm auf der Diskette sucht. Die rote Leuchtdiode signalisiert, daß das Laufwerk nun in Betrieb ist. Hat er dieses gefunden, so bestätigt er es mit "LOADING". Nun geschieht eine Weile nichts, bis das Programm vollständig von der Diskette in den Speicher des Rechners geladen wird. Nach erfolgtem Ladevorgang meldet der Rechner "READY" und die rote Leuchtdiode erlischt. Das Programm wird dann automatisch gestartet.

Laden von Kassette

Programme von der Datassette zu laden ist erheblich einfacher. Nachdem Sie die Kassette zurückgespult haben, geben Sie nur den Befehl

LOAD

ein, ohne weitere Angaben zu machen. Nun fordert der Rechner Sie auf, die 'PLAY'-Taste zu drücken (PRESS PLAY ON TAPE). Nachdem Sie dies gemacht haben, verschwindet der Bildschirm und der Ladevorgang beginnt. Nach einiger Zeit erscheint wieder der Bildschirm und der Rechner meldet 'READY'. Nun können Sie das Programm mit dem Befehl 'RUN' starten.

Ladeprobleme von Diskette

Welche Fehler können beim Laden von der Diskette auftreten und wie werden sie behoben? Grundsätzlich werden Fehler durch eine Meldung und/oder durch blinkende Leuchtdiode am Laufwerk signalisiert. Der erste Fehler kann nach dem Befehl Ladeversuch mit 'SHIFT' - 'RUN/STOP' auftreten. Die rote Leuchtdiode blinkt und der Rechner meldet "?FILE NOT FOUND ERROR". Tritt dieser Fehler auf, so ist entweder keine Diskette eingelegt - kann jedem 'mal passieren - oder die Klappe ist nicht geschlossen. Dies läßt sich aber noch beheben. Schlimmer ist, wenn die Diskette eingelegt ist und trotzdem dieser Fehler auftritt. Dann gibt es meist nur noch eine Möglichkeit: Die Diskette ist zerstört.
Dies kann z.B. auf dem Postwege geschehen, da die Post oft von magnetischen Feldern umgebene Transportbänder einsetzt. Hier bleibt Ihnen nichts anderes über, als die Diskette beim Händler umzutauschen. Ob dies nun kostenlos oder mit einem kleinen Unkostenbeitrag erfolgt, ist von Händler zu Händler verschieden.

Weiterhin kann ein Fehler nach der Meldung "LOADING" auftreten. Hier wird keine Meldung ausgegeben, sondern es blinkt die rote Leuchtdiode am Laufwerk. Diese Situation ist etwas verzwickter. Die Diskette ist zwar nicht zerstört, hat aber eine kleine "Macke". Welcher Fehler genau aufgetreten ist, können Sie mit einer Befehlsfolge selbst ermitteln. Für den Händler ist es sehr aufschlußreich, wenn Sie ihm gleich mit der Reklamation die Fehlermeldung mitteilen können. Ist der Cursor nicht auf dem Bildschirm, so "hängt" der Rechner. Sie können dann meist mit gelichzeitigem Druck der Tasten "RUN/STOP" und "RESTORE" den Cursor zurückholen.
Geben Sie nun den folgenden Befehl ein:

PRINT DS$

Auch hier schicken Sie den Befehl wie immer mit 'RETURN' zum Rechner.
Nun wird die genaue Beschreibung des Fehlers ausgegeben. Was der Fehler nun besagt, überlassen Sie Ihrem Händler,

der sich ein besseres Bild von der fehlerhaften Diskette machen kann.

Ladeprobleme von Kassette

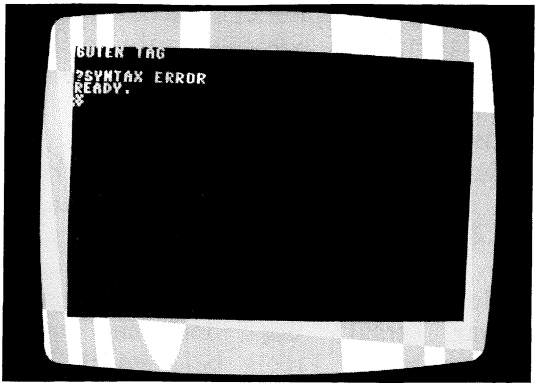
Auch eine Kassette besteht aus magnetischem Aufzeichnungsmaterial, das leicht beschädigt werden kann. Wenn Sie bei einer Musikkassette nur ein leichtes, kaum störendes Kratzen hören, so kann dies bereits das Laden des Programms unmöglich machen. Die einzige Fehlermeldung, die auftreten kann, ist "LOAD ERROR". Viele, die mit der Datassette arbeiten,. kennen diesen Fehler sicherlich sehr gut. Oft ist eine falsche Spurlage des Tonkopfes die Ursache dafür. Ein Programm, das mit falscher Spurlage aufgenommen wird, läßt sich nur von diesem Rekorder laden. Ein anderer Rekorder, desses Spurlage stimmt, kann dieses Programm jedoch nicht laden. So kann es vorkommen, daß Sie Ihre eigenen Programme laden und speichern können, gekaufte Programme aber einen "LOAD ERROR" verursachen. Dann ist meist der falsch eingestellte Tonkopf Ihres Rekorders der Übeltäter. Sollten Sie oft derartige Schwierigkeiten mit Ihrer Datassette haben, so lassen Sie diese von Ihrem Händler justieren.

DER ERSTE BEFEHL

Zur weiteren Erklärung der Tastatur reicht es nun nicht mehr aus, irgendwelche Zeichen auf den Bildschirm zu schreiben. Die wenigsten unserer Leser erwarben ihren COMMODORE 128 für diesen Zweck. Für alle diejenigen, die sich im letzten Kapitel etwas langweilten, wird es in diesem Kapitel interessanter. Wir wollen nun den Computer dafür nutzen, wofür er ja einzig und allein geeignet ist, nämlich zum Ausführen von Befehlen.

Die RETURN-Taste

RETURN ("ritörn" gesprochen) ist die wichtigste Taste an Ihrem Rechner. Wenn Sie diese Taste drücken, wird der Inhalt der aktuellen Zeile an den Rechner übergeben. Der Rechner interpretiert diese Zeile als Befehl und leitet die notwendigen Schritte ein. Nach Abschluß des Befehls meldet sich der Rechner wie nach dem Einschalten mit 'READY' wieder. Er ist also wieder bereit zur Aufnahme weiterer Instruktionen. Versuchen Sie nun einmal, den Rechner zu begrüßen. Geben Sie den Text "GUTEN TAG" ein und drücken anschließend die Taste RETURN. Was meinen Sie, wird Ihr Rechner antworten? Doch sehen Sie selbst.



Hier erscheint die erste Fehlermeldung Ihres COMMODORE 128, den SYNTAX ERROR (Syntax-Fehler). Die Syntax, also die Zusammensetzung der Zeichen, wird vom Computer nicht verstanden. Zwar gibt es heutzutage Großrechner, die sich weitgehend mit dem Menschen über eingegrenzte Themen "unterhalten" können, jedoch erfordert dies eine Anlage mit huntertausendfachem Preis und Speicherkapazität. Das, was Sie mit der Tastatur eingeben und mit RETURN "abschicken", wird vom BASIC-Interpreter analysiert und bei fehlerfreier Eingabe ausgeführt.

Wenn Sie die Taste RETURN zusammen mit der Taste SHIFT betätigen, wird die Zeile nicht an den Rechner weitergegeben, sondern nur der Cursor zum Anfang der nächsten Zeile befördert. Sie schließen somit die Zeile ab und schreiben in der nächsten Zeile weiter.

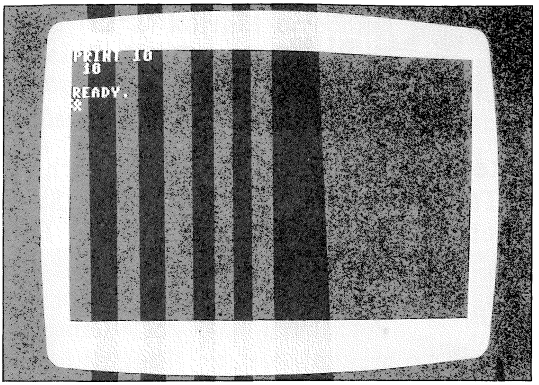
Ihr COMMODORE 128 ist mit der Programmiersprache BASIC ausgerüstet und versteht nur Befehle dieser Sprache. BASIC ist die am meisten verbreitetste Programmiersprache, die gerade dem Anfänger optimale Voraussetzungen zum Programmieren bietet. Die Befehle sind weitgehend der englischen Sprache angelehnt. Wie schnell kann ich diese Sprache erlernen? Dies ist eine beliebte Frage, die jedoch nicht hundertprozentig beantwortet werden kann. Viele Kriterien spielen hier eine Rolle. Einmal ist die Lernfähigkeit und Begeisterung des Anfängers ausschlaggebend, zum anderen ist die Zeit von Bedeutung, die man am Rechner investiert. Da gibt es

Leute, die jede freie Minute dem hochverehrten Computer widmen. Zeitungsanzeigen wie "Verkaufe meinen CBM 128, Scheidung droht" sind keine Seltenheit mehr. Grundsätzlich gilt es, ein gesundes Maß an Zeit zu opfern, um das Hobby der Programmierung in BASIC zur hellen Freude werden zu lassen. Mit einem gewissen Grundwissen in Mathematik, das vielleicht bis zu den Grundlagen der Algebra reicht, können bei einem wöchentlichen Zeitaufwand von ca. 10 Stunden schon nach etwa drei Monaten die ersten Erfolgserlebnisse verbucht werden. Zwar trägt das logische und abstrakte Denkvermögen mit zum Erfolg bei, jedoch wird sich dies auch von Programm zu Programm ständig weiterentwickeln. Zum Erstellen von anspruchsvollen Programmen wie z.B. einer kleinen Textverarbeitung oder einer Datenverwaltung ist neben der Programmierkenntnis die Programmiererfahrung unerläßlich. Von Programm zu Programm steigert sich erfahrungsgemäß die Qualität des Programmierstils. Ein kleiner Anhaltspunkt für Sie: Viele haben schon drei Monate nach Erwerb ihres ersten Computers viel Freude am Entwickeln von kleinen Spielen gehabt. Solange die Begeisterung da ist, werden auch Sie bald einer der vielen Hobby-Programmierer sein.

Der PRINT-Befehl

Ohne diesen Befehl ist kaum ein Programm denkbar. Er übernimmt die gesamte Ausgabe in einem Programm. Ob Sie nun ein Rechenergebnis auf dem Bildschirm oder eine Adresse auf das Kassettenlaufwerk ausgeben wollen, ohne diesen Befehl läuft nichts. Auch die gesamte Druckausgabe wird vom Befehl 'PRINT' übernommen.

Zur Zeit wenden wir uns nur dem Direkt-Modus zu, wir schreiben also noch keine Programme. Direkt-Modus bedeutet, Sie geben Befehle ein, schließen sie mit RETURN ab und erhalten ein sofortiges Ergebnis. Dies wollen wir nun erstmalig ohne Fehlermeldung durchführen. Geben Sie nun den Befehl 'PRINT 10' gefolgt von einem RETURN ein.



PRINT leitet also alles, was dem Befehl als Parameter angefügt wird, zum Bildschirm (die Ausgabe auf externe Geräte erfolgt mit einer anderen Form des PRINT). Parameter sind Bestandteile eines Befehls, die genauestens beschreiben, welche Auswirkung der Befehl haben soll.

Hier war die Zahl '10' der Ausgabeparameter. 'PRINT 10' bedeutet also 'zeige die Zahl 10 auf dem Bildschirm'. Natürlich können Sie nicht nur Zahlen, sondern auch andere beliebige Zeichen und sogar Steuerzeichen (CRSR LINKS, CLR usw) ausgeben. Wie vielseitig dieser Befehl ist, stellen Sie spätestens im weiteren Verlauf dieses Kapitels fest.

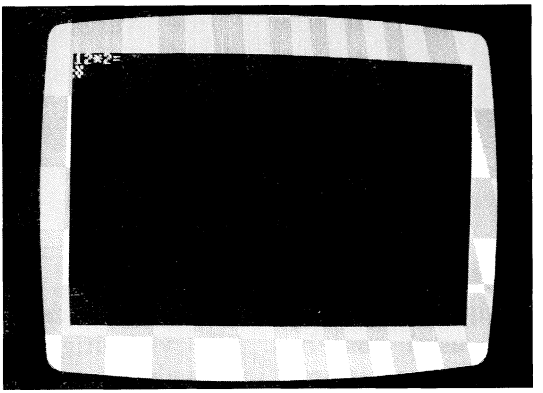
Rechnen mit PRINT

Wenn Sie einmal ausrechnen möchten, wieviel Lohnsteuer Sie wohl zurückerstattet bekommen und gerade keinen Taschenrechner zur Hand haben, schalten Sie den COMMODORE 128 ein! Ich selbst schalte auch oft den Rechner nur für kleine Berechnungen ein. Viele meiner Bekannten haben dann oft ironisch gefragt "was, rechnen kann man damit auch?". Das wäre ja auch ein Unding, wenn man mit einem Computer, der immerhin das zwanzigfache eines Taschenrechners kostet, nicht rechnen könnte.

Doch nun zur Sache. Wenden wir uns zunächst den vier Grundrechenarten zu. Hierfür gibt es auf der Tastatur vier Symbole:

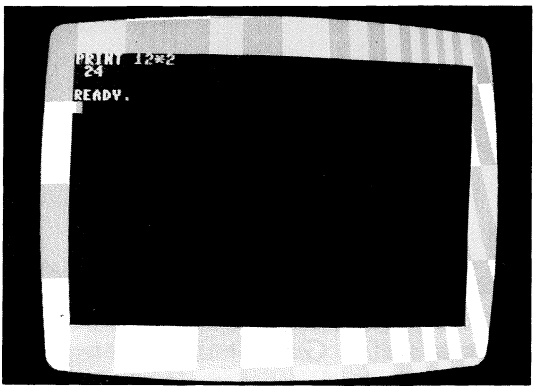
```
+  für Addition
-  für Subtraktion
*  für Multiplikation
/  für Division
```

Nur ist das Rechnen mit Ihrem Computer nicht vergleichbar mit einem Taschenrechner. Hier können Sie z.B. nicht '12*2=' eingeben, da bekanntlich ohne Befehl nichts läuft. Es gibt immer experimentierfreudige Leser, die dies trotzdem versuchen werden. Was dann geschieht, ist etwas verwirrend. Versuchen Sie es selbst. Geben Sie '2*12=' ein und schicken diesen "Befehl" mit der Taste 'RETURN' ab.



Was dann geschieht, ist ohne vorzugreifen schwer zu erklären. Der Rechner zeigt nicht das Ergebnis, sondern meldet sich mit READY wieder. Der Cursor springt zum Anfang der nächsten Zeile. Aber was geschieht dazwischen? Der Rechner muß etwas ausgeführt haben, da er sonst eine Fehlermeldung ausgegeben hätte. Ich möchte hier nicht zu weit ausholen. Doch ich muß etwas vorgreifen. Ein Programm besteht aus mehreren Zeilen, die nach Programmstart hintereinander abgearbeitet werden. Jede dieser Zeilen ist am Anfang mit einer Zahl gekennzeichnet, die die Reihenfolge beim Ablauf bestimmt. In diesem Fall wurde die Programmzeile 12 mit dem Inhalt '*2=' eingegeben. Dies ist zwar kein korrekter Befehl, jedoch wird das nicht bei der Eingabe der Zeile erkannt, sondern erst beim Ablauf des Programms. Doch lassen Sie uns dies zunächst einmal ignorieren.

Wie wird nun '12*2' korrekt errechnet und ausgegeben? Die Lösung ist einfach: Geben Sie dazu den Befehl 'PRINT 12*2' ein. Beachten Sie: Immer wenn Sie einen Befehl eingeben, müssen Sie diesen mit RETURN abschließen. Was Sie auf dem Bildschirm schreiben, ist dem Rechner egal. Ihn interessiert nur das, was Sie ihm mit RETURN zur Ausführung überlassen.

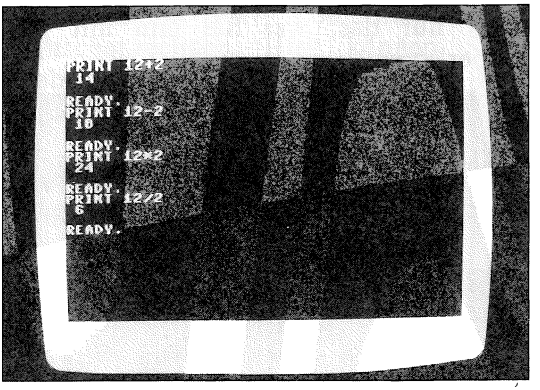


Das Ergebnis entlockt Ihnen vielleicht das erste "AHA". Ihr Rechner hat zum ersten Mal für Sie gearbeitet. Er folgte treu Ihrer Anweisung, 12 mit 2 zu multiplizieren und das Ergebnis auf dem Bildschirm anzuzeigen.

Nebenbei möchte ich bemerken, daß Programmiersprachen eigentlich nur kompliziert erscheinen, weil Sie die Anweisungen an den Rechner wesentlich verkürzen müssen. Jede Programmiersprache muß bis ins letzte Detail klar organisiert sein. Eine Programmiersprache, die Anweisungen wie 'RECHNE 2*12 UND ZEIGE ERGEBNIS' akzeptiert, gibt es nicht und wird es auch nie geben. Verdrängen Sie bitte derartige Erwartungen. Ganz so einfach ist das Programmieren auch nicht. Wenn Sie es beherrschen, dürfen Sie sich nicht umsonst selbst auf die Schulter klopfen.

Hinter dem PRINT-Befehl ist ein Leerzeichen eingegeben worden. Dies ist nicht unbedingt erforderlich, trägt jedoch zur Übersichtlichkeit des Programms bei. Sie können das Leerzeichen auch ignorieren und einfach 'PRINT2*12' eingeben. Leerzeichen dienen demnach nur der Übersichtlichkeit des Programms.

Doch nun weiter mit den Grundrechenarten. Berechnen Sie nun hintereinander '12+2', '12-2', '12*2' und '12/2'. Ihr Bildschirm sollte nachher dem folgenden Bildschirm gleichen:



Haben Sie es geschafft? Gut, dann dürfen Sie wieder eine Sprosse höher auf der Erfolgsleiter zum Hobby-Programmierer steigen. Nein, dann lesen Sie bitte den letzten Abschnitt nochmals aufmerksam durch.

Vielleicht haben Sie es schon probiert: Die Zifferntasten des kleinen Tastenfeldes rechts auf der 128er Tastatur entsprechen in ihrer Wirkung genau den Zifferntasten der oberen Tastaturreihe. Das gilt auch für die Tasten '+ - *'. Die Taste mit der Aufschrift 'ENTER' entspricht der 'RETURN' Taste. Wenn Sie viele Zahlen einzugeben haben oder eben mit Ihrem 128er wie mit einem Taschenrechner rechnen wollen, so werden Sie dieses zusätzliche Tastenfeld noch sehr begrüßen. Probieren Sie die obigen Beispielrechnungen doch einmal mit dem Ziffernfeld aus.
Neben solch simplen Berechnungen können Sie auch lange Berechnungen bis zu zwei Zeilen ausrechnen! Beachten Sie jedoch die Hierarchie bei solchen Berechnungen: Punktrechnung geht stets vor Strichrechnug. Geben Sie nun dem Rechner die Aufgabe, das Ergebnis von '1+2+3+4+5+6+7+8+9' zu ermitteln.

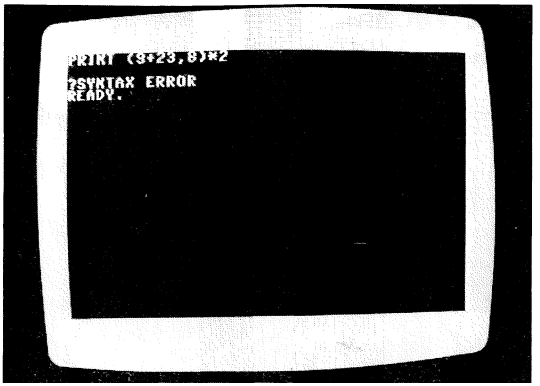
Die Klammerrechnung

Es besteht also die Möglichkeit, beliebige Kettenrechnungen durchzuführen. Ein praktisches Beispiel: Es soll der Gesamtpreis von drei Teppichstücken ermittelt werden. Der Quadratmeterpreis beträgt 23.80 DM. Das erste Stück umfaßt 2.45 m * 2.80 m, das zweite Stück 4.50 m * 3.85 m und das dritte Stück 2.75 m * 4.80 m. Zusätzlich soll 14% Mehrwertsteuer addiert werden. Wie hoch ist nun der Gesamtpreis für diese drei Teppichzuschnitte? Wenn Sie die Klammern richtig setzen, genügt ein PRINT-Befehl:

PRINT (2.45*2.8+4.5*3.85+2.75*4.8)*23.8*1.14

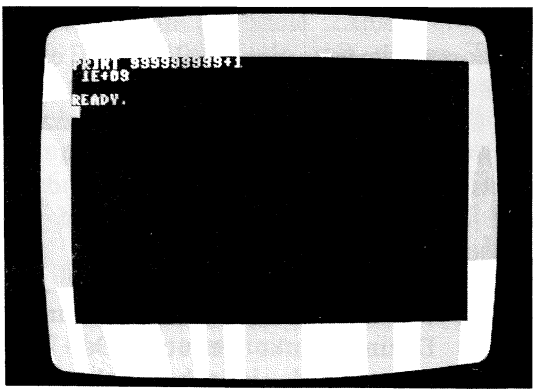
Das sieht komplizierter aus, als es eigentlich ist. Innerhalb der Klammer werden die Quadratmeter ausgerechnet. Dann wird die Quadratmeteranzahl mit dem Quadratmeterpreis multipliziert. Schließlich wird noch die Mehrwertsteuer hinzugerechnet. Und das alles mit einem Befehl! Bei korrekter Eingabe erhalten Sie den Wert 1014.32982, also den Preis 1014.33 DM.

Wie Sie bereits erkannt haben, wird der Punkt und nicht das Komma als Dezimalpunkt benutzt. Das ist zwar eine amerikanische Norm, wird jedoch auch bei deutschen Rechnern praktiziert. Wenn Sie das Komma in Verbindung mit der Klammerrechnung eingeben, z.B. PRINT (9+23,8)*2, so erscheint die Fehlermeldung 'SYNTAX ERROR'.



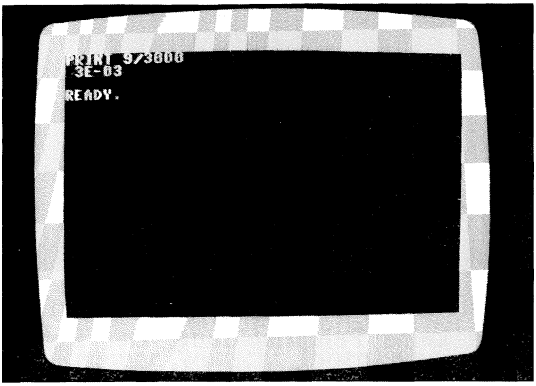
Exponentialschreibweise

Der Rechner stellte das Ergebnis mit 5 Nachkommastellen dar. Es ist eine sogenannte Fließ- oder Gleitkommazahl. Grundsätzlich wird auf neun Stellen genau gerechnet. Ergibt sich eine Zahl mit drei Stellen vor dem Komma, so wird automatisch auf 6 Stellen nach dem Komma gerechnet. Sollte die Zahl 999999999 überschreiten, so wird wieder auf neun Stellen genau gerechnet. Eine größere Zahl erhält dann einen Zusatz, der die Größe des echten Ergebnisses kennzeichnet. Geben Sie z.B. einmal die Rechnung 'PRINT 999999999+1' ein. Das Ergebnis ist eine zehnstellige Zahl, die nicht mehr normal dargestellt werden kann.



Verzweifeln Sie nicht bei diesem Ergebnis. Es bedeutet einfach, daß das Ergebnis'1 * 10 hoch 9' ergibt, also

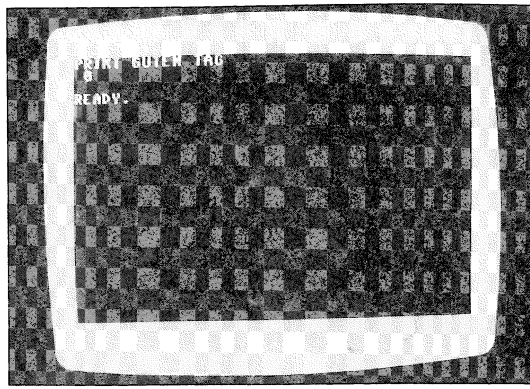
eine 1 mit 9 Nullen. 'E+09' bedeutet Basis 10, Exponent +9. Der Exponent kann aber auch negativ sein. Die Rechnung 'PRINT 9/3000'bestätigt dies.



Obwohl das Ergebnis eigentlich 0.003 lauten sollte, was nicht mehr als 9 Stellen umfaßt, benutzt der Rechner die Exponentenschreibweise. '3E-03' ist gleichbedeutend mit '3 * 10 hoch -3'. Die Zahl '3' befindet sich also an dritter Stelle nach dem Komma. Das ist vorerst alles, was Sie über das Rechnen mit den vier Grundrechenarten im Direkt-Modus wissen sollten. Weitere mathematische Funktionen finden Sie in Ihrem Handbuch zum COMMODORE 128.

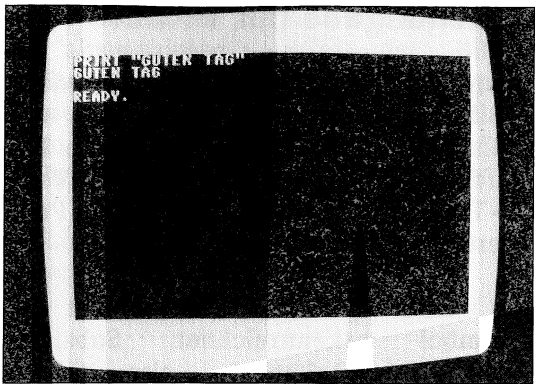
Textausgabe mit PRINT

Neben Zahlen können auch Texte, sogenannte "Strings" (Zeichenketten), mit PRINT ausgegeben werden. Voreilige Versuche wie 'PRINT GUTEN TAG' werden kläglich scheitern.



Obwohl der Befehl nicht den gewünschten Erfolg verbuchen kann, erscheint keine Fehlermeldung. Der Rechner hat den Befehl also ausgeführt. Nur was hat er ausgeführt? Woher die Null? Dies sind Probleme, die bei den ersten Gehversuchen mit BASIC immer wieder auftreten. Die Fragen sind ohne vorzugreifen schwer zu beantworten. Kurz erklärt, wird der Inhalt einer Variablen (rechnerinterner Zahlenspeicher) angezeigt. Die Bezeichnung dieses Speichers besteht aus maximal 2 Zeichen. Werden mehr als zwei Zeichen angegeben, so werden nur die ersten (in diesem Fall GU) akzeptiert. Da wir in der Variablen GU nichts abgelegt haben, wird der Wert 0 ausgegeben. Sollten Sie dies nicht ganz verstanden haben, so ist das kein Grund zur Beunruhigung. Die Variablen werden später noch ausführlich behandelt.

Doch wie kann man nun Zeichenketten (bleiben wir im weiteren Verlauf beim Fachwort "Strings"), wie kann man Strings nun mit PRINT ausgeben? Die Lösung ist einfach: Strings werden in Gänsefüßchen (Anführungszeichen) eingeschlossen. Setzen Sie dies nun in die Tat um und ändern den letzen Befehl. Geben Sie den String 'GUTEN TAG' nun in Anführungszeichen ein.



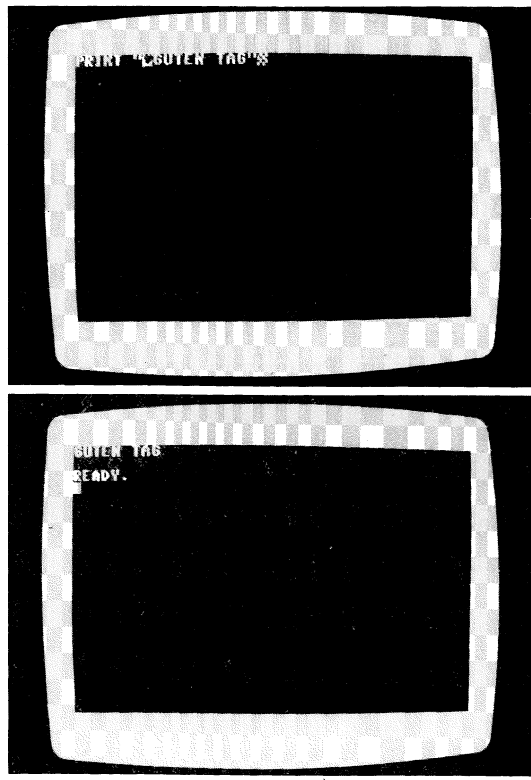
Na also, nach dem ersten Fehlversuch hat es nun doch geklappt. Der Rechner wünscht Ihnen einen guten Tag (er macht bekanntlich all das, was Sie von ihm verlangen).

Übrigens: Wenn Sie bisher allzu zaghaft mit Ihrem Rechner umgegangen sind und jeden Befehl vorher dreimal überlegten, um ja nichts kaputt zu machen, so kann ich Sie beruhigen. Kein noch so fehlerhafter Befehl kann Ihrem Rechner Qualmwolken entlocken. In kaum einem anderem Hobby ist die Weisheit "aus Fehlern lernt man" so angebracht wie bei Ihrem. Ein Elektronik-Bastler, der die 45,- DM Endstufe seines selbstgebauten Verstärkers falsch einlötet, beißt sicher in den Teppich. Wenn Sie jedoch einen Fehler in Ihrem 23 KByte-Programm verursachen, so kann er in fünf Minuten wieder korrigiert sein. Der einzige und gleichzeitig schlimmste Fehler, den Sie verursachen können, ist, wenn Sie z.B. acht Stunden hintereinander an einem Programm gearbeitet haben und die Mutter den defekten Staubsauger einschaltet. Ein Kurzschluß hat hier große Auswirkung, nämlich acht Stunden verschenkte Arbeit. Auch können Sie z.B. eine wichtige und wertvolle Diskette durch einen falschen Befehl zerstören. Doch wenn Sie die wichtigsten Richtlinien zur Datensicherung beachten, kann nichts schiefgehen.

Steuerzeichenaugabe mit PRINT

Ein Programm muß selbstverständlich in der Lage sein, z.B. den Bildschirm zu löschen, um weitere Daten auszugeben. Es wäre ja undenkbar, wenn ein Programm den Benutzer auffordern würde, die SHIFT-Taste mit der CLR/HOME-Taste zu drücken, damit der Programmablauf fortgesetzt werden kann.

Steuerzeichen werden in Strings übernommen und mit einem reversen Symbol gekennzeichnet. Sobald Sie den String-Modus einschalten, also das erste Anführungszeichen setzen, werden die Steuertasten nicht sofort ausgeführt, sondern im String abgelegt. Wollen Sie z.B. vor der Ausgabe von "GUTEN TAG" den Bildschirm löschen, so geben Sie nach dem ersten Anführungszeichen, also vor dem "GUTEN TAG" das Steuerzeichen für 'Bildschirm löschen' ein. Dazu drücken Sie wie üblich die Taste SHIFT und die Taste CLR/HOME gleichzeitig.

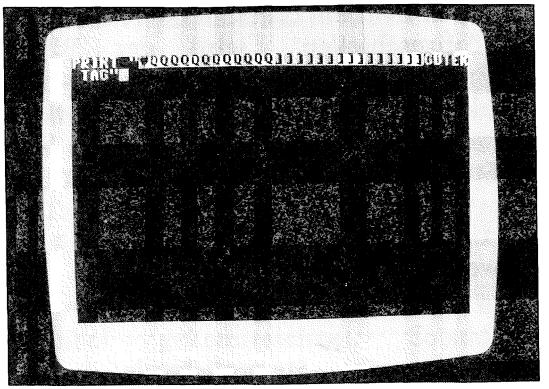


In dem Moment, in dem Sie die RETURN-Taste drücken, um den Befehl "abzuschicken", wird der Bildschirm vor Ausgabe des Strings gelöscht. Was passiert wohl, wenn Sie dieses Steuerzeichen hinter "GUTEN TAG" setzen? Richtig, der Bildschirm wird nach Ausgabe des Strings sofort gelöscht. Der String ist somit kaum zu erkennen.

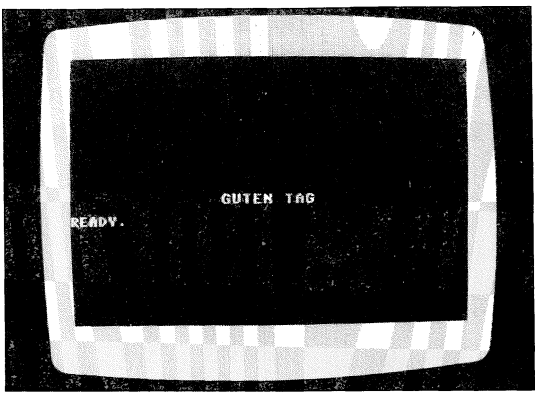
Hier eine Liste der Steuerzeichen, die in Strings übernommen werden können:

Steuerzeichen	Steuerzeichen
CRSR RECHTS	GRÜN
CRSR LINKS	BLAU
CRSR UNTEN	GELB
CRSR OBEN	ORANGE
HOME	BRAUN
CLR	HELLROT
SCHWARZ	GRAU 1
WEISS	GRAU 2
ROT	HELLGRÜN
TÜRKIS	HELLBLAU
VIOLETT	GRAU 3
RVS ON	RVS OFF

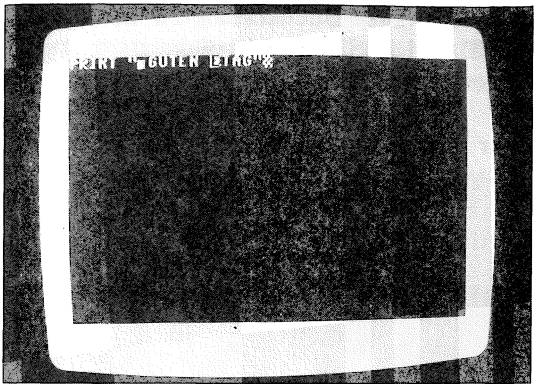
Sämtliche Cursor-Steuerungen können demnach auch in einem String untergebracht werden. Damit der Text "GUTEN TAG" nach Löschen des Bildschirms in der Mitte des Bildschirms erscheint, wollen wir nun die entsprechenden Cursor-Steuerzeichen im String unterbringen. Die Reihenfolge soll sein: Bildschirm löschen, Cursor 12 mal nach unten, Cursor 15 mal nach rechts und schließlich "GUTEN TAG". Der Befehl müßte dann so aussehen:



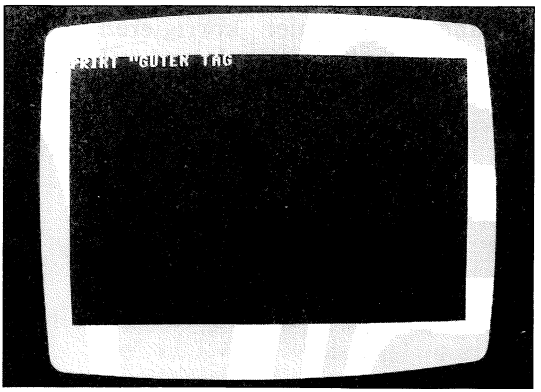
Wenn Sie diesen Befehl nun mit RETURN dem Rechner übergeben, sehen Sie den Erfolg:



Doch nicht nur Cursorsteuerungszeichen, sondern auch Farben und reverse Zeichendarstellung können mit einem PRINT-Befehl eingesetzt werden. Auch hier geben Sie an der gewünschten Stelle im String die Farbe entweder mit der CTRL- oder mit der C= - Taste ein. Die Tasten werden genauso bedient, als ob Sie die Farbe direkt umschalten möchten. Der Unterschied ist wieder, daß, solange der String noch nicht abgeschlossen ist, die Farbe nicht unmittelbar geändert wird. Es wird im String ein Steuerzeichen festgehalten, das erst bei Ausgabe mit PRINT zur Wirkung kommt. Versuchen Sie nun einmal, den String "GUTEN TAG" zweifarbig auf dem Bildschirm darzustellen. Das erste Wort in schwarz und das zweite Wort in weiß.



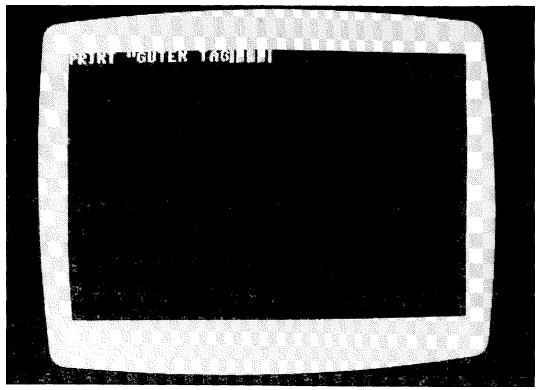
Haben Sie es geschafft? Es ist eigentlich recht einfach, Steuerzeichen in Strings einzubinden. Sie müssen nur beachten, daß sobald das erste Anführungszeichen gesetzt wurde, sämtliche Steuertasten bis auf INST/DEL nicht mehr zu verwenden sind. Der Nachteil ist ersichtlich: Wenn Sie ein String mit einem Anführungszeichen begonnen haben und feststellen, daß Sie sich ein paar Zeichen zuvor vertippt haben, so ist es nicht mehr möglich, dies ohne weiteres zu korrigieren. Ein Beispiel sagt mehr als viele Worte: Geben Sie einmal folgenden noch nicht abgeschlossenen Befehl ein:



Sie stellen nun fest, daß das Leerzeichen zwischen den beiden Worten vergessen wurde. Würden wir uns nicht in einem noch nicht abgeschlossenen String befinden, so wäre der Ablauf zur Korrektur folgender:

1. Cursor nach links zum Buchstaben 'N' bewegen.
2. Tasten SHIFT und INST/DEL drücken.
3. Cursor wieder rechts hinter dem 'G' positionieren.

Da hier aber noch ein String erstellt wird, scheitert schon der erste Punkt. Versuchen Sie es selbst. Halten Sie die Taste SHIFT gedrückt und tippen viermal auf die Taste CRSR LINKS/RECHTS. Das Ergebnis war abzusehen:

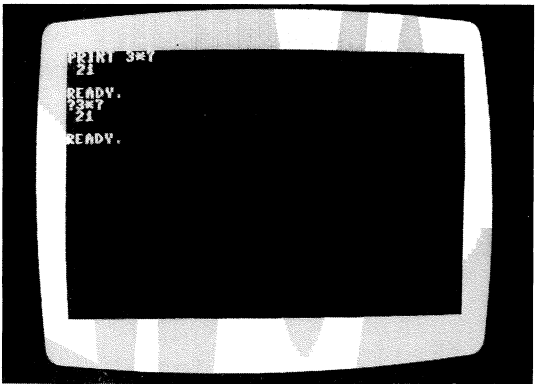


Es werden vier Steuerzeichen für Cursor nach links abgebildet. Der beste Ausweg aus dieser Situation ist, den String zunächst mit dem zweiten Anführungszeichen abschließen, dann den Fehler korrigieren und schließlich an der gewünschten Stelle weiterschreiben. Sie können also daß zweite Anführungszeichen wieder überschreiben und es an der endgültigen Stelle wieder setzen.

Vereinfachte PRINT-Eingabe

Anstelle des Befehls PRINT kann man auch ein Fragezeichen eingeben. "Warum lese ich das jetzt erst?" werden viele hier fragen. Alle Befehle können in verkürzter Form eingegeben werden. Für den Anfänger ist das jedoch nicht empfehlenswert, da man von der eigentlichen Bedeutung der Befehle zu sehr abgelenkt wird. Der Anfänger sollte die Befehle grundsätzlich ausschreiben, um Verwirrung und zusätzliche Fehlerquellen zu vermeiden.

Geben Sie anstelle des Befehls PRINT das Fragezeichen ein, so erspart man sich damit etwas Arbeit, zumal der Befehl PRINT der am häufigsten verwendete Befehl ist. Es ist also egal, ob Sie nun 'PRINT 3*7' oder '? 3*7' eingeben. Probieren Sie es selbst einmal aus.

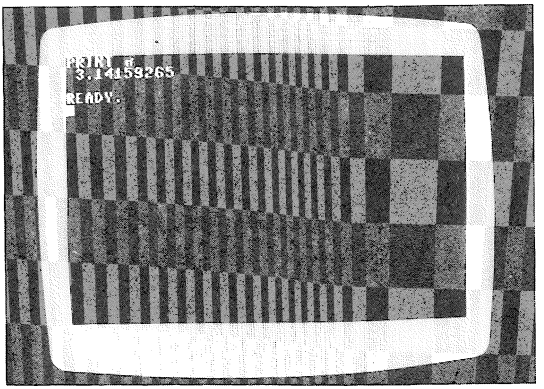


Es liegt nun an Ihnen, ob Sie den PRINT-Befehl ausschreiben oder das Fragezeichen verwenden. Bei Berechnungen, die "mal eben" eingetippt werden, ist das Fragezeichen recht praktisch. Man kann damit ebenso schnell rechnen wie mit einem Taschenrechner.

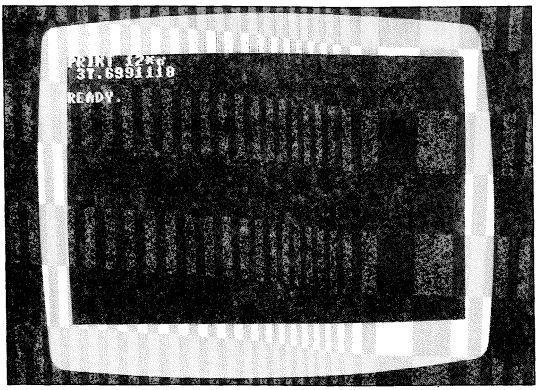
PI und Potenzierung

Wenden wir uns nun wieder der Mathematik zu. Die Konstante PI dürfte jedem bekannt sein. Bei der Kreis- und Kugelberechnung wird sie meistens benötigt. Welchen Wert hat denn nun PI? War es 3.1412 oder 3.1214 oder ... Zerbrechen Sie sich nicht den Kopf. Meist wird mit 3.14

gerechnet, also auf zwei Stellen nach dem Komma. Der COMMODORE 128 hält diese Konstante für Berechnungen bereit. Rufen Sie das PI einmal auf den Bildschirm. Geben Sie dazu PRINT ein. Das PI erhalten Sie mit SHIFT und der Taste PFEIL HOCH/PI.



PI auf acht Stellen genau! Das ist vollkommen ausreichend. Berechnen Sie nun einmal den Umfang eines Kreises mit dem Durchmesser von 12 cm. Die Formel lautet: Umfang = Durchmesser * PI.

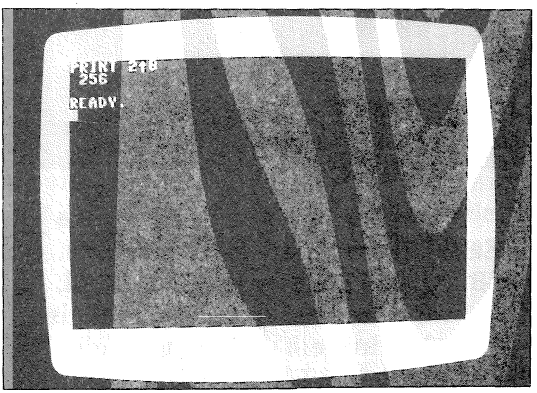


Der Kreisumfang beträgt also rund 37.7 cm.

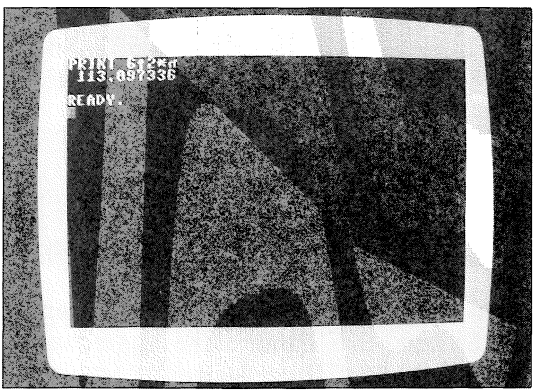
Nun zur Potenzierung. Potenzierung bedeutet, eine Zahl mit sich selbst zu multiplizieren. Der Exponent (die Hochzahl) bestimmt, wie oft die Basis (Grundzahl) mit sich selbst multipliziert werden soll.

2 hoch 3 bedeutet 2*2*2, also 8
10 hoch 4 bedeutet 10*10*10*10, also 10000

Im ersten Beispiel ist '2' die Basis und '3' der Exponent. Doch nun Schluß mit dem Mathematik-Nachhilfeunterricht. Beim COMMODORE 128 wird zur Potenzierung zwischen der Basis und dem Exponent ein 'Pfeil nach oben' gesetzt. Rechnen Sie nun einmal 2 hoch 8 aus.



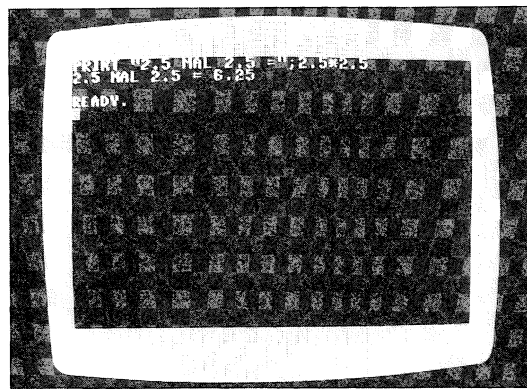
Das war sicher nicht schwer. Berechnen Sie nun die Kreisfläche des Kreises mit dem Durchmesser von 12 cm. Die Formel lautet: Kreisfläche = Radius hoch 2 mal PI (Radius ist der halbe Durchmesser).



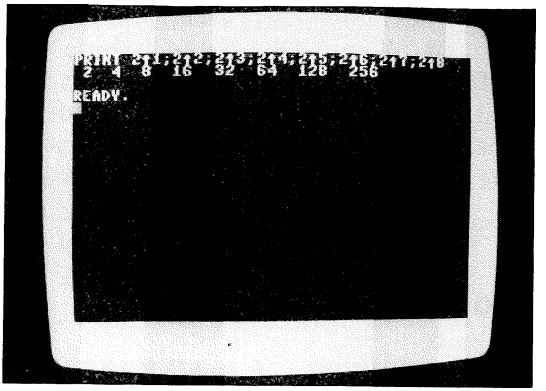
Die Kreisfläche beträgt also ca. 113 qcm. Doch nun genug von der trockenen Mathematik. Wenden wir uns wieder interessanteren Sachen zu.

Kombinieren von Strings mit Zahlen

Die Tasten ';' und ',' spielen eine große Rolle, wenn Strings und Zahlen kombiniert werden sollen. Aber welche? Wenn Sie z.B. eine Zeile wie "2.5 MAL 2.5 = 6.25" ausgeben wollen, indem das Ergebnis gleichzeitig errechnet wird, so wird ein String und eine anschließende Berechnung benötigt. Die Lösung sieht dann wie folgt aus:

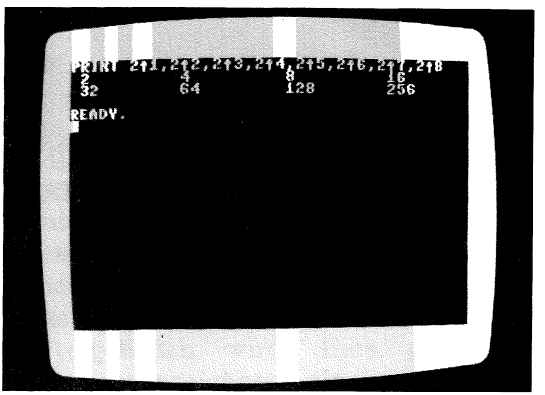


Das Semikolon trennt also Strings von Zahlen. Doch nicht nur das, auch Strings und Zahlen bzw. Berechnungen können untereinander durch Semikolon getrennt werden. So können Sie z.B. mehrere Berechnungen mit einem PRINT-Befehl ausführen und die Ergebnisse nebeneinander anzeigen. Versuchen Sie nun, die Potenzen der Zahl '2' vom Exponent '1' bis zum Exponent '8' zu berechnen und nebeneinander in einer Zeile anzuzeigen.



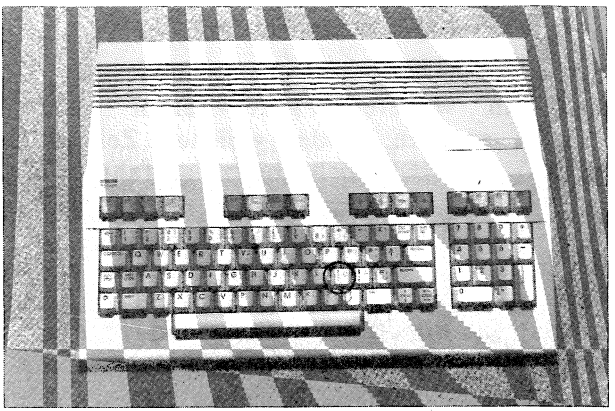
Ein durch SEMIKOLON abgeschlossener PRINT-Befehl hat zur Folge, daß ein weiterer PRINT-Befehl nicht in der nächsten Zeile, sondern in der gleichen Zeile hinter dem ersten PRINT erfolgt. Doch hierzu erfahren Sie später mehr.

Geben Sie nochmals den letzten Befehl ein, ersetzen aber alle Semikolons durch ein Komma. Ein Tip: Benutzen Sie den komfortablen Bildschirm-Editor und "laufen" Sie mit dem Cursor über die schon eingegebene Zeile. Wenn Sie nun anstelle der Semikolons ein Komma setzen, drücken Sie RETURN und der Befehl wird ausgefürt. Grundsätzlich können Sie jede noch auf dem Bildschirm befindliche Anweisung nochmals abschicken, indem Sie den Cursor auf die entsprechende Zeile positionieren und RETURN drücken. Auch können Sie diese Zeile noch abändern.

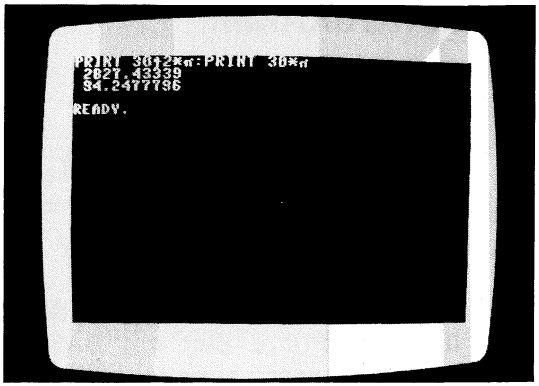


Das Komma trennt auch die einzelnen Werte, aber mit größerem Abstand. Wenn Sie die Abstände auszählen, so werden Sie feststellen, daß der Abstand zwischen den Ziffern jeweils 10 Zeichen beträgt. Dadurch wird eine leicht formatierte Ausgabe von Zahlenkolonnen ermöglicht. Zur Trennung von Strings wird das Komma nur selten genutzt.

Trennen von Befehlen

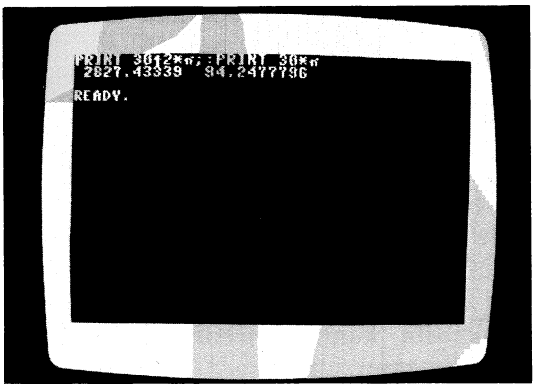


Eine Befehlszeile kann aus maximal 80 Zeichen, also 2 Zeilen bestehen. BASIC-Befehle erreichen diese Länge sehr selten. Durch den Doppelpunkt kann man nun Befehle voneinder trennen und somit mehrere Befehle in einer Programmzeile oder einer direkt eingegebenen Befehlszeile darstellen. Ein Beispiel soll hier Klarheit schaffen: Sie wollen zwei Berechnungen mit einem RETURN durchführen. Die Ergebnisse sollen untereinander erscheinen. In unserem Beispiel berechnen wir einmal 30 hoch 2*PI und einmal 30*PI. Versuchen Sie nun, zwei PRINT-Befehle in einer Zeile unterzubringen und durch den Doppelpunkt zu trennen.



Sie können natürlich auch drei oder vier Befehle eingeben und durch Doppelpunkte trennen. Es ist nur darauf zu achten, daß zwei Zeilen nicht überschritten werden.

Hier läßt sich auch sehr gut die Wirkung des Semikolons nach dem ersten PRINT-Befehl verdeutlichen. Löschen Sie den Bildschirm und geben Sie die beiden PRINT-Befehle nochmals ein. Setzen Sie hinter das erste PRINT ein Semikolon.



Das Semikolon hinter dem ersten PRINT bewirkt, daß diese Zeile noch nicht abgeschlossen wird. Das zweite PRINT erfolgt unmittelbar hinter dem ersten PRINT. Die leeren Stellen zwischen den Zahlen kommen dadurch zustande, daß vor der ersten Ziffer der Zahl ein Zeichen für das Vorzeichen freigehalten wird. Eine positive Zahl wird nicht mit '+' gekennzeichnet, eine negative Zahl jedoch erhält als erstes Zeichen ein Minus (-).

DAS ERSTE PROGRAMM

Mit dem ersten Programm ist nicht etwa die ARD gemeint, sondern Ihre ersten Gehversuche bei der Programmierung in BASIC. Verdrängen Sie Ihren eventuellen Unmut dieses "heiße Eisen" anzufassen. BASIC ist eine Programmiersprache für den Einsteiger, vor der niemand allzu großen Respekt zu haben braucht. Die Tatsache, daß in Kaufhäusern und Computerläden oft Kinder anzutreffen sind, die nach der Schule ihre Zeit mit BASIC vertreiben, sollte die letzten Hemmungen beiseite schieben. Seien Sie kein ewiger Anwender, der seine ganze Zeit den Fertigprogrammen widmet. Mal ehrlich, haben Sie nicht auch oft daran gedacht, eigene Programme zu entwickeln? Dieses und das folgende Kapitel soll für Sie ein kleiner Schritt in die große und interessante Welt der Programmierung sein.

Ein Programm, was ist das?

Auf Fachchinesisch wird ein Programm als eine "Folge von Befehlen zur Lösung einer bestimmten Aufgabe" definiert. Diese Folge von Befehlen ist es also, die in logisch richtiger Zusammensetzung ein Programm ergibt. Der Weg zu einem Programm führt von der Aufgabenstellung über die logische Befehlsfolge zum Ziel, nämlich dem Programm. Logisch richtig bedeutet, daß nicht nur sämtliche BASIC-Befehle gekannt werden müssen, sondern daß sie erst durch ausgetüftelte Zusammenstellung zur Problemlösung führen. So ist es z.B. sinnlos, wenn Sie eine Unmenge von englischen Vokabeln kennen, wenn Sie diese nicht durch sinnvolle Zusammensetzung zur Kommunikation nutzen können.

Die Zeilennumerierung

Ein Programm ist also eine Folge von Befehlen. Doch wodurch wird diese Reihenfolge bestimmt? Nun bei der Programmiersprache BASIC wird jeder Befehl (auch Statement genannt) mit einer Nummer versehen, deren Reihenfolge bestimmt, wie das Programm ablaufen wird. Stellen Sie sich z.B. Ihr im Gehirn gespeichertes Programm zur Lösung der Rechenaufgabe '48/12' mit dem Taschenrechner vor. Sie werden feststellen, daß auch hier eine gewisse Reihenfolge eingehalten werden muß:

1. Suche Taschenrechner.
2. Schalte Rechner ein.
3. Wenn Batterie leer, dann Punkt 12.
4. Drücke die Taste '4'.
5. Drücke die Taste '8'.
6. Drücke das Operationzeichen für Divison.
7. Drücke die Taste '1'.
8. Drücke die Taste '2'.
9. Drücke die Taste '='.
10. Lese Ergebnis ab.
11. Speichere Ergebnis im Kleinhirn.
12. Schalte Rechner aus.

Erstaunlich, in wieviel Einzelschritte doch eine solche simple Tätigkeit zerlegt werden kann.

Sie erkennen, daß alle Schritte numeriert sind, um die Reihenfolge festzulegen. Doch dies ist nicht der einzige Grund. Der Schritt Nummer 3 schneidet eine wichtige Programmierlogik an, den Sprungbefehl. Wenn eine bestimmte Bedingung erfüllt ist (Batterie leer?), verzweigt das Programm zum Schritt 12. Diese Zahl ist die sogenannte Adresse des Befehls. Ohne Zeilennummer wäre es also nicht möglich, bestimmte Befehle gezielt anzuspringen.

Wir haben im vorherigen Kapitel bereits einen Befehl kennengelernt, den PRINT-Befehl, mit dem Daten auf den Bildschirm ausgegeben werden. Diesen Befehl setzten wir jedoch nur im Direkt-Modus ein, d.h. der Befehl wird nach

dem RETURN direkt ausgeführt. Wir wollen nun eine Folge von drei PRINT-Befehlen als Programm codieren. Was ist zu tun? Richtig, jeder Befehl erhält eine Zeilennummer, die bestimmt, in welcher Reihenfolge das Programm abgelaufen wird.

```
DIE ZEILENNUMMERN DÜRFEN BEIM COMMODORE 128
IM BEREICH VON 0 BIS 63999 LIEGEN. DIE SCHRITT-
WEITE IST OHNE BEDEUTUNG.
```

Die Schrittweite bestimmt, um welche Zahl die Folgezeile größer als die vorherige Zeile ist. So kann ein vierzeiliges Programm z.B. aus den Zeilennummern 1, 8, 10, 20 bestehen. Auch die Reihenfolge 100 ,200 ,300 ,400 ist möglich. Die Schrittweite darf also innerhalb eines Programms variieren.

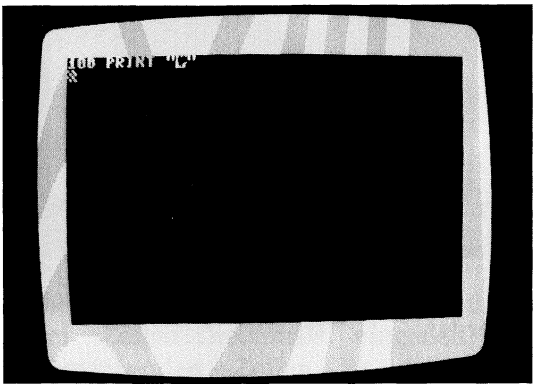
Doch wozu ist eine Schrittweite von größer als 1 nützlich. Ist doch klar: Wenn zwischen zwei Zeilen in einem Programmn noch eine Zeile eingefügt werden soll, so muß die Schrittweite zwischen diesen beiden Zeilen größer als 1 sein, da Zeilennummern immer ganzzahlig sind. Zwischen der Zeile 11 und der Zeile 12 kann keine Zeile mehr eingefügt werden. Zwischen den Zeilennummern 11 und 15 jedoch gibt es mehrere Möglichkeiten, eine Zeile "einzumogeln". Diese Zeile erhält die Nummer 12, 13 oder 14 und wird automatisch zwischen die Zeilen 11 und 15 eingeordnet. In der Praxis hat sich eine Schrittweite von 10 bewährt.

Doch nun zu unserer Aufgabe, die wie folgt aussehen soll:

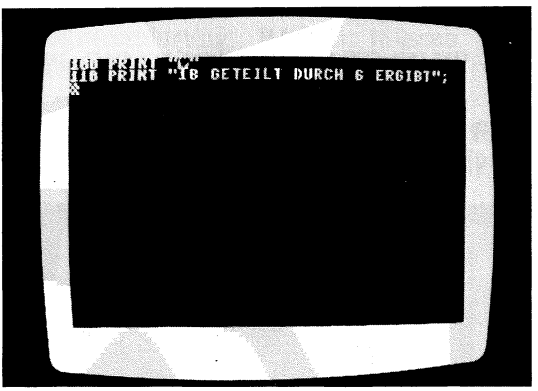
1. Lösche den Bildschirm.
2. Gebe den Text "18 GETEILT DURCH 6 ERGIBT:" aus.
3. Gebe das Ergebnis unmittelbar hinter dem Text aus.

Zur Lösung der Aufgabe sollen drei BASIC-Zeilen verwendet werden. Wie geht man nun diese Aufgabenstellung an? Zunächst muß die erste Zeilennummer ausgewählt werden. In unserem Beispiel soll dies die Zeile 100 sein. Wir geben dazu die Zahl 100 gefolgt von dem Befehl zum Löschen des

Bildschirms ein. Erinnern wir uns: Das Steuerzeichen zum Bildschirmlöschen ist SHIFT mit CLR/HOME. Diese Zeile sollte dann wie folgt aussehen:

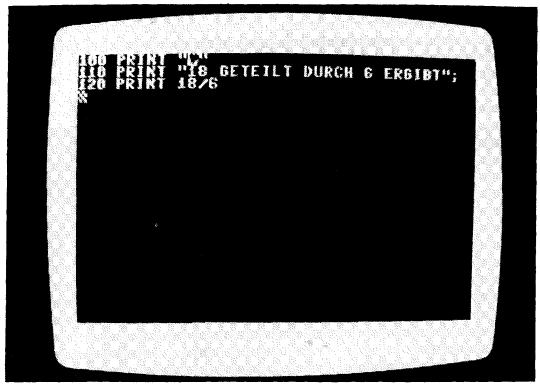


Dies ist nun die erste Zeile unseres Programms. Die zweite Zeile soll eine Zeichenkette ausgeben. Dies erfolgt bekanntlich ebenfalls mit dem Befehl PRINT. Wir arbeiten mit einer Schrittweite von 10. Die zweite Zeile erhält demnach die Nummer 110. Geben Sie diese Zeile nun ein.



Hinter den String wird hier ein Semikolon gesetzt, damit das folgende PRINT noch in dieser Zeile, also hinter dem Text ausgegeben wird.

Die dritte Zeile sollten Sie nun alleine ermitteln. Es soll 18/6 berechnet und ausgegeben werden.



Nun ist das Programm fertig und braucht nur noch gestartet zu werden.

Programmstart

Der 128er hat die soeben eingegebenen Programmzeilen nicht nur auf dem Bildschirm, sondern auch in seinem BASIC-Speicher festgehalten. Auch wenn Sie den Bildschirm löschen geht das Programm nicht verloren. Es kann zu jeder Zeit gestartet werden.

DER BEFEHL 'RUN' STARTET EIN BASIC-PROGRAMM

Geben Sie nun den Befehl 'RUN' ein und beachten Sie, was auf dem Bildschirm geschieht.

Dieses Programm ist zwar noch sehr klein, für den Anfänger jedoch ein Erfolgserlebnis.

Sie können dieses Programm nun so oft starten, wie es Ihnen gefällt. Überzeugen Sie sich selbst: Geben Sie nochmals 'RUN' ein und nochmals und...

Programmänderung

Es kann durchaus vorkommen, daß an einem Programm Änderungen vorgenommen werden müssen. Stellen Sie sich vor, Sie möchten mit dem zuletzt eingegebenen Programm nicht 18/6, sondern 21/7 berechnen. Dazu müssen Sie nicht etwa alle drei Zeilen nochmals eingeben, sondern Sie können die bestehenden Zeilen an den entsprechenden Stellen abändern. Dazu müssen Sie aber erst wieder das Programm auf dem Bildschirm holen.

```
DER BEFEHL 'LIST' ZEIGT DIE PROGRAMMZEILEN
AUF DEM BILDSCHIRM AN
```

"Schon wieder ein neuer Befehl" werden Sie vielleicht denken. Doch dies wird mit Sicherheit nicht der Letzte sein. Zur Programmierung BASIC müssen Sie mit den wichtigsten Befehlen vertraut sein.

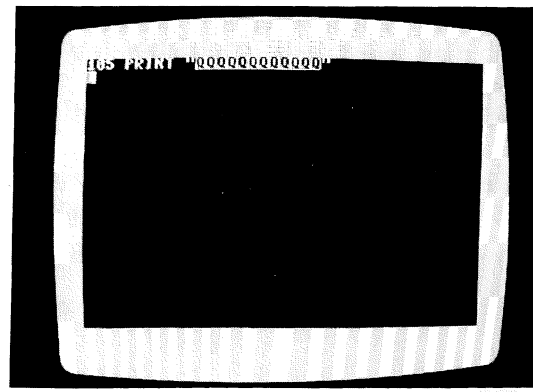
Geben Sie nun den Befehl 'LIST' gefolgt von der Taste RETURN ein und beobachten, was geschieht. Ihre soeben eingegebenen Programmzeilen erscheinen auf dem Bildschirm.

Was Sie noch wissen sollten: Wenn größere Programme aufgelistet werden, so "scrollt" der Bildschirm. Scrollen bedeutet, es werden von unten Zeilen nachgeschoben, wobei gleichzeitig die obersten Zeilen verschwinden. Um dieses Scrollen zu verlangsamen, können Sie die Taste 'CTRL' gedrückt halten. Gänzlich abgebrochen werden kann die Auflistung mit der Taste 'RUN/STOP'.

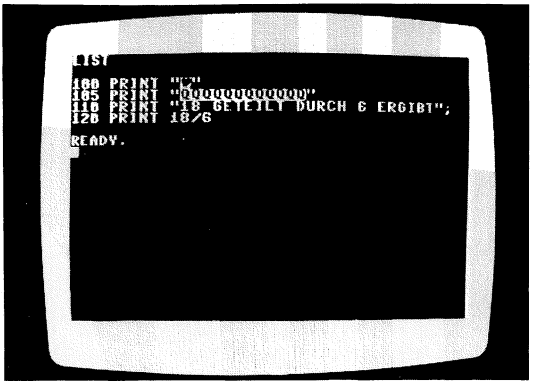
Kommen wir nun auf die Programmänderung zurück. Um die Zeilen nun zur Berechnung von 21/7 abzuändern, benutzen

Sie den Bildschirmeditor. Bewegen Sie zunächst den Cursor auf die entsprechende Zeile. Danach ändern Sie die Zeile, indem Sie die Zahl 18 gegen 21 und die Zahl 6 gegen 7 austauschen. Sie werden dazu einfach überschrieben. Schließlich geben Sie diese Zeilen nochmals mit der Taste RETURN als Programmzeilen ein. Sie können die zu ändernden Zeilen aber auch neu eingeben. Die neu eingegebenen Zeilen ersetzen dann die ursprünglich vorhandenen. In unserem Beispiel belassen wir es allerdings bei 18/3.

Zu Anfang wurde kurz das Einfügen von Zeilen angeschnitten. Dies ist bekanntlich nur möglich, wenn ein Zeilenabstand von größer als 1 vorhanden ist. In dem kleinen "Dreizeiler" ist dies berücksichtigt worden. Wenn das Ergebnis z.B. in der mittleren Bildschirmzeile ausgegeben werden soll, so muß nach der Zeile 100 und vor der Zeile 110 ein PRINT mit den Steuerzeichen eingefügt werden. Diese Steuerzeichen sind 12 mal "CURSOR UNTEN". Dazu geben Sie einfach den entsprechenden Befehl mit der Zeilennummer 105 ein.



Wenn Sie das Programm anschließend listen, werden Sie feststellen, daß diese Zeile erwartungsgemäß eingefügt wurde.



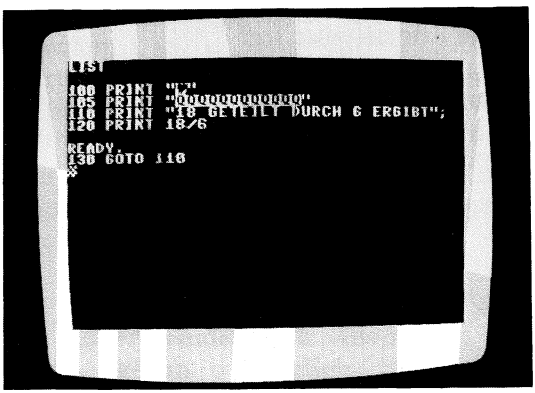
Verzweigung

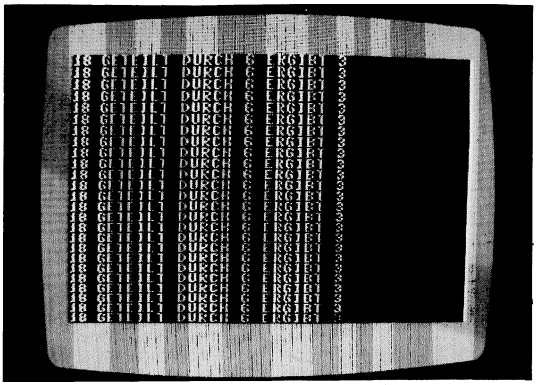
Programme werden nur selten wie im letzten Beispiel stur von vorne nach hinten abgearbeitet. Oft werden, meist zu bestimmten Bedingungen, Zeilen übersprungen und an anderer Stelle fortgesetzt. Hier ist ein Sprung erforderlich.

DER BEFEHL 'GOTO' SPRINGT ZU EINER ANGEGEBENEN ZEILE

In BASIC gibt es den Befehl 'GOTO', der eine hinter dem Befehl angegebene Zeilennummer anspringt.

Lassen Sie uns dies gleich in die Praxis umsetzen. Bekanntlich endet das Programm nach der Zeile 120. Was geschieht, wenn wir danach, z.B. in Zeile 130, wieder in die Zeile 110 springen? Ich behaupte, es wird ununterbrochen "21 GETEILT DURCH 7 ERGIBT 3" ausgegeben. Doch sehen Sie selbst. Geben Sie die Zeile '130 GOTO 110' ein und starten das Programm mit 'RUN'.





Wir haben mit der Zeile 130 eine sogenannte Endlosschleife erstellt, da das Programm ständig die Zeilen 110 bis 130 durchläuft. Wenn wir nun nicht eingreifen, läuft das Programm noch stundenlang so weiter.

```
DIE TASTE 'RUN/STOP' UNTERBRICHT DEN PROGRAMMABLAUF
```

Drücken Sie also nun die Taste 'RUN/STOP'. Das Programm hält an und meldet 'BREAK IN ...". In welcher Zeile das Programm nun gestoppt wird, ist rein zufällig.

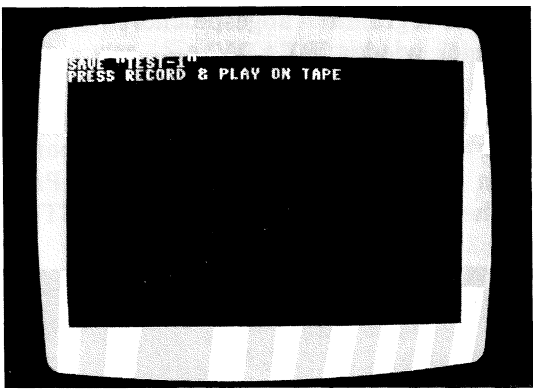
Soll der Programmablauf fortgesetzt werden, so ist auch hierfür ein Befehl vorgesehen. Fortsetzen heißt auf englisch "continue". Da BASIC-Befehle der englischen Sprache angelehnt sind, heißt der entsprechende Befehl 'CONT'.

```
DER BEFEHL 'CONT' SETZT DEN PROGRAMMABLAUF
NACH EINER UNTERBRECHUNG FORT
```

Geben Sie nun diesen Befehl ein. Das Programm befindet sich wieder in der Endlosschleife und kann wiederum mit der Taste 'RUN/STOP' unterbrochen werden.

Speichern und Laden von Programmen

Wenn Sie den Rechner nun ausschalten, so ist das Programm verloren. Da Sie sicherlich nicht nach jedem Einschalten des Geräts das gewünschte Programm eintippen möchten, können die Programme auf ein externes Speichermedium abgelegt werden. Beim 128er ist dies entweder die Datassette oder die Diskettenstation. Speichern wir zunächst das Programm auf der Datassette. Legen Sie eine Kassette ein und spulen diese zurück. Nun bestimmen Sie einen Namen für das Programm, der maximal 16 Zeichen lang sein darf. Wenn Sie nun den Befehl 'SAVE "....."' eingeben, so fordert der Rechner Sie auf, den Recorder auf Aufnahme zu schalten. Nach dem Befehl 'SAVE' geben Sie den Programmnamen in Anführungszeichen an. Wenn wir z.B. den Namen "TEST-1" wählen, wäre der Ablauf folgender:

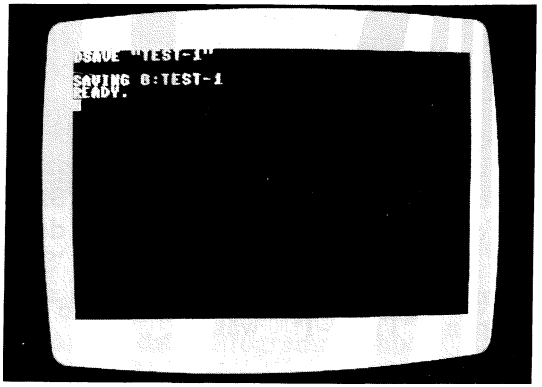


Wenn Sie nun anweisungsgemäß die Aufnahmetaste des Recorders drücken, so erlischt der Bildschirm und das Programm wird auf die Kassette gespeichert. Nach dem Speichern erscheint wieder die Meldung 'READY'.

Das Programm kann nun beliebig von der Kassette in den Rechner geladen werden. Dazu verfahren Sie genauso wie im 3. Kapitel beschrieben.

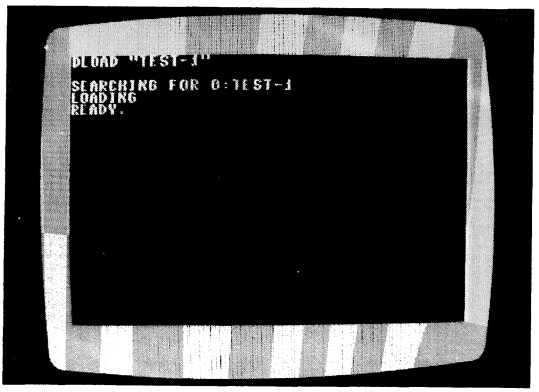
Der Befehl, um das Programm auf eine Diskette abzuspeichern, unterscheidet sich kaum von dem für die Kassette. Sie geben lediglich DSAVE"name" als

Speicherbefehl an.
Wenn Sie das Programm unter dem gleichen Namen auf einer Diskette abspeichern möchten, so geben Sie den Befehl 'DSAVE"TEST-1"' ein. Was dann weiter geschieht, sehen Sie im folgendem Bild:



Nachdem das Programm abgespeichert ist, meldet der Rechner auch hier wieder 'READY'.

Wenn Sie dieses Programm nun wieder von der Diskette in den Rechner laden möchten, so geben Sie den entsprechenden Ladebefehl ein. Da der Programmname "TEST-1" ist, heißt dieser Befehl 'DLOAD"TEST-1"'.



Beachten Sie, daß auf einer Diskette nicht zwei Programme mit ein und demselben Namen angelegt werden können!

Löschen eines Pogramms

Wie bereits bekannt, erlischt das Programm beim Ausschalten des Rechners. Doch auch ohne Ausschalten kann das Programm gelöscht werden.

```
DER BEFEHL 'NEW' LÖSCHT DAS IM SPEICHER
BEFINDLICHE PROGRAMM
```

Geben Sie nun den Befehl 'NEW' ein, so wird das Programm gelöscht. Man sollte natürlich das zu löschende Programm vorher abspeichern.

Für alle skeptischen Leser: Wenn Sie nach dem 'NEW' den Befehl 'LIST' eingegeben, so werden Sie feststellen, daß das Programm wirklich gelöscht wurde. Es werden keine Zeilen mehr angezeigt. Auch der Befehl 'RUN' ist nun wirkungslos.

Dies ist alles, was Sie an Vorwissen für die folgende Einführung in BASIC benötigen.

BASIC EINFÜHRUNG

Dieses Kapitel soll eine Einführung in die Programmiersprache BASIC darstellen. Dabei sollen nicht alle BASIC-Befehle stur nacheinander beschrieben werden, sondern es wird eine Adressenverwaltung Schritt für Schritt aufgebaut, in der die Befehle dann zum gegebenen Zeitpunkt beschrieben und eingesetzt werden. Der Leser dieses Buches soll durch dieses Kapitel nicht zum perfekten Programmierer werden, sondern einen intensiven Einblick in die praxisnahe Programmierung erhalten. Dieser Einblick bietet ihm dann beste Voraussetzungen, sich durch weitere Fachbücher weiterzubilden.

Problembeschreibung zur Adressenverwaltung

Eine Adressenverwaltung ist eines der beliebtesten Programme auf dem Heimcomputer. Die breite Einsatzfähigkeit trägt wesentlich dazu bei. Es gibt kaum jemand, der dieses Programm nicht einzusetzen weiß. Zwar ist eine Adressenverwaltung mit nur wenig Adressen per Heimcomputer nicht unbedingt effektiver als die herkömmliche Methode mit dem Adressenbüchlein, aber erheblich eindrucksvoller. Wenn sich jedoch sehr viele Adressen ansammeln, so sind sie in einem Heimcomputer bestens aufgehoben.

Welche Möglichkeiten sollte ein solches Programm bieten? Nun, auf jeden Fall müssen Adressen ein- und ausgegeben werden.

Bevor wir weitere Ansprüche an das Programm stellen, muß noch etwas klargestellt werden. In einem Programm, das Daten verwaltet, unterscheidet man zwischen zwei grundsätzlichen Bestandteilen:

1. **Das Programm**
2. **Die Daten**

Die Daten sind nicht Bestandteil des Programms. Dies wäre zwar möglich, aber diese Daten können dann nur vom Programmierer und nicht vom Anwender verwaltet werden. Eine Änderung der Daten hätte gleichzeitig eine Änderung des Programms zur Folge, was wir sofort wieder vergessen können.

Das Programm befindet sich also auf dem externen Speichermedium (Kassette oder Diskette) und wird bei Bedarf in den Rechner eingeladen. Aber was ist nun mit den Daten? Sie fallen erstmalig bei der Ersterfassung an. Wohin dann damit?

Wir müssen eine Datei organisieren. Eine Datei ist eine Ansammlung von Daten auf einem externen Speichermedium. Es gibt zwar mehrere Dateiorganisationsformen, wir beschäftigen uns aber nur mit der einfachsten Methode der sequentiellen Datei. Sequentiell bedeutet, daß die Daten hintereinander angeordnet sind. Die zwei wichtigen Bestandteile unserer Adressenverwaltung sind nun

1. **Das Programm**
2. **Die Datei**

Dateiorganisation

Wenn ich jetzt fragen würde, welcher Teil zuerst organisiert werden muß, werden viele antworten, "das Programm". Die Antwort ist aber falsch. Zuerst muß man sich im Klaren sein, was wie wo abgespeichert wird. Über das "wie" sind wir uns im klaren, sequentiell soll gespeichert werden. Das "wo" ist noch nicht geklärt worden. Es gibt zwei Möglichkeiten:

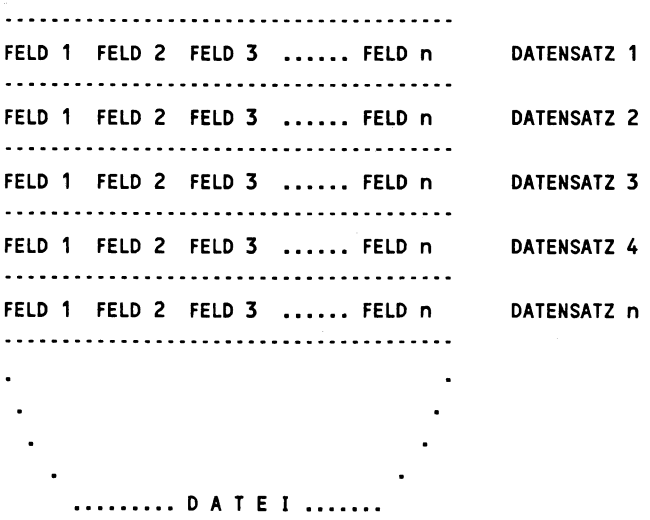
1. **Speicherung auf Datassette**
2. **Speicherung auf Diskette**

Da sowohl Kassetten- als auch Diskettenlaufwerke bei den Lesern vorzufinden sind, wird zunächst die Diskettenspeicherung behandelt. Im Anhang finden Sie entsprechende Änderungen zur Speicherung auf eine Kassette.

Bleibt nun noch die Frage, was gespeichert werden soll. "Adressen natürlich" werden viele spontan reagieren, doch sind wir uns überhaupt im Klaren, was alles zu einer Adresse gehören soll? Natürlich nicht. Sammeln wir doch einmal alles, woraus sich eine Adresse zusammensetzen könnte:

- **Anrede**
- **Vorname**
- **Name**
- **Strasse**
- **Postleitzahl**
- **Ort**
- **Telefonnummer**
- **Bemerkung**

Diese einzelnen Bestandteile einer Adresse nennt man DATENFELD. Man spricht so z.B. vom Feld "Anrede". Die Gesamtheit der Datenfelder ergibt den DATENSATZ. Jeder Datensatz besteht also in unserem Beispiel aus diesen 8 Feldern. Alle Datensätze zusammen ergeben schließlich die Datei. Sehen wir uns diese Struktur einmal in folgendem Bild an:



Die Hierarchie ist also DATEI - DATENSATZ - DATENFELD

Rechnerinterne Speicherung der Daten

Mit einer sequentiellen Datei läßt sich zwar sehr bequem arbeiten, jedoch gibt es hier auch Nachteile gegenüber anderen Organisationsformen. Stellen Sie sich vor, die gesamte datei mit den Adressen befindet sich auf einer Kassette. Sie möchten nun eine ganz bestimmte Adresse aus dieser Datei haben. Hier tritt dann das Problem auf. Sie können nicht aus einer Datei nur einen Datensatz lesen. Der Kassettenrekorder und auch die Floppystation ist nicht in der Lage, aus einer sequentiellen Datei einzelne Datensätz zu lesen. Wie kann man nun auf die Adressen zugreifen?

Um auf einen Datensatz zuzugreifen, muß die gesamte Datei in den Rechner eingelesen werden. Eine sequentielle Datei darf demnach nicht größer als der vorhandene Speicherplatz des Rechners sein. Ein Vorteil stellt sich jedoch wieder heraus. Wenn die Datei einmal eingelesen ist, so kann im Rechner blitzschnell auf die Datensätze zugegriffen werden. Für den gesamten Programmablauf gilt also folgende Regelung:

1. **Datei laden**
2. **Datensätze lesen, ändern, löschen .....**
3. **Datei speichern**

Nach dem Start des Programm muß also zuerst die Adressendatei komplett eingelesen werden. Danach können die Datensätze verarbeitet werden. Vor Beendigung des Programms muß die Datei jedoch wieder gespeichert werden, sofern sie geändert wurde. Wurden die Datensätze jedoch nicht geändert und auch keine gelöscht oder hinzugefügt, so stimmt die Datei noch mit der ursprünglich eingeladenen Datei überein. Sie braucht also nicht wieder gespeichert zu werden.

Variablen

Wie bekannt, werden die Datensätze im Rechner gespeichert. Die Frage ist nur wo und wie werden sie gespeichert?

```
RECHNERINTERNE DATEN WERDEN IN VARIABLEN GESPEICHERT
```

Variablen sind also das Zauberwort. Variablen sind Speicherbereiche im 128er, die mit einem Namen versehen werden. Durch Angabe dieses Namens kann auf diese Speicherbereiche zugegriffen werden. Wir haben schon zwei verschiedene Datentypen kennengelernt. Da gibt es numerische Daten (Zahlen) und alphanumerische Daten (Strings, Text). Stringvariablen werden durch das Zeichen '$' hinter der Variablenbezeichnung gekennzeichnet. Woraus setzen sich nun die Bezeichnungen der Variablen zusammen? Nun zunächst sind die Bezeichnungen höchsten zweistellig. Das erste Zeichen MUß ein Buchstabe sein, das zweite Zeichen DARF eine Zahl sein. Variablennamen für numerische Daten sind:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

AA AB AC AD AE AF AG AH AI AJ AK AL AM AN ..... AZ
BA BB BC BD BE BF BG BH BI BJ BK BL BM BN ..... BZ
.
.
.
ZA ZB ZC ZD ZE ZF ZG ZH ZI ZJ KK ZL ZM ZN .......ZZ

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9
.
.
.
Z0 Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9
```

Hier gibt es wenige Ausnahmen. Zweistellige BASIC-Befehle oder Funktionen dürfen nicht verwendet werden:

OR IF FN TI ST

Beispiele numerische Variablen:

A
XY
I1
SD

Beispiele Stringvariablen:

XC$
X$
A1$
ZZ$

Vom BASIC Interpreter des CBM 128 werden jedoch auch längere Variablennamen angenommen. Zu beachten ist jedoch, daß nur die ersten zwei Zeichen als Variable anerkannt werden. Der Variablenname "BETRAG" wird als Variable "BE" angelegt! Somit ist die Variablenbezeichnung "BETRAG" vollkommen identisch mit "BEZEICHNUNG". Dies kann zu schwer lokalisierbaren Fehlern führen.

Bei der Auswahl der Variablen sollten Sie sinnvolle Abkürzungen verwenden. So wird die Variable zum Speichern des Vornamens z.B. "VN$" genannt. Den Einkaufpreis würde man demnach "EK" nennen, usw.

Variablenverarbeitung

Wenden wir uns nun wieder nach langer Zeit dem Rechner zu. Wir werden nun Variablen einrichten, verarbeiten und ausgeben.

Die Speicherung von numerischen Daten in Variablen z.B. ist eigentlich ein Kinderspiel. Nehmen wir an, die Zahl 45.12 soll in der Variablen EK gespeichert werden. Der entsprechende Befehl ist dann

EK=45.12

Ist doch einfach, oder? Soll diese Variable ausgegeben werden, so lautet der Befehl

PRINT EK

geben Sie beide Befehle nun einmal in den Rechner ein. Soll die Variable EK wieder gelöscht werden, so geben Sie sinngemäß den Befehl

EK=0

ein. Die Variablen können aber auch für Rechenoperationen benutzt werden. Geben wir einmal den doppelten Wert von EK aus. Wenn Sie die Variable gelöscht haben, so geben Sie erneut 'EK=45.12' ein.

PRINT EK*2

Der Rechner verdoppelt EK und gibt das Ergebnis (90.24) aus. Die Variable selbst behält den Wert 45.12. Was aber ist zu tun, wenn die Variable EK verdoppelt werden soll? Da EK einen neuen Wert erhalten soll, beginnt der entsprechende Befehl mit 'EK='. Der gesamte Befehl zur Verdoppelung von EK ist:

EK=EK*2

Es wird immer erst der Wert rechts vom Gleichheitszeichen errechnet und dann in der Variablen links vom Gleichheitszeichen abgelegt.

Stringvariablen werden auf ähnliche Art und Weise eingerichtet. Speichern wir nun den String "FRANKFURT" in die Stringvariable S$. Der Befehl dazu lautet

S$="FRANKFURT"

Auch Stringvariablen werden mit dem Befehl PRINT auf dem Bildschirm ausgegeben. Sehen Sie selbst:

PRINT S$

Was geschieht? Natürlich, der in S$ gespeicherte String wird ausgegeben. Soll diese Stringvariable wieder gelöscht werden, so muß ein sogenannter Leerstring in S$ gespeichert werden.

```
S$=""
```

Ein Leerstring besteht also aus zwei unmittelbar aufeinanderfolgenden Anführungszeichen.

Selbstverständlich kann mit diesen Strings nicht gerechnet werden. Auch nicht, wenn eine Zahl als String abgelegt wird. Das heißt, wenn Sie z.B. X$="123" eingeben würden, so können Sie trotzdem nicht ohne weiteres mit X$ rechnen.

Strings können jedoch verkettet werden. Die folgende Befehlsfolge bestätigt es:

```
A$="DUESSELDORF"
B$="FRANKFURT"
C$=A$+B$
PRINT C$
```

Zuerst wurden die beiden Städte in A$ und B$ gespeichert. Danach wurde ein dritter String (C$) gebildet, der beide Strings aufnimmt. Der vierte Befehl hat den String C$ dann ausgegeben.

Es wäre sinnvoll, zwischen den beiden Städten ein Leerzeichen zu setzen. Dazu müßte der dritte Befehl entsprechend geändert werden:

```
C$=A$+" "+B$
```

Es wurde also ein String mit einem Leerzeichen zwischen beide Strings gesetzt.

Wir haben jetzt die zwei wichtigsten Variablentypen kennengelernt. Die numerische Variable und die Stringvariable, die durch ein zusätzliches '$' gekennzeichnet ist. In solchen Variablen können auch Adressen gespeichert werden. Dies ist jedoch nicht sinnvoll. Zur Speicherung von mehreren gleichartig aufgebauten Datensätzen im Rechner werden andere Variablen benötigt.

Tabellen

Tabellen werden in der Datenverarbeitung oft eingesetzt. Wenn Sie mehrere Daten einer Gruppe im Rechner speichern wollen, so ist es sehr umständlich, jedem Datenelement dieser Gruppe einen Variablennamen zu geben. In diesem Fall erhalten alle Datenelemente den gleichen Variablennamen, der jedoch zusätzlich gekennzeichnet ist. Nehmen wir das Beispiel einer Gruppe von fünf Städtenamen:

BONN
PARIS
LONDON
ROM
MADRID

Um diese fünf Städte im Rechner zu speichern, wären nach der bisherigen Methode 5 Variablen erforderlich. Wenn wir diese Datengruppe jedoch in einer Tabelle (auch Array genannt) speichern würden, so wäre ein Variablenname ausreichend. Da die Datenelemente jedoch unterschieden werden müssen, werden sie gekennzeichnet. Dazu wird hinter der Variablen eine in Klammern gesetzte Zahl angehängt. Diese Zahl bezeichnet man als Index. Das erste Datenelement erhält so den Index 1, das zweite den Index 2, usw. Der Index fängt zwar bei 0 und nicht bei 1 an, jedoch geht dadurch die Übersicht verloren, wenn das erste Datenelement den Index 0 erhält. Man kann diesen Index 0 ignorieren. Dies vergeudet zwar Speicherplatz, doch werden gleichzeitig Fehlerquellen vermieden.

Eine Tabellen- oder Arrayvariable ist z.B. A$(1). Auch XY(212) ist eine numerische Arrayvariable. Der Index ist also beliebig groß anzusetzen. Es gibt hier nur zwei Dinge zu beachten:

1. **Übersteigt der Index die Zahl 10, so muß für diese Tabelle Speicherplatz reserviert werden.**

2. **Die Tabelle darf nicht so groß definiert werden, daß sie den Speicher überschreitet.**

Es werden also Tabellen bis zum Index 10 selbständig vom Rechner verwaltet. Doch wie werden größere Arrays definiert? Hierfür gibt es einen speziellen BASIC-Befehl, den Befehl 'DIM'.

Problem:	Dimensionierung von Arrays
Befehl:	DIM v1 (n1,n2,n3...)
Parameter:	v1 - Variablenname des Arrays n1,n2... - maximaler Index der einzelnen Dimensionen
Beispiel:	DIM D$(20) Es soll ein Array mit der Bezeichnung D$ bis zum Index 20 eingerichtet werden.
Bemerkung:	Jedes Array darf im Programm nur EINMAL dimensioniert werden, sonst Fehlermeldung "REDIM'D ARRAY ERROR"

Wenn der maximale Index z.B. 5 sein soll, so ist keine DIM-Anweisung notwendig, da bis zum Index 10 der Rechner dies selbst durchführt. Die Sache hat nur einen kleinen Haken: Wird nur bis zum Index 5 gearbeitet, so wird auch Platz für den Index 6-10 reserviert, der eigentlich nicht benötigt wird. Hier kann man Abhilfe schaffen, indem eine DIM-Anweisung mit dem Index 5 eingesetzt wird. Der Rechner reserviert dann auch nur Platz bis zum 5. Index.

Doch nun zurück zu unseren fünf Städtenamen, die in einer Tabelle abgelegt werden sollen. Wenn die Variable D$ verwendet wird, werden die Städte mit den folgenden Zuweisungen gespeichert.

```
10 D$(1)="BONN"
20 D$(2)="PARIS"
30 D$(3)="LONDON"
40 D$(4)="ROM"
50 D$(5)="MADRID"
```

Man könnte hier auch zuvor die Anweisung 'DIM D$(5)' geben, wenn der Index 6-10 nicht benötigt wird.

Wird ein Index benutzt, der über die reservierte Tabelle hinausschießt, so wird die Fehlermeldung "BAD SUBSCRIPT ERROR" gegeben. Den Index bezeichnet man auch als Subskript.

In einem weiteren Beispiel sollen nun die fünf Länder der zuvor gespeicherten Hauptstädte ebenfalls in einer Tabelle abgelegt werden. Die Tabelle erhält den Namen L$. Folgende Zuweisungen müssen gemacht werden:

```
 60 L$(1)="BRD"
 70 L$(2)="FRANKREICH"
 80 L$(3)="ENGLAND"
 90 L$(4)="ITALIEN"
100 L$(5)="SPANIEN"
```

Nun haben wir zwei Tabellen erstellt, deren Index in unmittelbarem Zusammenhang stehen. Das heißt, D$(1) enthält die Hauptstadt von L$(1). Allgemein kann gesagt werden, D$(X) enthält die Hauptstadt von L$(X).

Der besondere Reiz an Tabellen ist, daß als Index nicht nur eine Zahl, sondern auch Variablen oder beliebige Ausdrücke eingesetzt werden können. Eine Arrayvariable mit der Bezeichnung 'DS$(X-2*I+13)' ist somit durchaus möglich. Der Vorteil dieser Indizierung ist unverkennbar.

Diese beiden soeben erstellten Tabellen bezeichnet man als eindimensional, da Sie nur einen Index besitzen. Nun können Tabellen aber beliebig viele Indizes erhalten. Anstatt der beiden zuvor aufgebauten Tabellen kann auch eine zweidimensionale Tabelle erstellt werden. Beachten Sie zunächst die entsprechende Befehlsfolge:

```
 10 DIM D$(1,5)
 20 D$(0,1)="BONN"
 30 D$(0,2)="PARIS"
 40 D$(0,3)="LONDON"
 50 D$(0,4)="ROM"
 60 D$(0,5)="MADRID"
 70 D$(1,1)="BRD"
 80 D$(1,2)="FRANKREICH"
 90 D$(1,3)="ENGLAND"
100 D$(1,4)="ITALIEN"
110 D$(1,5)="SPANIEN"
```

Die DIM-Anweisung in Zeile 10 ist zwar nicht erforderlich, da der Rechner ohnehin 'DIM D$(10,10)' ausführen würde, jedoch würden mit dieser Anweisung 11*11 (121) Plätze reserviert. Wir benötigen jedoch nur 10 Tabellenplätze. Bei zwei- und mehrdimensionalen Tabellen sollte man die Größe der Tabellen möglichst genau bestimmen, da sonst viel Speicherplatz vergeudet wird.

Wir haben nun eine zweidimensionale Tabelle, in der der erste Index die Datengruppe (Stadt oder Land) identifiziert und der zweite Index den Tabellenplatz. Man kann diese Tabelle auch als Matrix darstellen:

	1	2	3	4	5
0	BONN	PARIS	LONDON	ROM	MADRID
1	BRD	FRANKREICH	ENGLAND	ITALIEN	SPANIEN

Kommen wir nun wieder zu unserer Adressenverwaltung zurück. Eine solche zweidimensionale Tabelle ist ideal für die Aufnahme der sequentiellen Adressendatei. Der erste Index numeriert die Datensätze, der zweite Index numeriert die Datenfelder innerhalb der Datensätze. Da unsere Datensätze 8 Felder enthalten, ist der zweite Index maximal 8. Der erste Index, also die Anzahl der Adressen, ist wahlweise. Legen wir uns hier auf maximal 200 Adressen fest. Der Name der Tabelle soll D$ sein. Die DIM-Anweisung zum Einrichten dieser Tabelle ist dann

```
DIM D$(200,8)
```

Wie die Tabelle verwaltet wird, erfahren Sie an späterer Stelle.

Dateneingabe über Tastatur

In den bisherigen Übungen haben wir nur Daten verarbeitet und ausgegeben. Interessant wird es erst, wenn dem Programm Daten über die Tastatur übergeben werden. Der Befehl dazu heißt INPUT. Doch INPUT alleine hat keine Wirkung. Dem Befehl folgen zwei Parameter: Ein String, der vor dem eigentlichen INPUT ausgegeben wird und eine Variable, in der die eingegebenen Daten gespeichert werden sollen. Wichtig: Der Befehl INPUT kann nicht im Direktmodus, sondern nur im Programm eingesetzt werden. Doch beachten Sie zunächst die Befehlsbeschreibung:

Problem:	Dateneingabe über Tastatur
Befehl:	INPUT "string";v1
Parameter:	"string" - beliebige Zeichenkette v1 - Variable, in der die Eingabe gespeichert wird.
Beispiel:	INPUT "ZAHL";X Es wird der String "ZAHL" ausgegeben und anschließend ein numerischer Wert von der Tastatur eingelesen und in der Variablen X gespeichert.
Bemerkung:	Der Befehl kann nicht im Direktmodus eingesetzt werden.

Geben Sie nun den Befehl NEW ein, der ein zuvor im Speicher befindliches Programm löscht. Die folgenden BASIC-Zeilen demonstrieren den Einsatz des Befehls INPUT:

```
10 PRINT"KREISBERECHNUNG"
20 PRINT"--------------"
30 INPUT"DURCHMESSER  ";D
40 PRINT"KREISFLAECHE:"(D/2)^2*3.14
```

Starten Sie dieses Programm mit dem Befehl RUN. In Zeile 30 wird der Durchmesser des zu berechnenden Kreises abgefragt. Die Eingabe wird stets mit RETURN abgeschlossen. Hinter dem String nach dem INPUT muß ein Semikolon angegeben werden. Der String kann aber auch weggelassen werden. Das folgende Programm bestätigt es:

```
10 PRINT"KREISBERECHNUNG"
20 PRINT"--------------"
30 INPUT D
40 PRINT"KREISFLAECHE:"(D/2)^2*3.14
```

Sie brauchen natürlich nicht das gesamte Programm neu einzugeben, sondern nur die Zeile 30 entsprechend zu ändern. Sie bemerken, daß der String vor dem INPUT nur dokumentierende Aufgaben hat. Die Dokumentation des INPUT kann aber auch ein PRINT in der Zeile zuvor übernehmen:

```
10 PRINT"KREISBERECHNUNG"
20 PRINT"--------------"
25 PRINT"DURCHMESSER  ";
30 INPUT D
40 PRINT"KREISFLAECHE:"(D/2)^2*3.14
```

Geben Sie zusätzlich die Zeile 25 ein und starten nochmals das Programm. Es läuft genauso wie die erste Version ab. Wichtig ist das Semikolon hinter dem PRINT in Zeile 25. Lassen Sie es weg, so erfolgt der INPUT-Befehl in der Zeile darunter. Probieren Sie es aus.

Auch Strings können über die Tastatur mit dem Befehl INPUT eingelesen werden. Auch hier zunächst ein Beispiel. Löschen Sie zuvor das alte Programm mit NEW.

```
10 PRINT"WIE HEISSEN SIE?"
20 INPUT N$
30 PRINT"GUTEN TAG ";N$;", WIE GEHT ES IHNEN?"
```

Aus der zeile 10 und 20 kann auch eine Zeile gebildet werden. Der String vor dem INPUT wird dann beim INPUT angegeben.

```
10 INPUT"WIE HEISSEN SIE?";N$
20 PRINT"GUTEN TAG ";N$;", WIE GEHT ES IHNEN?"
```

Diese Beispiele demonstrieren alles, was der Befehl INPUT zu bieten hat. Beachten Sie, daß das Programm auch weiterläuft, wenn nichts eingegeben, also nur RETURN gedrückt wird. Die Variablen nehmen dann den Wert 0 oder bei Strings ""(Leerstring) an.

Schleifen

Lassen Sie uns zunächst eine Übung machen, die dann später zur Schleifensteuerung führt. Es sollen fünf Namen über die Tastatur eingelesen werden. Diese Namen sollen in dem Array N$ gespeichert werden. Mit den bisher beschriebenen Befehlen würden wir folgendes Programm aufbauen:

```
10 PRINT"EINGABE DER NAMEN"
20 PRINT"----------------"
30 INPUT"NAME ";N$(1)
40 INPUT"NAME ";N$(2)
50 INPUT"NAME ";N$(3)
60 INPUT"NAME ";N$(4)
70 INPUT"NAME ";N$(5)
```

Mit diesem Programm werden jetzt nacheinander die fünf Namen eingelesen. Doch ist dies nicht sehr umständlich? Stellen Sie sich vor, es sollten 100 Namen eingelesen werden!

In der Praxis werden Sie nie einen solchen Programmierstil sehen. Man bedient sich dazu einer Schleife. Doch wie wird nun eine Schleife aufgebaut? Schauen Sie sich dazu zunächst die Beschreibung des entsprechenden Befehls an:

Problem:	Schleifensteuerung
Befehl:	FOR v1 = n1 TO n2 STEP n4
Parameter:	v1 - numerische Variable n1 - Anfangswert von v1 n2 - Endwert von v1 n3 - Inkrement, Wert um den v1 nach jedem Schleifendurchlauf erhöht erhöht wird

Beispiel:	FOR A=1 TO 20 STEP 1 Die Variable A wird nach jedem Durchlauf um 1 erhöht, bis sie den Wert 20 erhält (20 Schleifendurchläufe)
Bemerkung:	Ist das Inkrement 1, so kann der Parameter STEP weggelassen werden. Der Befehl aus dem Beispiel könnte demnach auch so eingegeben werden: FOR A=1 TO 20

Jede Schleife beginnt mit einer FOR-Anweisung. Alle anschließenden Befehle gehören dann zu dieser Schleife und werden demnach mehrmals ausgeführt. Doch wie wird das Ende der Schleife gekennzeichnet? Dafür gibt es eine weitere Anweisung, die die Aufgabe hat, die Schleifenvariable auf den Endwert zu prüfen. Ist der Endwert noch nicht erreicht, so wird die Schleifenvariable erhöht und die Schleife nochmals durchlaufen. Entspricht die Schleifenvariable jedoch dem Endwert, so wird hinter dem Befehl NEXT mit dem Programmablauf fortgefahren.

Problem:	Schleifenende festlegen
Befehl:	NEXT v1
Parameter:	v1 - Schleifenvariable
Beispiel:	NEXT A Kennzeichnet das Ende der im vorherigen Beispiel enthaltenen Schleife.
Bemerkung:	Wird die Variable v1 nicht angegeben, so bezieht sich der Befehl NEXT auf die zuletzt eingerichtete Schleife.

Mit diesen beiden Befehlen können wir nun beliebige Schleifen programmieren. Wenden wir uns wieder der anfangs beschriebenen Dateneingabe mit INPUT zu. Hier sollten fünf Namen in ein Array mit dem Index 1 bis 5 eingelesen werden. Wenn wir nun eine Schleife einsetzen, so benötigen wir nur eine INPUT-Anweisung, in der der Index als Variable, nämlich der Schleifenvariablen, eingesetzt wird. Die Schleife soll von 1 bis 5 laufen. Wie sieht nun der entsprechende FOR-Befehl mit allen Parametern aus?

```
FOR I=1 TO 5
```

Das gesamte, geänderte Programm ist folgendes:

```
10 PRINT"EINGABE DER NAMEN"
20 PRINT"----------------"
30 FOR I=1 TO 5
40 INPUT "NAME";N$(I)
50 NEXT I
```

Eine sehr elegante Lösung zum Einlesen von 5 Strings, oder? Interessant ist der genaue Ablauf nach dem Starten dieses Programms. Zuerst werden die beiden PRINT-Befehle ausgeführt. Nun beginnt die Schleife mit der Variablen I, dem Anfangswert 1, dem Endwert 5 und dem Inkrement (Erhöhungswert) 1. Erinnern Sie sich: Wird kein STEP-Parameter angegeben, so wird mit dem Inkrement 1 gearbeitet. Zu Anfang der Schleife erhält I den Wert 1. In Zeile 40 wird also die Arrayvariable mit dem Index 1 eingelesen. Anschließend prüft der Befehl NEXT, ob das Schleifenende (Variable I=5) erreicht ist. Da dies nicht der Fall ist, wird I um das Inkrement 1 erhöht und das Programm hinter dem Befehl FOR, also in Zeile 40, fortgesetzt. Die Variable I enthält nun den Wert 2. Der folgende INPUT liest also nun den nächsten Namen in dem Array mit dem Index 2 ein. So geht es nun weiter, bis der Befehl NEXT die Schleife abschließt. Hat die Variable I den Wert 4, so wird sie nochmals um 1 erhöht und die Schleife mit dem Wert I=5 durchlaufen. Anschließend wird die Variable I vom Befehl NEXT auf den Endwert (5) überprüft. Dies ist der Fall und das Programm wird hinter dem NEXT fortgesetzt, also beendet.

Wir können das Programm nun dahingehend erweitern, daß nach Eingabe der Namen alle nochmals ausgegeben werden. Dazu muß eine weitere Schleife angehängt werden:

```
60 FORI=1 TO 5
70 PRINT "NAME ";N$(I)
80 NEXT I
```

Es ist also alles nicht so kompliziert, wie Sie es vielleicht zu Anfang angenommen haben.

Was kann man noch mit einer Schleife anfangen? Nun, z.B. können Sie eine Warteschleife aufbauen, die den Programmablauf verzögert. Hier bauen Sie nur eine Schleife mit FOR auf und schließen sie unmittelbar dahinter mit NEXT ab. Nun ist es wichtig, wie oft so eine Schleife durchlaufen werden muß, um z.B. eine Warteschleife von 1 Sekunde zu erzeugen. Es kann davon ausgegangen werden, daß eine derartige Schleife 1000 mal in der Sekunde durchlaufen wird. Eine Warteschleife von ca. 3 Sekunden wird dann wie folgt aufgebaut:

```
FOR X=1 TO 3000:NEXT
```

Erinnern Sie sich: In einer Zeile können mehrere Befehle eingesetzt werden, wenn diese mit einem Doppelpunkt getrennt werden. Lassen Sie uns nun in unserem Programm eine Warteschleife zwischen der Dateneingabe und der Datenausgabe einfügen. Eine Zeilennummer, die dazwischen liegt, ist z.B. die Zeile 55. Wie sieht diese Zeile nun aus, wenn wir ca. 2 Sekunden verzögern möchten?

```
55 FOR X=1 TO 2000:NEXT
```

Lassen Sie nun einmal das programm mit LIST auf dem Bildschirm anzeigen. Sie werden sehen, daß alle nachträglich eingegebenen Zeilen ordnungsgemäß eigesetzt wurden. Starten Sie das Programm nun mit RUN, dann werden Sie die Auswirkung der Zeile 55 bemerken.

Bisher haben wir nur mit dem Inkrement 1 gearbeitet, das ja bekanntlich auch jeden anderen Wert annehmen kann.

Sollen die Namen im vorherigen Beispiel nicht von 1 bis 5, sondern umgekehrt von 5 bis 1 eingelesen werden, so ist dies durchaus möglich. Der Anfangswert der Schleife ist dann 5, der Endwert 1 und das Inkrement, also der STEP-Parameter -1. Wie müssen die Zeilen 30 und 60 nun abgeändert werden? Sicher haben Sie die Lösung längst parat:

```
30 FOR I=5 TO 1 STEP -1
60 FOR I=5 TO 1 STEP -1
```

Lassen Sie uns noch ein paar Übungen zum Befehl FOR machen. Es sollen die FOR-Befehle zu folgenden Schleifen aufgebaut werden:

	Schleifen-variable	Anfangs-wert	End-wert	Inkrement
A)	I	10	18	1
B)	XX	30	15.5	-0.5
C)	Y	590	1800	0.25
D)	B	1	10	0.3

Die Lösungen:

```
A) FOR I=10 TO 18 STEP 1
   oder
   FOR I=10 TO 18
B) FOR XX=30 TO 15.5 STEP -0.5
C) FOR Y=590 TO 1800 STEP 0.25
D) FOR B=1 TO 10 STEP 0.3
```

Noch einige Anmerkungen zum Abschluß des Kapitels. Sollten Sie mit FOR Schleifen aufbauen, die logisch falsch sind (z.B. FOR I= 10 TO 0), so wird die Schleife grundsätzlich einmal durchlaufen!
Ihr 128er kennt noch weitere Schleifenbefehle (ELSE, DO, UNTIL, WHILE) und einen eigenen Pausenbefehl (SLEEP). Sie sollten sich diese Befehle für Ihre fortgeschrittenen BASIC-Übungen vorbehalten um nicht den Überblick zu verlieren.

Erste Reaktionen des Programms

Nachdem wir so weit in BASIC fortgeschritten sind, wird es Zeit, mit dem Adressenprogramm zu beginnen. Wie sollte ein solches Programm anfangen? Wenn Sie schon einmal vergleichbare Programme bedient haben, wissen Sie sicher, was ein solches Programm nach dem Starten auf dem Bildschirm zeigt. Es meldet sich meist mit einem Programmkopf. Diesen Kopf, der nur aus wenigen PRINT-Anweisungen besteht, wollen wir zunächst erstellen. Anweisungen, die Ihnen gänzlich unbekannt sind, sollen Sie nicht daran hindern, trotzdem den Anfang des Programms einzugeben und zu starten. Diese Anweisungen werden danach beschrieben.

```
100 REM ====================
110 REM    PROGRAMMKOPF
120 REM ====================
130 PRINT CHR$(147);
140 FOR I=1 TO 40:PRINT"=";:NEXT I
150 PRINT"          ADRESSENVERWALTUNG"
160 FOR I=1 TO 40:PRINT"=";:NEXT I
170 PRINT:PRINT
```

Nachdem Sie diesen Teil des Programms nun eingegeben und gestartet haben, wollen Sie natürlich wissen, was die neuen Befehle auf sich haben. Fangen wir bei den Zeilen 100-120 an. Hier wird dieser Abschnitt des Programms dokumentiert. Damit der Rechner den dokumentierenden Teil nicht als Befehle interpretiert, werden diese Zeilen mit REM gekennzeichnet. REM bedeutet REMARKS, also Anmerkungen. Solche REM-Zeilen können Sie an beliebigen Stellen im Programm einfügen.

ASCII-Code

Nun zu der Funktion in Zeile 130. Um dies zu erklären, muß etwas weiter ausgeholt werden. Jedes Zeichen wird im Rechner codiert. Es erhält einen Wert zwischen 0 und 255. Diese Codierung ist genormt nach der amerikanischen ASCII-Norm.DieCOMMODORE-Zeichenweichengeringfügigvon

diesem Code ab. Auch die Steuerzeichen (CLR/HOME, CURSOR LINKS/RECHTS usw.) haben einen bestimmten ASCII-Code. Zum Codieren der Zeichen nach dem ASCII-Code gibt es zwei Funktionen, die zunächst beschrieben werden.

Problem:	Umwandlung eines ASCII-Codes in ein Zeichen
Funktion:	CHR$(n)
Parameter:	n - ASCII-Code (0-255)
Beispiel:	FOR I=32 TO 127: PRINT I;CHR$(I):NEXT I zeigt alle in der Betriebsart GROSS/ GRAFIK darstellbaren Zeichen und deren ASCII-Code an.
Bemerkung:	Wird ein ASCII-Code über 255 angegeben, so folgt die Fehlermeldung "ILLEGAL QUANTITY ERROR"

Problem:	Umwandlung eines Zeichens in den ASCII-Code
Funktion:	ASC("z")
Parameter:	"z" - ein Zeichen, oder eine Stringvariable, die ein Zeichen enthält
Beispiel:	PRINT ASC("A") zeigt den ASCII-Code des Buchstabens "A" (65) auf dem Bildschirm an.
Bemerkung:	- Ist die Stringvariable leer, so wird die Fehlermeldung "ILLEGAL QUANTITY ERROR" ausgegeben. - Ist die Stringvariable länger als ein Zeichen, so wird der ASCII-Code des ersten Zeichens ermittelt.

Dies ist eine Menge an Informationen, die erst einmal verdaut werden muß. Wie gesagt, arbeitet der Rechner intern nur mit den ASCII-Codes der Zeichen. Demnach versteht er auch, wenn Sie diesen ASCII-Code und nicht das Zeichen selbst eingeben. Doch warum haben wir zum Löschen des Bildschirms nicht das übliche Steuerzeichen verwendet? Wenn Sie schon einmal Programme aus Fachzeitschriften eingegeben haben, so sind diese meist mit Unmengen dieser Steuerzeichen bestückt. Wenn Sie diese Zeichen nun eingeben möchten, so müssen Sie erst einmal wissen, was das für ein Steuerzeichen ist. Ein Beispiel: Ein reverses 'S' taucht in einem Programm, das Sie eingeben möchten auf. Wissen Sie sofort, was das für ein Steuerzeichen ist? Bei den 16 Steuerzeichen für die Farbe werden Sie noch mehr verwirrt. Um dies zu umgehen, werden in den folgenden BASIC-Zeilen stets die ASCII-Codes der Steuerzeichen verwendet.

Wenn Sie wissen möchten, welchen ASCII-Code ein bestimmtes Zeichen hat, so ermitteln Sie ihn mit der Funktion ASC("z"). Sie geben dazu im Direktmodus nur den Befehl PRINT ASC("z") - für z setzen Sie das gewünschte Zeichen - ein. Der 128er antwortet dann prompt mit dem entsprechenden Code.

Die Schleifen in den Zeilen 140 und 160 dürften Ihnen klar sein. Es wird 40 mal das Zeichen '=' ausgegeben, was eine Zeile mit diesem Zeichen füllt.

Setzen der Hintergrund- und Rahmenfarbe

Sicherlich gefallen Ihnen die Standardfarben der 128ers überhaupt nicht. Warum sollen wir diese nicht ändern? Wie die Zeichenfarbe geändert wird, wissen Sie schon. Doch wie wird nun die Hintergrund- und die Rahmenfarbe geändert?
Zu diesem Zweck kennt Ihr 128er den BASIC-Befehl COLOR. Schauen wir uns zunächst die Beschreibung dieses Befehls an:

Problem:	Setzen von Bildschirm- und Textfarben
Befehl:	COLOR zi,fa
Parameter:	zi - Code für Farbziel (0-6) fa - Code für Farbe (1-16)
Beispiel:	COLOR 4,1 setzt den Farbcode 1 (schwarz) für das Farbziel 4 (Rahmen)
Bemerkung:	Liegen 'zi' oder 'fa' außerhalb des erlaubten Bereiches , so folgt die Fehlermeldung "ILLEGAL QUANTITY ERROR"

Mehr über den COLOR-Befehle erfahren Sie im Kapitel über Grafik.

Was ist aber jetzt mit den Codes für Farbziel und Farbe gemeint?
Wie wir bereits besprochen haben, arbeitet der Rechner intern nur mit Zahlen und nicht etwa mit Bezeichnungen wie "schwarz" oder "rot". Somit stehhen auch für Farbziel und Farbe bestimmte Codes. Welcher Codes das sind, entnehmen Sie bitte den folgenden Tabellen:

Code	Farbe	Code	Farbe
1	schwarz	9	orange
2	weiß	10	braun
3	rot	11	hellrot
4	türkis	12	grau 1
5	violett	13	grau 2
6	grün	14	hellgrün
7	blau	15	hellblau
8	gelb	16	grau 3

Code	Farbziel
0	Hintergrund (40 Zeichen)
4	Bildschirmrand
5	Schriftfarbe

Die Farbziel-Codes 1-3 und 6 werden wir im Kapitel über Grafik besprechen. Versuchen Sie nun einmal, den Bildschirmrahmen rot darzustellen. Der Befehl dazu ist

```
COLOR 4,3
```

Nun fehlt nur noch die Farbe für den Hintergrund. Dafür ist der Farbzielcode 0 zuständig. Es gelten die Farbcodes für den Rahmen. Nun wollen wir uns zwei Farben für das Adressenprogramm auswählen. Nehmen wir für den Rahmen und den Hintergrund die Farbe rot (Farbcode 3) und für die Zeichendarstellung die Farbe gelb (Farbcode 8). Die drei notwendigen Befehle für unsere Farbwahl bringen wir in Zeile 10 unter. Geben Sie diese Zeile nun ein.

```
10 COLOR 4,3:COLOR 0,3:COLOR 5,8
```

Wenn Sie das Programm wieder mit RUN starten, werden Sie die Auswirkung dieser Zeile kaum übersehen.

Unterprogramme

Im weiteren Verlauf der Adressenverwaltung werden wir die BASIC-Zeilen zur Ausgabe des Programmkopfes noch öfter benötigen. Damit wir nicht immer wieder diese Zeilen eingeben müssen, was das Programm auch unnötig verlängern würde, machen wir daraus ein Unterprogramm (auch Routine genannt). Doch wie wird ein Unterprogramm gekennzeichnet und wie führt man es aus? Zwei Fragen, die wie immer sofort geklärt werden.

Sie haben bereits im vorherigen Kapitel den Befehl GOTO kennengelernt, der es ermöglicht, das Programm in irgendeiner Zeile fortzusetzen. Da nach Ablauf eines Unterprogrammes jedoch wieder zurück verzweigt werden muß und die Routine nicht wissen kann, von wo Sie mit GOTO aufgerufen wurde, ist der Befehl GOTO hier fehl am Platze. Zum Aufruf von Routinen gibt es einen ähnlichen Befehl, der nun zunächst beschrieben wird.

Problem:	Aufruf von Unterprogrammen (Routinen)
Befehl:	GOSUB zn
Parameter:	zn - Zeilennummer, ab der die Routine beginnt
Beispiel:	GOSUB 100 ruft ein Unterprogramm bei Zeile 100 auf
Bemerkung:	- Die mit GOSUB aufgerufenen Unterprogramme müssen mit dem Befehl RETURN abgeschlossen werden. - Stößt das Programm während dem Ablauf auf ein RETURN, ohne daß ein GOSUB-Befehl ausgeführt wurde, so führt dies zur Fehlermeldung RETURN WITHOUT GOSUB ERROR

Dies sieht eigentlich recht einfach aus. Ist es auch, wenn man nur einige Male damit gearbeitet hat. Wir machen

nun die Zeilen 100-170 zu einem Unterprogramm, das von beliebiger Stelle im späteren Programm aufgerufen werden kann. Wie Sie der Befehlsbeschreibung entnommen haben, müssen wir dazu dieses Unterprogramm mit dem Befehl RETURN abschließen. Dazu hängen wir wieder eine Zeile an:

```
180 RETURN
```

Nun dürfen wir dieses Programm jedoch nicht mehr starten. Machen Sie es doch, so erscheint die Fehlermeldung "RETURN WITHOUT GOSUB ERROR". Dies ist auch klar, denn wir steigen mit RUN einfach in dieses Unterprogramm ein, das nur mit GOSUB aufgerufen werden darf. Wie machen wir nun weiter?

Beachten Sie zunächst, wie die Zeilennummern des zukünftigen Programms organisiert werden.

```
0-99      : Hier sollen alle Vorabinformationen für das
            Programm untergebracht werden. Z.B. Setzen
            der Farben, Einrichtung von Tabellen, Zuwei-
            sung von Variablen.

100-999   : In diesen Zeilen werden alle Routinen unter-
            gebracht.

1000-     : Ab der Zeile 1000 beginnt das eigentliche
            Programm, das Hauptprogramm.
```

Das Programm beginnt also in unserem Fall bei Zeile 10 und muß die Unterprogramme überspringen. Dazu geben wir die Zeile 99 ein:

```
99 GOTO 1000
```

In Zeile 1000 beginnt das Hauptprogramm. Hier setzen wir den Befehl GOSUB 100 ein:

```
1000 GOSUB 100
```

Nun ist das Programm wieder lauffähig. Es wird zuerst das Unterprogramm zum Anzeigen des Programmkopfes ausgeführt.

DAS MENÜ

Schauen wir zunächst, welche Fortschritte unser Programm gemacht hat:

```
10 COLOR 4,3:COLOR 0,3:COLOR 5,8
99 GOTO 1000
100 REM ====================
110 REM    PROGRAMMKOPF
120 REM ====================
130 PRINT CHR$(147);
140 FOR I=1TO40:PRINT"=";:NEXT I
150 PRINT"          ADRESSENVERWALTUNG"
160 FOR I=1TO40:PRINT"=";:NEXT I
170 PRINT:PRINT
180 RETURN
1000 GOSUB 100
```

Das Programm meldet sich nun nach dem Starten mit seinem Namen. Doch dies ist sicherlich nicht alles, sondern lediglich ein kleiner Anfang. Da wir mit diesem Programm mehrere Möglichkeiten haben möchten, die Adressen zu bearbeiten, müssen wir diese auch anbieten. Überlegen wir, was das Programm alles können soll. Zunächst soll es die Daten auf ein externes Speichermedium speichern und von dort wieder laden können. Wir benötigen dazu die beiden folgenden Auswahlmöglichkeiten:

```
-1- DATEI LADEN
-2- DATEI SPEICHERN
```

Weiterhin müssen wir natürlich Adressen eingeben. Dazu benötigen wir die Funktion 3:

```
-3- ADRESSEN EINGEBEN
```

Die eingegebenen Daten müssen auch zu ändern sein. Der folgende Punkt 4 soll dies ermöglichen:

```
-4- ADRESSEN AENDERN
```

Wenn wir vom Onkel Karl ab sofort nichts mehr wissen

möchten, sollten wir Ihn aus der Adressendatei nehmen. Wir benötigen also eine Funktion "löschen":

```
-5- ADRESSEN LOESCHEN
```

Nun kommt das wichtigste: Alle Daten nutzen uns wenig, wenn wir diese nicht ausgeben, also abrufen können. Mit folgendem Teilprogramm sollen Adressen ausgegeben werden:

```
-6- ADRESSEN AUSGEBEN
```

Kein Programm sollte mit dem Netzschalter des Rechners beendet werden. Wir schaffen deshalb mit dem letzen Punkt den Programmausgang:

```
-7- PROGRAMM BEENDEN
```

Nachdem wir uns über alle Programmfunktionen im Klaren sind, erweitern wir unser Programm so, daß diese Funktionen als Menü ausgegeben werden. Die folgenden Anweisungen werden hinzugefügt:

```
1005 COLOR 5,4:PRINT"    ++++++++++++++++++++++++"
1010 PRINT"    +";:COLOR 5,8:PRINT" PROGRAMMFUNKTIONEN:";:COLOR 5,4
:PRINT" +"
1015 PRINT"    ++++++++++++++++++++++++":COLOR 5,8
1030 PRINT
1040 COLOR 5,1:PRINT"     -1- ";:COLOR 5,8:PRINT"DATEI LADEN"
1050 COLOR 5,1:PRINT"     -2- ";:COLOR 5,8:PRINT"DATEI SPEICHERN"
1060 COLOR 5,1:PRINT"     -3- ";:COLOR 5,8:PRINT"ADRESSEN EINGEBEN"
1070 COLOR 5,1:PRINT"     -4- ";:COLOR 5,8:PRINT"ADRESSEN AENDERN"
1080 COLOR 5,1:PRINT"     -5- ";:COLOR 5,8:PRINT"ADRESSEN LOESCHEN"
1090 COLOR 5,1:PRINT"     -6- ";:COLOR 5,8:PRINT"ADRESSEN AUSGEBEN"
1100 COLOR 5,1:PRINT"     -7- ";:COLOR 5,8:PRINT"PROGRAMM BEENDEN"
```

Hier wurde das '+' Zeichen und Farbcodes verwendet, die zwar das Programm etwas unübersichtlich, jedoch optisch schöner machen. Nun muß der Anwender des Programms entscheiden, welche Programmfunktion er benötigt. Wir lesen dazu eine numerische Variable ein, die den Wert 1 bis 7 entsprechend der ausgewählten Funktion enthalten soll. Vorher wird noch ein PRINT mit dem Inhalt "AUSWAHL"

in schwarz ausgegeben.

```
1105 PRINT:PRINT
1110 COLOR 5,1:PRINT"     AUSWAHL ";:COLOR 5,8
1115 INPUT F
```

Die Abfrage mit IF

Was aber, wenn der Anwender eine ungültige Zahl eingibt, z.B. 9? Dies sollte vom Programm unterbunden werden. Die Frage ist nur wie? Wir müssen feststellen, welcher Wert eingegeben wurde, also die Variable F auf einen ungültigen Wert abfragen. Die Anweisung dazu wird nun beschrieben:

Problem:	Abrage auf Bedingungen
Befehl:	IF Bedingung THEN Anweisung
Parameter:	Bedingung - Vergleichoperation (z.B. A=10 oder B=12) Anweisung - Befehle, die ausgeführt werden, wenn die Bedingung wahr ist
Beispiel:	IF F=0 THEN GOTO 1000 Wenn die Variable F den Wert 0 hat, wird zur Zeile 1000 verzweigt, ansonsten wird in der Zeile danach das Programm fortgesetzt.
Bemerkung:	Folgt dem THEN ein GOTO, so kann dieses GOTO ignoriert werden. Das letzte Beispiel könnte auch so eingegeben werden: IF F=0 THEN 1000

Eine Bedingung kann nicht nur den Vergleichsoperator '=' beinhalten, sondern auch die Operatoren '>' (größer als) und '<' (kleiner als). Auch Kombinationen der drei sind erlaubt. Alles in Allem können folgende

Vergleichoperatoren eingesetzt werden:

SYMBOL	BEDEUTUNG
=	GLEICH
>	GRÖSSER ALS
<	KLEINER ALS
>=, =>	GRÖSSER ALS ODER GLEICH
<=, =<	KLEINER ALS ODER GLEICH
<>, ><	UNGLEICH

Der IF-Befehl prüft also zuerst, ob die Bedingung wahr ist. Ist dies der Fall, so werden die Anweisungen hinter dem THEN ausgeführt. Trifft die Bedingung nicht zu, so wird das Programm hinter der IF-Zeile fortgesetzt.

Doch dies ist nicht alles, was eine IF-Abfrage zu leisten vermag. Es können auch mehrere Bedingungen logisch miteinander verknüpft werden. Dies hört sich komplizierter an, als es eigentlich ist. Nehmen wir hier ein Beispiel zu Hilfe:

```
IF A=1 OR B>12 THEN 1000
```

Zwei Anweisungen, die miteinander verknüpft wurden. OR heist ODER und somit kann der Befehl wie folgt interpretiert werden: Wenn A gleich 1 oder B größer als 12, dann verzweige nach 1000. Zum Sprung nach Zeile 1000 muß also entweder A=1 oder B>12 oder beides zutreffen. Nur wenn beide Bedingungen verneint werden, wird das Programm hinter der IF-Zeile fortgesetzt. Die OR-Verknüpfung ist eine von zwei möglichen Verknüpfungen. Ein weiteres Beispiel:

```
IF A=1 AND B>12 THEN 1000
```

Dies ist eine UND-Verknüpfung, die nur dann erfüllt ist, wenn A gleich 1 und B größer als 12 ist.
Im Zusammenhang mit dem IF-Befehl sind noch die Befehle 'ELSE', 'BEGIN' und 'BEND' von Bedeutung. Auch diese

Befehle bleiben dem Fortgeschrittenen vorbehalten.

Kommen wir zu unserem Problem zurück. Wir wollten abfragen, ob der Anwender eine Zahl eingegeben hat, die nicht in dem Bereich von 1-7 liegt. Wie muß die entsprechende IF-Anweisung codiert werden? Schauen Sie sich die Lösung an:

```
IF F=0 OR F>7 THEN ...
```

Ja, was dann? Geben wir einfach eine Fehlermeldung in der letzten Bildschirmzeile aus und gehen zurück zur Zeile 1000. Zur Ausgabe der Fehlermeldung benötigen wir ein weiteres Unterprogramm, daß die Art des Fehlers einem String entnimmt und diesen reverse in Zeile 24 ausgibt. Sehen wir uns dieses Programm zunächst einmal an:

```
200 REM ==================
210 REM   FEHLERMELDUNG
220 REM ==================
230 PRINT CHR$(19);
240 FOR X=1 TO 24:PRINT CHR$(17);:NEXT X
250 PRINT CHR$(18);FE$;CHR$(146);
260 SLEEP 1
270 RETURN
```

Zunächst wird in Zeile 230 das Steuerzeichen zum Positionieren des Cursors in die obere linke Ecke des Bildschirms (HOME) ausgegeben. Dann wird der Cursor 24 mal nach unten gesetzt. Die Zeile 250 gibt den String FE$, der vom aufrufenden Programm gesetzt wurde, reverse aus. Danach wird eine Sekunde gewartet, bis das Unterprogramm abgeschlossen wird.

Wir können nun unsere IF-Abfrage hinter dem INPUT vervollständigen:

```
1120 IF F<1 OR F>7 THEN FE$="UNGUELTIGER WERT":GOSUB 200:
GOTO 1000
```

Versuchen Sie es selbst. Starten Sie das Programm und geben Sie einen falschen Wert ein. Das Programm stellt

die Falscheingabe fest und gibt die Fehlermeldung aus.

Wenn ein gültiger Wert eingegeben wurde, müssen wir nun die nötigen Schritte einleiten. Das heißt, es muß zum entsprechenden Programmteil verzweigt werden.

Berechnetes GOTO

Um abhängig von der Variablen F im Programm zu verzweigen, gibt es einen Befehl, der dies auf leichteste Art und Weise ermöglicht. Schauen Sie sich die Beschreibung dieses Befehls an:

Problem:	Berechneter Sprung
Befehl:	ON v1 GOTO n1,n2,n3,...
Parameter:	v1 - numerische Variable n1,n2,n3,... - Zeilennummer, zu denen abhängig vom Inhalt der Variablen F gesprungen werden soll
Beispiel:	ON F GOTO 100,200,300,400 Ist F=1, dann wird zur Zeile 100 verzweigt, ist F=2, dann wird zur Zeile 200 verzweigt, usw. Ist F größer als 4, so wird nicht verzweigt, sondern in der nächsten Zeile weitergemacht.
Bemerkung:	Ist F keine ganze Zahl (z.B. 3.45), so wird immer die Zahl vor dem Komma benutzt.

Ein idealer Befehl also für unseren Zweck. Wir bestimmen erst, bei welchen Zeilennummern die Programmteile begonnen werden sollen. Unsere nächste Zeile soll wie folgt eingegeben werden:

```
1130 ON F GOTO 5000,10000,15000,20000,25000,30000,35000
```

In welchen Zeilen nun die Programmteile untergebracht werden, ist im folgenden ersichtlich:

F	Zeile	Programmteil
1	5000	DATEI LADEN
2	10000	DATEI SPEICHERN
3	15000	ADRESSEN EINGEBEN
4	20000	ADRESSEN AENDERN
5	25000	ADRESSEN LOESCHEN
6	30000	ADRESSEN AUSGEBEN
7	35000	PROGRAMM BEENDEN

Wir werden nun ein Programmteil nach dem anderen fertigstellen. Sinngemäß fangen wir mit der Funktion 3, ADRESSEN EINGEBEN an. Doch zunächt müssen wir noch die Tabelle zur Aufnahme der Adressen dimensionieren. Fügen Sie dazu die Zeile 20 ein:

```
20 DIM D$(200,7)
```

Wir haben Platz für 200 Adressen reserviert. Das dürfte in der Regel vollkommen ausreichen. Wer hat schon mehr als 200 Bekannte? Für den kommerziellen Einsatz ist das Programm nur beschränkt einsatzfähig. Eine Firma, die Ihre 2000 Kunden verwalten möchte, sollte sich nach einem anderen geeigneten Programm umsehen.

Lassen Sie uns vor den einzelnen Programmteilen eine Routine schreiben, die für jedes Programm einen schönen Kopf ausgibt. Diese Routine soll vorher auch den Programmkopf ausgeben. Doch sehen Sie selbst:

```
300 REM ====================
310 REM KOEPFE PROGRAMMTEILE
320 REM ====================
330 PT$(1)="   DATEI LADEN   "
340 PT$(2)=" DATEI SPEICHERN "
350 PT$(3)="ADRESSEN EINGEBEN"
360 PT$(4)="ADRESSEN AENDERN "
370 PT$(5)="ADRESSEN LOESCHEN"
380 PT$(6)="ADRESSEN AUSGEBEN"
390 PT$(7)="PROGRAMM BEENDEN "
400 GOSUB 100
410 PRINT"        ";:COLOR 5,4
420 PRINT"++++++++++++++++++++"
430 PRINT"        +";:COLOR 5,8:PRINTPT$(F);:COLOR 5,4:PR
INT"+"
440 PRINT"        ++++++++++++++++++++":COLOR 5,8
450 RETURN
```

In den Zeilen 330-390 werden die Namen der Programmteile in einem Array gespeichert. Die Zeile 400 ruft das Unterprogramm"PROGRAMMKOPF"(Zeile100)auf.Danachwird der grün umrahmte Name der ausgewählten Programmfunktion ausgegeben. Dieses Unterprogramm wird als erstes von jedem Programmteil aufgerufen.

Adressen eingeben

Nun wird es ernst. Der erste anspruchsvolle Programmteil muß erstellt werden. Doch fangen wir erst einmal mit dem einfachsten an:

```
15000 REM ====================
15010 REM  ADRESSEN EINGEBEN
15020 REM ====================
15030 GOSUB 300
```

So weit, so gut. Als erstes wird also nun der Programmkopf sowie der Name des Programmteils ausgegeben. Danach müssen wir den Zähler für den zuletzt eingegebenen Datensatz um eins erhöhen. Diesen Zähler nennen wir Z.

```
15040 Z=Z+1
```

Wenn z.B. Z ungleich 0 ist, so wurden vorher bereits Daten geladen oder eingegeben, die dann hier erweitert werden. Wir müssen dann keine Meldung ausgeben, sondern erweitern die Adressen automatisch.

Nun müßten wir 7 INPUT-Zeilen eingeben, um die 7 Datenfelder einzulesen. Doch dies geht einfacher. Zunächt speichern wir ab Zeile 30 alle Feldnamen in dem Array F$(1) bis F$(7).

```
30 F$(1)="ANREDE     "
31 F$(2)="VORNAME    "
32 F$(3)="NAME       "
33 F$(4)="STRASSE    "
34 F$(5)="PLZ/ORT    "
35 F$(6)="TELEFON    "
36 F$(7)="BEMERKUNG  "
```

Diese Feldnamen werden zu Anfang des Programms im Array gespeichert und können im gesamten Programmablauf eingesetzt werden. Wozu wir sie jetzt benötigen, werden Sie im folgenden Verlauf des Programmteils feststellen.

```
15050 PRINT
15060 FOR I=1 TO 7
15070 PRINT F$(I);
15080 INPUT D$(Z,I)
15090 NEXT I
```

Starten Sie das Programm nachdem Sie diese Zeilen eingegeben haben und wählen die Funktion 3 aus. Was nun geschieht, bewirkt allein diese ausgetüftelte Schleife. Ein gesamter Adressensatz wird mit 4 Befehlen eingelesen. Innerhalb der Schleife wird zunächst der Name des Feldes ausgegeben. Die Variable I wird als Index für das Array der Feldnamen benutzt. Gleich hinter dem Feldnamen (Semikolon!) wird dann der Feldinhalt eingelesen. Hier wird zweimal indiziert, da die Adressen bekanntlich in einem zweidimensionalen Array gespeichert werden. Der erste Index ergibt sich aus der Variablen Z, die nach jeder Eingabe einer Adresse um 1 erhöht wird. Als zweiter Index wird wiederum die Variable I eingesetzt. Es werden also alle 7 Felder nacheinander über die Tastatur eingegeben.

Nachdem wir die Felder eingelesen haben, müssen wir dem Anwender die Möglichkeit geben, die Eingabe zu wiederholen. Er kann sich irgendwo verschrieben haben und muß dann korrigieren.

```
15100 PRINT
15110 PRINT"DATEN RICHTIG EINGEGEBEN (J/N)";
15120 X$="":INPUT X$
15130 IF X$="J" THEN 15160
15140 IF X$="N" THEN Z=Z-1:GOTO 15000
15150 PRINT CHR$(145);:GOTO 15110
```

Das Löschen der Variablen X$ ist notwendig, da sonst mit RETURN der letzte Inhalt übernommen wird. Wenn die Daten richtig eingegeben wurden, wird das Programm fortgesetzt. Sind die Daten nicht richtig eingegeben worden, wird der Programmteil nochmals gestartet. Vorher wird jedoch der Satzzähler um eins vermindert, da sonst in Zeile 15040 ein Datensatz weiter geschaltet wird. Wir wollen den letzen Satz jedoch nochmals eingeben. In Zeile 15150 wird

der Cursor einmal nach oben gesetzt, damit die erneute Abfrage nicht eine Zeile unter der letzten Abfrage nochmals erscheint.

Das Programm wird nun mit einer weiteren Frage fortgesetzt.

```
15160 PRINT"WEITERE EINGABEN (J/N)";
15170 X$="":INPUT X$
15180 IF X$="J" THEN 15000
15190 IF X$="N" THEN 1000
15200 PRINT CHR$(145);:GOTO 15160
```

Hier muß der Anwender entscheiden, ob er weitere Adressen eingeben möchte oder nicht. Wenn ja, dann wird der Programmteil erneut gestartet. Wenn nein, so wird zurück ins Menü verzweigt.

Damit haben wir den ersten Programmteil abgeschlossen. Man kann nun beliebig viele Daten eingeben, aber noch nicht viel damit anfangen. Zum Schluß folgt nochmals das gesamte Programmteil "ADRESSEN EINGEBEN!", das Sie mit Ihrem nun vergleichen können:

```
15000 REM ====================
15010 REM  ADRESSEN EINGEBEN
15020 REM ====================
15030 GOSUB 300
15040 Z=Z+1
15050 PRINT
15060 FOR I=1 TO 7
15070 PRINT F$(I);
15080 INPUT D$(Z,I)
15090 NEXT I
15100 PRINT
15110 PRINT"DATEN RICHTIG EINGEGEBEN (J/N)";
15120 X$="":INPUT X$
15130 IF X$="J" THEN 15160
15140 IF X$="N" THEN Z=Z-1:GOTO 15000
15150 PRINT CHR$(145);:GOTO 15110
15160 PRINT"WEITERE EINGABEN (J/N)";
15170 X$="":INPUT X$
```

```
15180 IF X$="J" THEN 15000
15190 IF X$="N" THEN 1000
15200 PRINT CHR$(145);:GOTO 15160
```

Adressen ändern

Nun möchten wir dem Anwender unseres Programms die Möglichkeit geben, einzelne Felder eines bestimmten Datensatzes zu ändern. Wir benutzen dazu die Funktionstasten. Mit der Taste F1 soll vorwärts, und mit der Taste F3 rückwärts "geblättert" werden. Soll eine Adresse geändert werden, so wird die Taste F5 gedrückt. Mit der Taste F7 wird zurück ins Menü verzweigt, die Änderung also abgeschlossen.

Das hört sich alles gut an, muß aber erst programmiert werden. Fangen wir zunächst mit den ersten Zeilen an:

```
20000 REM ====================
20010 REM   ADRESSEN AENDERN
20020 REM ====================
```

Diese Zeilen benötigen keinen Kommentar. Nun aber wird es kritisch. Wie gehen wir dieses Problem an? Wir wählen uns zunächst eine Variable aus, die den ersten Index des Arrays der Adressen darstellen soll. Dies ist die Varibale ZZ, die den Anfangswert 1 erhält.

```
20030 ZZ=1
```

Nun können wir den Programmkopf und den Namen des Arrays ausgeben:

```
20040 GOSUB 300
20050 PRINT
```

In den folgenden Zeilen können wir die erste Adresse (ZZ=1) ausgeben. Sie besteht bekanntlich aus 7 Feldern, deren Namen in einem Array gespeichert sind. Wir codieren wieder eine Schleife, die die Nummer des Feldes, den Namen und den Inhalt ausgibt.

```
20060 FOR I=1 TO 7
20070 PRINT I;F$(I);D$(ZZ,I)
20080 NEXT I
```

Nun müssen wir solange die Tastatur abfragen, bis irgendeine Funktionstaste gedrückt ist. Der Befehl dazu ist GET.

Problem:	Lesen eines Zeichens von Tastatur
Befehl:	GETKEY v1
Parameter:	v1 - Stringvariable, in der die gedrückte Taste gespeichert werden soll
Beispiel:	GETKEY X$ Testet, ob eine Taste gedrückt wurde und speichert diese Taste in X$. Danach wird Programmablauf sofort fortgesetzt.
Bemerkung:	Ist keine Taste gedrückt, so wird der Programmab lauf angehalten bis eine Taste betätigt wird.

Wollen Sie das Programm nur dann fortsetzen, wenn eine Taste gedrückt ist, so ist z.B. folgende Zeile notwendig:

```
10 GETKEY X$
```

Geben Sie diese Zeile nicht ein, es handelt sich nur um ein Beispiel. Es wird immer wieder die Zeile 10 ausgeführt, bis eine Taste gedrückt ist. So eine Anweisung folgt auch in unserem Programm.

```
20090 GETKEY X$
```

Die Funktionstasten des 128ers haben eine Besondere Eigenschaft. Mittels des BASIC-Befehls 'KEY,String' läßt sich jeder der Funktionstasten ein beliebiger String zuordnen. Bei Betätigen der Funktionstaste wird dann der

zugeordnete String ausgegeben. Geben Sie einmal den Befehl

```
KEY 1,"TESTSTRING"
```

ein. Betätigen Sie dann die f1-Taste. Sie sehen, daß das Wort 'TESTSTRING' so oft ausgegeben wird, wie Sie die f1-Taste betätigen.
Wenn Sie nur den Befehl 'KEY' ohne Zahl oder Komma eingeben, so werden die Belegungen aller Funktionstasten auf dem Bildschirm ausgegeben. Wenn Sie das einmal ausführen, so sehen Sie, daß alle Funktionstasten bereits mit einem String belegt sind. Key 1 (f1) ist mit 'TESTSTRING' belegt. Die anderen Funktionstasten sind mit BASIC-Befehlen belegt. Die f7-Taste ist beispielsweise mit dem 'LIST'-Befehl belegt. Betätigen Sie also einmal die f7-Taste. Sie sehen, der 'LIST'-Befehl wird mit einem einzigen Tastendruck ausgeführt.
Von den anderen Befehlen sind Ihnen auch einige bereits bekannt (RUN, DLOAD, DSAVE). Die übrigen lassen wir vorerst unberücksichtigt.
In unserem Adressprogramm sollen die Funktionstasten allerdings zur Steuerung des Programmes verwendet werden. Daher ordnen wir den Funktionstasten in folgender Zeile Codes zu, die sonst keine Bedeutung haben.

```
50 N=1:FORI=1TO7STEP2:KEYI,CHR$(132+N):KEYI+1,CHR$(136+N) :N=N+1:NEXT
```

Die Belegung der Funktionstasten ist dann wie folgt:

Taste	ASCII-Code
F1	133
F3	134
F5	135
F7	136
F2	137
F4	138
F6	139
F8	140

Die Funktionstasten F2, F4, F6 und F8 werden mit SHIFT bedient.

Da wir nur mit den Funktionstasten F1, F3, F5 und F7 arbeiten, müssen alle anderen Tasten vom Programm ignoriert werden.

```
20100 IF ASC(X$)<133 OR ASC(X$)>136 THEN 20090
```

Alle ASCII-Codes unterhalb von 133 und überhalb von 136 werden nicht akzeptiert. Das Programm verzweigt wieder zum GET-Befehl.

Wenn das Programm diese Zeile passiert hat, wurde eine gültige Taste gedrückt. Wir müssen nur noch feststellen, welche, und dementsprechend reagieren. Bei F7 wird abgeschlossen und zum Menü (Zeile 1000) verzweigt:

```
20110 IF ASC(X$)=136 THEN 1000
```

Wird die Taste F1 gedrückt, so soll der nächste Datensatz ausgegeben werden. Wir erhöhen dazu den Index ZZ um 1, aber nur wenn ZZ nicht schon den letzten Satz erreicht hat (ZZ=Z). Die folgende Zeile erfüllt diese Aufgabe:

```
20120 IF ASC(X$)=133 AND ZZ<Z THEN ZZ=ZZ+1:GOTO 20040
```

Bei der Taste F3 ist es ähnlich, nur muß ZZ um 1 vermindert werden, wenn ZZ nicht 1 ist.

```
20130 IF ASC(X$)=134 AND ZZ>1 THEN ZZ=ZZ-1:GOTO 20040
```

Bei Zeile 20040 wird der Datensatz mit dem Index ZZ ausgegeben. ZZ wurde von den Funktionstasten geändert.

Nun müssen wir noch abfragen, ob die Taste F5 gedrückt wurde. Mit dieser Taste wird die Änderung eingeleitet.

```
20140 IF ASC(X$)=135 THEN 20160
20150 GOTO 20090
```

Die Zeile 20150 wird erreicht, wenn die Tasten F1 oder F3

die Datei über- oder unterschreiten wollten. Dann werden die Tasten einfach ignoriert und zur Zeile 20090 verzweigt. In dieser Zeile wird erneut die Tastatur abgefragt.

Nun fehlt nur noch die eigentliche Änderung des Datensatzes ab Zeile 20160. Dort soll zuerst nach der Nummer und anschließend nach dem neuen Inhalt des zu ändernden Feldes gefragt werden.

```
20160 PRINT
20170 INPUT"FELDNUMMER (1-7) ";X
20180 IF X<1 OR X>7 THEN PRINT CHR$(145);:GOTO 20170
20190 PRINT
20200 INPUT"NEUER INHALT: ";D$(ZZ,X)
20210 GOTO 20040
```

Dies war schon der Rest unseres Programmteils. Zunächst wird die Nummer des zu ädernden Feldes in X eingelesen. Danach wird getestet, ob X einen ungültigen Wert enthält. Ist dies nicht der Fall, so wird der neue Inalt dieses Feldes eingelesen und der neu entstandene Adressensatz ab der Zeile 20040 ausgegeben. Dieser Programmteil kann nur über die Taste F7 verlassen werden. Nun folgt auch hier die Auflistung des kompletten Teils "ADRESSEN AENDERN":

```
20000 REM ====================
20010 REM   ADRESSEN AENDERN
20020 REM ====================
20030 ZZ=1
20040 GOSUB 300
20050 PRINT
20060 FOR I=1 TO 7
20070 PRINT I;F$(I);D$(ZZ,I)
20080 NEXT I
20090 GETKEY X$
20100 IF ASC(X$)<133 OR ASC(X$)>136 THEN 20090
20110 IF ASC(X$)=136 THEN 1000
20120 IF ASC(X$)=133 AND ZZ<Z THEN ZZ=ZZ+1:GOTO 20040
20130 IF ASC(X$)=134 AND ZZ>1 THEN ZZ=ZZ-1:GOTO 20040
20140 IF ASC(X$)=135 THEN 20160
20150 GOTO 20090
```

```
20160 PRINT
20170 INPUT"FELDNUMMER (1-7) ";X
20180 IF X<1 OR X>7 THEN PRINT CHR$(145);:GOTO 20170
20190 PRINT
20200 INPUT"NEUER INHALT: ";D$(ZZ,X)
20210 GOTO 20040
```

Adressen löschen

Beim Löschen der Adressen wenden wir eine andere Technik als beim Ändern an. Sie sollen ja möglichst vielseitg programmieren lernen. Wir müssen den Vor- und Zunamen der zu löschenden Adresse angeben. Ist er nicht bekannt, so kann man ihn mit dem "blättern" in "ADRESSE AENDERN" ermitteln.

Doch bevor dieser Teil des Programms beginnt, soll geprüft werden, ob überhaupt Daten im Rechner gespeichert sind. Wir bauen dazu wieder eine Routine auf, die ab Zeile 500 liegen soll. Schauen wir uns zunächst die komplette Routine an:

```
500 REM =================
510 REM DATEN IM RECHNER?
520 REM =================
530 IF Z>0 THEN 560
540 FE$="KEINE DATEN IM RECHNER!"
550 GOSUB 200
560 RETURN
```

Diese Routine ist zwar kurz, aber trotzdem sinnvoll. Wir benötigen Sie noch in den anderen Programmteilen. In Zeile 530 wird geprüft, ob Daten im Rechner sind. Die Varibale Z enthält immer den letzten Datensatz und somit die Anzahl der im Rechner gespeicherten Adressen. Ist diese Variable 0, so sind keine Daten eingegeben und auch keine geladen worden. Wir geben dann die Fehlermeldung "KEINE DATEN IM RECHNER!" mit der Routine ab Zeile 200 aus.

Fangen wir nun mit unserem Programmteil an. Die ersten Zeilen sind erfahrungsgemäß immer die leichtesten.

```
25000 REM ==================
25010 REM  ADRESSEN LOESCHEN
25020 REM ==================
25030 GOSUB 300:REM KOPF
25040 PRINT
25050 GOSUB 500:REM DATEN IM RECHNER
```

```
25055 IF Z=0 THEN 1000
25060 INPUT "VORNAME: ";D1$
25070 INPUT "NAME   : ";D2$
```

Wir erzeugen wieder den Programmkopf in Zeile 25030. Danach rufen wir die Routine ab Zeile 500 auf, die bekanntlich überprüft, ob überhaupt Daten im Rechner vorhanden sind. Nun lesen wir Vorname und Name der zu löschenden Adresse ein. Diese beiden Eingaben speichern wir in D1$ und D2$. Nun müssen wir die gesamte Adressentabelle nach diesem Namen durchsuchen. Eine FOR...NEXT-Schleife kann hier nicht eingesetzt werden, da diese mit Sicherheit vorzeitig verlassen wird, nämlich dann, wenn die Adresse gefunden wurde. FOR...NEXT-Schleifen sollte man nicht vorzeitig abbrechen. Der Grund dafür ist folgender: Die Adresse des Schleifenanfangs wird im Rechner abgelegt. Erst nach ordnungsgemäßen Ablauf der Schleife wird diese Adresse wieder gelöscht. Wird die Schleife nun mehrmals vorzeitig verlassen, so füllt sich dieser Speicherbereich auf. Dies kann dazu führen, daß keine Schleife mehr akzeptiert wird.
Wir bauen also eine selbst gesteuerte Schleife auf.

```
25080 X=1
25090 IF D$(X,2)=D1$ AND D$(X,3)=D2$ THEN 25130
25100 IF X<Z THEN X=X+1:GOTO 25090
25110 FE$="NAMEN NICHT GEFUNDEN!:"
25120 GOSUB 200:GOTO 1000
```

Der Anfangswert der Schleifenvariable X wird zunächst auf 1 gesetzt. Innerhalb der Schleife wird gprüft, ob die mit X indizierte Adresse mit der gesuchten übereinstimmt. Der zweite Index 2 und 3 entspricht dem Vornamen und dem Namen, der mit D1$ und D2$ verglichen wird. Stimmt der Name überein, so wird das Program in Zeile 25130 fortgesetzt. In der Zeile 25100 wird die Schleifenvariable X um 1 erhöht, sofern sie noch nicht das Ende der Adresstabelle erreicht hat.

Wird die Schleife komplett durchlaufen, also nicht durch auffinden des Namens abgebrochen, so ist der Name nicht gefunden worden. Hier geben wir eine Meldung aus und verzweigen zurück ins Menü.

Wir müssen das Programm nun an der Stelle fortsetzen, zu der beim Auffinden des gesuchten Namens verzweigt wird. Dies ist die Zeile 25130. Hier soll die komplete zu löschende Adresse nochmals angezeigt werden, wonach der Anwender dann entscheidet, ob sie wirklich gelöscht werden soll.

```
25130 GOSUB 300:PRINT
25135 FOR I=1 TO 7
25140 PRINT F$(I);D$(X,I)
25150 NEXT I
25155 PRINT
25160 INPUT"ADRESSE LOESCHEN (J/N) ";X$
25170 IF X$="J" THEN 25200
25180 IF X$="N" THEN 1000
25190 PRINT CHR$(145);:GOTO 25160
```

Hier wird zunächst ein neuer Bildschirm aufgebaut (Zeile 25130). Danach wir mit der Schleife in Zeile 25135 bis 25150 der Adressensatz ausgegeben. Nun fragen wir nach, ob diese Adresse wirklich gelöscht werden soll. Soll diese Adresse nicht gelöscht werden (unentschlossene Leute gibt es überall), so wird zurück ins Menü verzweigt. Soll die Adresse doch gelöscht werden, so geschieht dies ab Zeile 25200. Nehmen wir auch nun zunächst Einblick in die letzten Zeilen dieses Programmteils.

```
25200 FOR Y=X TO Z-1
25210 FOR I=1 TO 7
25220 D$(Y,I)=D$(Y+1,I)
25230 NEXT I
25240 NEXT Y
25250 Z=Z-1
25260 GOTO 1000
```

Zum Löschen einer Adresse reicht es nicht, den

entsprechenden Tabellenplatz zu löschen. Würden wir dies tun, so bleibt die Größe der Tabelle trotz Löschen einer Adresse erhalten. Dies ist nicht im Sinne des Erfinders. Alle Adressen, die sich hinter der zu löschenden Adresse befinden, müssen eins nach vorne aufrücken. Wenn z.B. die Adresse mit dem Index 5 gelöscht wird und 7 Adressen im Rechner gespeichert sind, so rückt die 6. Adresse auf die 5., die somit gelöscht wird. Die letzte, also 7. Adresse rückt auf die 6. Wenn wir nun die Anzahl der Adressen auf 6, also um eins vermindern, so ist die Adresse mit dem Index 5 "echt" gelöschtworden.

Bei der Programmierung dieses Prinzips taucht zum ersten Mal eine verschachtelte Schleife auf. Innerhalb der Schleife mit der Variablen Y befindet sich die Schleife I. Die Schleife Y durchläuft alle Datensätze ab dem zu löschenden Datensatz. Die Felder werden alle einzeln nachgerückt. Sind alle Felder aufgerückt, so kan der nächste Datensatz bearbeitet werden. Die äußere Schleife läuft nur bis Z-1, da der letzte Datensatz in Zeile 25220 mit Y+1 indiziert wird. Wenn Sie die entsprechenden Zeilen studieren, so erkennen Sie die Technik dieses "Aufrückens".

Nun haben wir auch diesen Teil unseres zur Zeit schon recht umfangreichen Programms fertiggestellt. Es folgt nun wie immer die komplette Auflistung dieses Teils:

```
25000 REM ====================
25010 REM  ADRESSEN LOESCHEN
25020 REM ====================
25030 GOSUB 300:REM KOPF
25040 PRINT
25050 GOSUB 500:REM DATEN IM RECHNER
25055 IF Z=0 THEN 1000
25060 INPUT "VORNAME: ";D1$
25070 INPUT "NAME   : ";D2$
25080 X=1
25090 IF D$(X,2)=D1$ AND D$(X,3)=D2$ THEN 25130
25100 IF X<Z THEN X=X+1:GOTO 25090
25110 FE$="NAMEN NICHT GEFUNDEN!:"
25120 GOSUB 200:GOTO 1000
```

```
25130 GOSUB 300:PRINT
25135 FOR I=1 TO 7
25140 PRINT F$(I);D$(X,I)
25150 NEXT I
25155 PRINT
25160 INPUT"ADRESSE LOESCHEN (J/N) ";X$
25170 IF X$="J" THEN 25200
25180 IF X$="N" THEN 1000
25190 PRINT CHR$(145);:GOTO 25160
25200 FOR Y=X TO Z-1
25210 FOR I=1 TO 7
25220 D$(Y,I)=D$(Y+1,I)
25230 NEXT I
25240 NEXT Y
25250 Z=Z-1
25260 GOTO 1000
```

Adressen ausgeben

Dieser Programmabschnitt gehört sicherlich nicht zu den leichtesten. Hier wollen wir zum ersten Mal auf dem Drucker ausgeben. Doch auch Bildschirmausgabe soll ermöglicht werden. Nach Auswahl, ob denn nun auf dem Bildschirm oder dem Drucker ausgegeben werden soll, geben Sie die Suchbegriffe ein. Doch schreiben wir zunächst die ersten Zeilen.

```
30000 REM ====================
30010 REM  ADRESSEN AUSGEBEN
30020 REM ====================
30030 GOSUB 300:PRINT
30040 GOSUB 500
30050 IF Z=0 THEN 1000
30060 INPUT"DRUCKER ODER BILDSCHIRM (D/B) ";G$
30070 IF G$<>"B" AND G$<>"D" THEN PRINT CHR$(145);:GOTO 3
0060
30075 IF G$="D" THEN OPEN 1,4
```

Hier wird wieder geprüft, ob Daten im Rechner sind (Zeilen 30040-30050). Danach wird gefragt, ob auf Drucker oder Bildschirm ausgegeben werden soll. Die Antwort ('D' oder 'B') wird in der Stringvariablen G$ festgehalten, da

dies später noch mehrmals benötigt wird. Falls der Drucker ausgewählt wurde, wird dieser nun mit dem Befehl OPEN geöffnet. Ein neuer Befehl, der sofort beschrieben wird.

Problem:	Öffnen eines Kanals
Befehl:	OPEN nr,ga,sa
Parameter:	nr - Nummer des Kanals (1-255) ga - Adresse des Geräts, zu dem der Kanal geöffnet werden soll. Die Drucker haben ab Werk die Adresse 4 sa - Sekundäradresse. Sie kann dem Gerät zusätzliche Informationen geben. Einige Drucker arbeiten zusätzlich mit anderen Sekundäradressen als 0, die bei Weglassen von 'sa' angenommen wird.
Beispiel:	OPEN 1,4 Öffnet den Drucker mit der Kanalnummer 1, der Geräteadresse 0 und der Sekundäradresse 0.
Bemerkung:	Wird ein Kanal, der bereits geöffnet ist, nochmals geöffnet, so wird die Fehlermeldung "FILE OPEN ERROR" ausgegeben.

Im weiteren Verlauf des Programms sollen nun die Suchbegriffe eingelesen werden.

```
30080 GOSUB 300:PRINT
30090 PRINT"SUCHBEGRIFFE:"
30100 PRINT"------------"
30110 PRINT
30120 FOR I=1 TO 7:S$(I)="":NEXT I
30130 FOR I=1 TO 7
30140 PRINT F$(I);
30150 INPUT S$(I)
30160 NEXT I
```

Zeile 300 erstellt einen neuen Bildschirm, bevor die Zeilen 30090-30100 siganlisieren, daß die Suchbegriffe eingegeben werden müssen. Die Eingabe erfolgt wieder in einer Schleife. Doch zuvor müssen die Suchbegriffe, die im Array S$ gespeichert werden, gelöscht werden. Sie können noch Daten aus vorherigem Suchen enthalten. Die Begriffe werden dann in den Zeilen 30130 bis 30160 eingelesen. Wenn z.B. alle Adressen aus Düsseldorf gesucht werden sollen, so geben Sie in den ersten 4 Feldern nur RETURN ein. Das 5. Feld, daß die Postleitzahl und den Ort enthält, wird mit "4000 DUESSELDORF" gefüllt und anschließend mit RETURN eingegeben. Die restlichen 2 Felder werden wiederum mit RETURN ignoriert. Wollen Sie aber alle Herren aus Düsseldorf suchen, so geben Sie zusätzlich bei der Anrede "HERR" ein. Sie können also nach mehreren Kriterien gleichzeitig suchen. Nach der Eingabe der Suchbegriffe soll die Suche beginnen.

```
30170 FOR Y=1 TO Z
30180 S=0
30190 FOR I=1 TO 7
30200 IF S$(I)="" THEN S=S+1:GOTO 30220
30210 IF D$(Y,I)=S$(I) THEN S=S+1
30220 NEXT I
```

Hier bemerken Sie wieder eine verschachtelte Schleife. Die äußere Schleife (Y) durchläuft alle Datensätze. Bevor die innere Schleife, die die einzelnen Felder mit den Suchbegriffen vergleicht, beginnt, wird ein Zähler auf Null gesetzt. Wozu, das werden Sie später feststellen. In der Schleife I wird zunächst abgefragt, ob zu dem

aktuellen Feld ein Suchbegriff eingegeben wurde. Wenn nein, wird der Zähler S, der bei einem den Suchbegriffen entsprechenden Datensatz den Wert 7 enthält, um eins erhöht und zum NEXT verzweigt. Ist ein Suchbegriff eingegeben worden, so wird dieser mit dem tatsächlichen Feld verglichen. Nur wenn beide übereinstimmen, wird der Erfolgszähler S um eins erhöht. Stimmen die beiden nicht überein, so wird S nicht erhöht. S erhält somit nicht mehr den Wert 7 und der Datensatz entspricht nicht den Bedingungen.

```
30230 IF S<>7 THEN 30300
30240 IF G$="D" THEN PRINT#1
30245 IF G$="B" THEN GOSUB 300:PRINT
```

Hier wird die Ausgabe des gefundenen Datensatzes vorbereitet. Nur wenn der Datensatz nicht den Suchbegriffen entspricht (S ungleich 7) wird die gesamte Ausgabe übersprungen. Wenn der Drucker ausgewählt wurde, wird ein Zeilenvorschub, also ein PRINT#1 ausgegeben. Dieser Befehl unterscheidet sich vom normalen PRINT, da er nur auf einen zuvor geöffneten Kanal ausgegeben wird. Die Nummer dieses Kanals muß mit der Nummer hinter diesem PRINT übereinstimmen.
Wurde die Bildschirmausgabe gewählt, so wird dieser Bildschirm in Zeile 30245 aufbereitet.
Es folgen die restlichen Zeilen dieses Programmabschnittes, die anschließend wieder dokumentiert werden.

```
30250 FOR I=1 TO 7
30260 IF G$="B" THEN PRINT F$(I);D$(Y,I)
30270 IF G$="D" THEN PRINT#1,F$(I);D$(Y,I)
30280 NEXT I
30285 IF G$="D" THEN 30300
30290 PRINT:INPUT"DRUECKEN SIE RETURN";X$
30300 NEXT Y
30310 GOSUB 300:PRINT:PRINT
30320 PRINT"DATEIENDE"
30330 INPUT"DRUECKEN SIE RETURN";X$
30340 IF G$="D" THEN CLOSE 1
30350 GOTO 1000
```

In diesen Zeilen dreht sich fast alles um die Ausgabe des gefundenen Datensatzes. Die Zeilen 30250 bis 30280 bilden eine Schleife, in der die 7 Felder der Adresse mit ihrer Bezeichnung ausgegeben werden. Abhängig vom Inhalt der Variablen G$ wird die Adresse auf dem Drucker oder auf dem Bildschirm dargestellt.

Wenn Bildschirmausgabe ausgewählt wurde, wird erst nach Drücken von RETURN mit der Suche fortgefahren. Weitersuchen heißt hier, die Schleife Y mit dem Befehl NEXT um eins zu erhöhen. Ist die Druckerausgabe aktuell, so wird die Zeile zum Betätigen von RETURN übersprungen. Es werden also alle Adressen hintereinander ausgedruckt.

Hinter dem Ende der Schleife Y ist die Suche beendet. Das wird durch die Meldung "DATEIENDE" signalisiert. Wenn der Anwender nun RETURN drückt, wird der Programmabschnitt beendet, also zurück zum Menü verzweigt. Vorher wird der Druckerkanal geschlossen, falls auf den Drucker ausgegeben wurde.

Dies war ein weiterer Meilenstein zu unserer Adressenverwaltung. Nun müssen nur noch die Teile zum Laden und Speichern der Datei auf Diskette geschrieben werden.

Mit dem folgenden Listing können Sie nochmals Ihr Programmteil "ADRESSEN AUSGEBEN" kontrollieren:

```
30000 REM ====================
30010 REM  ADRESSEN AUSGEBEN
30020 REM ====================
30030 GOSUB 300:PRINT
30040 GOSUB 500
30050 IF Z=0 THEN 1000
30060 INPUT"DRUCKER ODER BILDSCHIRM (D/B) ";G$
30070 IF G$<>"B" AND G$<>"D" THEN PRINT CHR$(145);:GOTO 3
0060
30075 IF G$="D" THEN OPEN 1,4
30080 GOSUB 300:PRINT
30090 PRINT"SUCHBEGRIFFE:"
30100 PRINT"------------"
```

```
30110 PRINT
30120 FOR I=1 TO 7:S$(I)="":NEXT I
30130 FOR I=1 TO 7
30140 PRINT F$(I);
30150 INPUT S$(I)
30160 NEXT I
30170 FOR Y=1 TO Z
30180 S=0
30190 FOR I=1 TO 7
30200 IF S$(I)="" THEN S=S+1:GOTO 30220
30210 IF D$(Y,I)=S$(I) THEN S=S+1
30220 NEXT I
30230 IF S<>7 THEN 30300
30240 IF G$="D" THEN PRINT#1
30245 IF G$="B" THEN GOSUB 300:PRINT
30250 FOR I=1 TO 7
30260 IF G$="B" THEN PRINT F$(I);D$(Y,I)
30270 IF G$="D" THEN PRINT#1,F$(I);D$(Y,I)
30280 NEXT I
30285 IF G$="D" THEN 30300
30290 PRINT:INPUT"DRUECKEN SIE RETURN";X$
30300 NEXT Y
30310 GOSUB 300:PRINT:PRINT
30320 PRINT"DATEIENDE"
30330 INPUT"DRUECKEN SIE RETURN";X$
30340 IF G$="D" THEN CLOSE 1
30350 GOTO 1000
```

Datei sichern

Wir nähern uns dem Abschluß der Adressenverwaltung. Die eingegebenen Daten gehen mit dem Abschalten des Rechners natürlich verloren. Doch wozu gibt es eine Floppydisk, auf der wir die Daten wie Musik aufnehmen können. Was muß nun zuerst beachtet werden? Beginnen wir diesen Teil wie üblich mit dem Bildschirmkopf und der Abfrage, ob Daten im Rechner gespeichert sind.

```
10000 REM ===================
10010 REM   DATEI SPEICHERN
10020 REM ===================
10030 GOSUB 300:PRINT
10040 GOSUB 500:IF Z=0 THEN 1000
```

Diese Zeilen sind uns nicht mehr unbekannt. Bevor wir die Daten auf die Diskette speichern können, muß diese erst eingelegt werden. Die folgenden Zeilen fordern dazu auf.

```
10050 PRINT"BITTE DATENDISKETTE EINLEGEN"
10060 PRINT
10070 INPUT"DRUECKEN SIE DANACH RETURN";X$
```

Auch diese Zeilen dürften Ihnen keine Verständnisschwierigkeiten bereiten. Ein paar PRINTs mit einem anschließendem INPUT dürften mittlerweile zu Ihren leichtesten Übungen gehören.

Doch nun geht es los. Da die Adressen sicher mehr als einmal auf Diskette gespeichert werden sollen, muß die alte Adressdatei auf der Datendiskette erst einmal gelöscht werden bevor die neue Adressdatei gespeichert werden kann. Das läßt sich einfach mit dem SCRATCH "Dateiname"-Befehl erreichen.
Die Daten müssen abgespeichert werden, nachdem der Kanal zum Schreiben auf Diskette geöffnet wurde. Wir nehmen den Kanal 1.

```
10075 SCRATCH "ADRESSEN"
10080 DOPEN #1,"ADRESSEN",W
10090 PRINT#1,Z
10100 FOR Y=1 TO Z
10110 FOR I=1 TO 7
10120 PRINT#1,D$(Y,I)
10130 NEXT I
10140 NEXT Y
10150 CLOSE 1
```

Wir öffnen also die Diskettendatei mit dem Namen "ADRESSEN". Wie Sie sehen geht das Öffnen einer Datei auf

Diskette mit dem DOPEN #x,"Dateiname",W(rite) noch einfacher als bei dem Drucker.
Das erste, was wir auf die Diskette schreiben, ist die Anzahl der zu übermittelnden Datensätze, die bekanntlich in Z gespeichert ist. Danach senden wir die einzelnen Felder der Adressentabelle mit Hilfe einer verschachtelten Schleife, die Ihnen nicht mehr unbekannt ist. Zuerst Feld 1 bis 7 des ersten Datensatzes, dann Feld 1 bis 7 des zweiten, usw. Nachdem wir alles übermittelt haben, schließen wir den Kanal 1 wieder.

Nach Speicherug der Daten soll dies noch einmal gemeldet werden. Die folgenden Zeilen sind dafür zuständig.

```
10160 PRINT:PRINT"DATEN SIND GESICHERT!"
10170 SLEEP 2
10180 GOTO 1000
```

Warum zwei PRINT-Befehle für eine auszugebende Meldung? Eine Frage, die vielleicht aufkommen kann. Der erste PRINT gibt eine Leerzeile aus, die die anschließende Meldung optisch vom vorher ausgegebenen Text trennen soll. Anschließend wird eine Warteschleife von ca. 2 Sekunden aufgebaut und ins Menü zurück verzweigt.

So einfach ist das Speichern von Daten auf der Floppydisk. Doch nun wollen wir diese Daten wieder zurückholen.

Datei laden

Das Laden der Adressen verläuft fast auf die gleiche Weise wie das Speichern. Hier wird nicht geprüft, ob Daten im Rechner vorhanden sind. Der Anwender sollte also neu eingegebenene Adressen erst abspeichern, bevor er eine Adressendatei lädt.

```
5000 REM ===============
5010 REM   DATEI LADEN
5020 REM ===============
5030 GOSUB 300:PRINT:PRINT
```

```
5040 PRINT"BITTE DATENDISKETTE INS LAUFWERK LEGEN!"
5050 PRINT
5060 INPUT"DRUECKEN SIE DANACH RETURN";X$
```

Hier wird wieder der Hinweis ausgegeben, die Datendiskette einzulegen.

```
5070 DOPEN #1,"ADRESSEN"
5080 INPUT#1,Z
5090 FOR Y=1TO Z
5100 FOR I=1 TO 7
5110 INPUT#1,D$(Y,I)
5120 NEXT I
5130 NEXT Y
5140 CLOSE 1
```

Hier wird die Datei "ADRESSEN" auf der Diskette zum Lesen geöffnet. Der Lesezugriff wird dadurch gekennzeichnet, daß kein W für W(rite) angegeben wurde. Danach wird die Anzahl der in dieser Datei gespeicherten Daten eingelesen. Diese Variable wird als Endwert für die Schleife Y benutzt. Es werden also soviel Datensätze eingelesen, wie zuvor gespeichert wurden. Der Kanal wird in Zeile 5160 wieder geschlossen.

```
5150 PRINT:PRINT"DATEN SIND GELADEN!"
5160 SLEEP 2
5170 GOTO 1000
```

Die letzten Zeilen dieses Abschnitts geben wieder eine Meldung aus und verzweigen nach der Warteschleife zurück ins Menü

Programm beenden

Wie zu Anfang des Kapitels bereits erwähnt, sollte ein Programm nicht mit Hilfe des Netzschalters ausgeschaltet werden. Auch die RUN/STOP-Taste ist kein sauberer Ausgang. Wie ein Programm beendet werden sollte, wird nun demonstriert. Blicken wir zunächst wieder auf die ersten Zeilen.

```
35000 REM ====================
35010 REM   PROGRAMM BEENDEN
35020 REM ====================
35030 GOSUB 300:PRINT
35040 IF Z=0 THEN 35150
```

Die ersten Zeilen werden Ihnen sicher schon langweilig. Doch die Zeile 35040 sieht ungewohnt aus. Wenn Z=0 ist, also wenn keine Daten im Rechner gespeichert sind, wird das Programm ohne weiteres sofort beendet. Dies geschieht in Zeile 35150

```
35050 PRINT"SIND ALLE DATEN GESICHERT (J/N) ";
35060 INPUT X$
35070 IF X$="N" THEN 1000
35080 IF X$="J" THEN 35100
35090 PRINT CHR$(145);:GOTO 35050
```

Langsam erkennt man den Sinn eines Programmteils "PROGRAMM BEENDEN". Es wird also festgestellt, ob alle Daten gesichert wurden. Immerhin kann man diesen Programmabschnitt versehentlich aufgerufen haben. Wenn Sie nun mit nein antworten, kommen Sie wieder in das Menü zurück. Wenn Sie die Adressen jedoch abgespeichert haben, kann Sie nichts daran hindern, das Programm zu beenden.

```
35100 GOSUB 300
35110 PRINT"DAS PROGRAMM KANN MIT 'GOTO 1000' WIEDER";
35120 PRINT"GESTARTET WERDEN, OHNE DASS DATEN VER-"
35130 PRINT"LOREN GEHEN!"
35140 PRINT
35150 END
```

Nun folgt vor Beendigung des Programms ein wichtiger Hinweis: Nach dem Verlassen des Programms kann man es mit dem Befehl 'GOTO 1000' wieder ohne Datenverlust starten. Der Befehl RUN löscht vor dem Start sämtliche Variablen, ist also nicht zu empfehlen.

Nun haben wir es geschafft. Sie besitzen nun eine bis aufs Letzte dokumentierte Adressenverwaltung, bei deren Erstellung Sie sicher viel gelernt haben. Das, was Sie

nun gelernt haben, können Sie entweder zur Verwirklichung eigener Ideen oder zur Durchführung individueller Programmändeungen, speziell bei dieser Adressenverwaltung verwenden. Falls Sie einige Abschnitte nicht sofort verstanden haben, arbeiten Sie diese ruhig nochmals durch. Möchten Sie BASIC bis ins letzte Detail lernen, so gibt es auf dem Markt ein reichhaltiges Angebot an BASIC-Lernbüchern. Wenn Sie tiefer in die fantastischen Möglichkeiten des 128er (Grafik, Sound usw.) einsteigen möchten, so empfehle ich Ihnen das reichhaltige Angebot der DATA BECKER BÜCHER.

```
10 COLOR 4,3:COLOR 0,3:COLOR 5,8
20 DIM D$(200,7)
30 F$(1)="ANREDE      "
31 F$(2)="VORNAME     "
32 F$(3)="NAME        "
33 F$(4)="STRASSE     "
34 F$(5)="PLZ/ORT     "
35 F$(6)="TELEFON     "
36 F$(7)="BEMERKUNG   "
50 N=1:FORI=1TO7STEP2:KEYI,CHR$(132+N):KEYI+1,CHR$(136+N):N=N+1:NEXT
99 GOTO 1000
100 REM ====================
110 REM     PROGRAMMKOPF
120 REM ====================
130 PRINT CHR$(147);
140 FOR I=1TO40:PRINT"=";:NEXT I
150 PRINT"           ADRESSENVERWALTUNG"
160 FOR I=1TO40:PRINT"=";:NEXT I
170 PRINT:PRINT
180 RETURN
200 REM =================
210 REM   FEHLERMELDUNG
220 REM =================
230 PRINT CHR$(19);
240 FOR X=1 TO 24:PRINT CHR$(17);:NEXT X
250 PRINT CHR$(18);FE$;CHR$(146);
260 SLEEP 1
270 RETURN
300 REM ====================
310 REM KOEPFE PROGRAMMTEILE
320 REM ====================
330 PT$(1)="   DATEI LADEN   "
340 PT$(2)=" DATEI SPEICHERN "
350 PT$(3)="ADRESSEN EINGEBEN"
360 PT$(4)="ADRESSEN AENDERN "
370 PT$(5)="ADRESSEN LOESCHEN"
380 PT$(6)="ADRESSEN AUSGEBEN"
390 PT$(7)="PROGRAMM BEENDEN "
400 GOSUB 100
410 PRINT"         ";:COLOR 5,4
420 PRINT"++++++++++++++++++++"
```

```
430 PRINT"        +";:COLOR 5,8:PRINTPT$(F);:COLOR 5,4:PRINT"+"
440 PRINT"        ++++++++++++++++++++":COLOR 5,8
450 RETURN
500 REM =================
510 REM DATEN IM RECHNER?
520 REM =================
530 IF Z>0 THEN 560
540 FE$="KEINE DATEN IM RECHNER!"
550 GOSUB 200
560 RETURN
1000 GOSUB 100
1005 COLOR 5,4:PRINT"    ++++++++++++++++++++++++"
1010 PRINT"    +";:COLOR 5,8:PRINT" PROGRAMMFUNKTIONEN:";:COLOR 5,4:PR
INT" +"
1015 PRINT"    ++++++++++++++++++++++++":COLOR 5,8
1030 PRINT
1040 COLOR 5,1:PRINT"     -1- ";:COLOR 5,8:PRINT"DATEI LADEN"
1050 COLOR 5,1:PRINT"     -2- ";:COLOR 5,8:PRINT"DATEI SPEICHERN"
1060 COLOR 5,1:PRINT"     -3- ";:COLOR 5,8:PRINT"ADRESSEN EINGEBEN"
1070 COLOR 5,1:PRINT"     -4- ";:COLOR 5,8:PRINT"ADRESSEN AENDERN"
1080 COLOR 5,1:PRINT"     -5- ";:COLOR 5,8:PRINT"ADRESSEN LOESCHEN"
1090 COLOR 5,1:PRINT"     -6- ";:COLOR 5,8:PRINT"ADRESSEN AUSGEBEN"
1100 COLOR 5,1:PRINT"     -7- ";:COLOR 5,8:PRINT"PROGRAMM BEENDEN"
1105 PRINT:PRINT
1110 COLOR 5,1:PRINT"     AUSWAHL ";:COLOR 5,8
1115 INPUT F
1120 IF F<1 OR F>7 THEN FE$="UNGUELTIGER WERT":GOSUB 200:GOTO 1000
1130 ON F GOTO 5000,10000,15000,20000,25000,30000,35000
5000 REM ===============
5010 REM   DATEI LADEN
5020 REM ===============
5030 GOSUB 300:PRINT:PRINT
5040 PRINT"BITTE DATENDISKETTE INS LAUFWERK LEGEN!"
5050 PRINT
5060 INPUT"DRUECKEN SIE DANACH RETURN";X$
5070 DOPEN #1,"ADRESSEN"
5080 INPUT#1,Z
5090 FOR Y=1TO Z
5100 FOR I=1 TO 7
5110 INPUT#1,D$(Y,I)
5120 NEXT I
5130 NEXT Y
```

```
5140 CLOSE 1
5150 PRINT:PRINT"DATEN SIND GELADEN!"
5160 SLEEP 2
5170 GOTO 1000
10000 REM ====================
10010 REM   DATEI SPEICHERN
10020 REM ====================
10030 GOSUB 300:PRINT
10040 GOSUB 500:IF Z=0 THEN 1000
10050 PRINT"BITTE DATENDISKETTE EINLEGEN"
10060 PRINT
10070 INPUT"DRUECKEN SIE DANACH RETURN";X$
10075 SCRATCH "ADRESSEN"
10080 DOPEN #1,"ADRESSEN",W
10090 PRINT#1,Z
10100 FOR Y=1 TO Z
10110 FOR I=1 TO 7
10120 PRINT#1,D$(Y,I)
10130 NEXT I
10140 NEXT Y
10150 CLOSE 1
10160 PRINT:PRINT"DATEN SIND GESICHERT!"
10170 SLEEP 2
10180 GOTO 1000
15000 REM ====================
15010 REM  ADRESSEN EINGEBEN
15020 REM ====================
15030 GOSUB 300
15040 Z=Z+1
15050 PRINT
15060 FOR I=1 TO 7
15070 PRINT F$(I);
15080 INPUT D$(Z,I)
15090 NEXT I
15100 PRINT
15110 PRINT"DATEN RICHTIG EINGEGEBEN (J/N)";
15120 X$="":INPUT X$
15130 IF X$="J" THEN 15160
15140 IF X$="N" THEN Z=Z-1:GOTO 15000
15150 PRINT CHR$(145);:GOTO 15110
15160 PRINT"WEITERE EINGABEN (J/N)";
15170 X$="":INPUT X$
```

```
15180 IF X$="J" THEN 15000
15190 IF X$="N" THEN 1000
15200 PRINT CHR$(145);:GOTO 15160
20000 REM ====================
20010 REM   ADRESSEN AENDERN
20020 REM ====================
20030 ZZ=1
20040 GOSUB 300
20050 PRINT
20060 FOR I=1 TO 7
20070 PRINT I;F$(I);D$(ZZ,I)
20080 NEXT I
20090 GETKEY X$
20100 IF ASC(X$)<133 OR ASC(X$)>136 THEN 20090
20110 IF ASC(X$)=136 THEN 1000
20120 IF ASC(X$)=133 AND ZZ<Z THEN ZZ=ZZ+1:GOTO 20040
20130 IF ASC(X$)=134 AND ZZ>1 THEN ZZ=ZZ-1:GOTO 20040
20140 IF ASC(X$)=135 THEN 20160
20150 GOTO 20090
20160 PRINT
20170 INPUT"FELDNUMMER (1-7) ";X
20180 IF X<1 OR X>7 THEN PRINT CHR$(145);:GOTO 20170
20190 PRINT
20200 INPUT"NEUER INHALT: ";D$(ZZ,X)
20210 GOTO 20040
25000 REM ====================
25010 REM  ADRESSEN LOESCHEN
25020 REM ====================
25030 GOSUB 300:REM KOPF
25040 PRINT
25050 GOSUB 500:REM DATEN IM RECHNER
25055 IF Z=0 THEN 1000
25060 INPUT "VORNAME: ";D1$
25070 INPUT "NAME   : ";D2$
25080 X=1
25090 IF D$(X,2)=D1$ AND D$(X,3)=D2$ THEN 25130
25100 IF X<Z THEN X=X+1:GOTO 25090
25110 FE$="NAMEN NICHT GEFUNDEN!:"
25120 GOSUB 200:GOTO 1000
25130 GOSUB 300:PRINT
25135 FOR I=1 TO 7
25140 PRINT F$(I);D$(X,I)
```

```
25150 NEXT I
25155 PRINT
25160 INPUT"ADRESSE LOESCHEN (J/N) ";X$
25170 IF X$="J" THEN 25200
25180 IF X$="N" THEN 1000
25190 PRINT CHR$(145);:GOTO 25160
25200 FOR Y=X TO Z-1
25210 FOR I=1 TO 7
25220 D$(Y,I)=D$(Y+1,I)
25230 NEXT I
25240 NEXT Y
25250 Z=Z-1
25260 GOTO 1000
30000 REM ====================
30010 REM  ADRESSEN AUSGEBEN
30020 REM ====================
30030 GOSUB 300:PRINT
30040 GOSUB 500
30050 IF Z=0 THEN 1000
30060 INPUT"DRUCKER ODER BILDSCHIRM (D/B) ";G$
30070 IF G$<>"B" AND G$<>"D" THEN PRINT CHR$(145);:GOTO 30060
30075 IF G$="D" THEN OPEN 1,4
30080 GOSUB 300:PRINT
30090 PRINT"SUCHBEGRIFFE:"
30100 PRINT"------------"
30110 PRINT
30120 FOR I=1 TO 7:S$(I)="":NEXT I
30130 FOR I=1 TO 7
30140 PRINT F$(I);
30150 INPUT S$(I)
30160 NEXT I
30170 FOR Y=1 TO Z
30180 S=0
30190 FOR I=1 TO 7
30200 IF S$(I)="" THEN S=S+1:GOTO 30220
30210 IF D$(Y,I)=S$(I) THEN S=S+1
30220 NEXT I
30230 IF S<>7 THEN 30300
30240 IF G$="D" THEN PRINT#1
30245 IF G$="B" THEN GOSUB 300:PRINT
30250 FOR I=1 TO 7
30260 IF G$="B" THEN PRINT F$(I);D$(Y,I)
```

```
30270 IF G$="D" THEN PRINT#1,F$(I);D$(Y,I)
30280 NEXT I
30285 IF G$="D" THEN 30300
30290 PRINT:INPUT"DRUECKEN SIE RETURN";X$
30300 NEXT Y
30310 GOSUB 300:PRINT:PRINT
30320 PRINT"DATEIENDE"
30330 INPUT"DRUECKEN SIE RETURN";X$
30340 IF G$="D" THEN CLOSE 1
30350 GOTO 1000
35000 REM ====================
35010 REM   PROGRAMM BEENDEN
35020 REM ====================
35030 GOSUB 300:PRINT
35040 IF Z=0 THEN 35150
35050 PRINT"SIND ALLE DATEN GESICHERT (J/N) ";
35060 INPUT X$
35070 IF X$="N" THEN 1000
35080 IF X$="J" THEN 35100
35090 PRINT CHR$(145);:GOTO 35050
35100 GOSUB 300
35110 PRINT"DAS PROGRAMM KANN MIT 'GOTO 1000' WIEDER";
35120 PRINT"GESTARTET WERDEN, OHNE DASS DATEN VER-"
35130 PRINT"LOREN GEHEN!"
35140 PRINT
35150 END
```

DIE ZUSATZGERÄTE

Viele Einsteiger haben Probleme bei der Wahl der Geräte, die zusätzlich an das Grundgerät angeschlossen werden. Grundsätzlich ist mindestens ein externes Aufzeichnungsgerät für Programme und Daten erforderlich. Da gibt es einmal die preisgünstige Kassettenaufzeichnung mit der COMMODORE-Datassette. Dem anspruchsvollen Einsteiger bietet sich die schnelle Speicherung auf Disketten mit dem COMMODORE Diskettenlaufwerk VC-1571.

Erfahrungsgemäß hat sich erwiesen, daß fast jeder Anwender der Datassette früher oder später auf die komfortable Diskettenspeicherung umsteigt und diese nie mehr missen möchte. Es ist also lohnenswert, direkt mit einem Diskettenlaufwerk einzusteigen. Auch ist das Angebot der Software ("softwär"), also der Fertigprogramme in Diskettenform wesentlich größer, wenn man von den Spielen absieht.

Ein Drucker, mit dem man sämtliche Informationen schwarz auf weiß ausgeben kann, ist der Clou jedes Homecomputersystems. Die Auswahl der Drucker ist größer als die der zuvor erwähnten Geräte. Der Grund dafür ist, daß Drucker fast aller Hersteller an dem 128er anzupassen sind. So sind Drucker bereits ab ca. 500,- DM im Handel erhältlich.

Im folgenden Kapitel werden diese und andere Geräte genauestens beschrieben.

Die Datassette

Die Datassette arbeitet so wie ein normaler Kassettenrecorder. Sie zeichnet jedoch keine Musik, sondern digitale Signale auf das magnetische Band auf. Digital bedeutet, daß nur zwei Zeichen abgespeichert werden, 0 und 1. Diese beiden Zeichen werden als unterschiedlich hohe Tonsignale auf die Kassette aufgenommen. Jeweils acht dieser kurzen "Piepser" ergeben

ein Zeichen (Buchstabe, Ziffer usw.). Wenn Sie eine vom 128er "bearbeitete" Kassette in Ihren herkömmlichen Rekorder einlegen, so können Sie dieses ins Kreischen übergehende Piepsen hören. Mit zwei Tönen werden also Programme, Adressen uns sonstige Daten verschlüsselt. Beachten Sie jedoch, daß dieses Verfahren von Computer zu Computer unterschiedlich ist. Selbst Kassetten vom VC-20 sind nicht mit dem 128er zu laden, da der VC 20 leichte Abweichungen bei der Aufzeichnung hat. So können z.B. die beiden Tonfrequenzen unterschiedlich sein oder auch der Abstand zwischen den Zeichen.

Grundsätzlich gibt es für dieses äußerst preisgünstige Gerät zwei Einsatzmöglichkeiten:

1. **PROGRAMMSICHERUNG**
2. **DATENSICHERUNG**

Das Speichern und Laden von Programmen ist bereits beschrieben worden. Auch Datensicherung haben wir schon angesprochen. Erinnern Sie sich: Die Daten (Adressen) der Adressenverwaltung können mit den Routinen im Anhang auch auf einer Kassette gesichert werden. Da wir uns dort auf die Adressenspeicherung konzentriert haben, folgt nun nochmals eine Beschreibung der grundsätzlichen Dinge.

Voraussetzung für beide Betriebsarten - sowohl Speichern als auch Laden - ist ein Kanal. Erst wenn der Kanal zur Datassette geöffnet wird, können Daten in eine bestimmte Richtung übermittelt werden. Dazu wird der bereits beschriebene OPEN-Befehl eingesetzt. Diesem Befehl müssen drei Parameter angefügt werden:

```
OPEN fn,ga,sa,"name"
```

Da es durchaus möglich ist, mehrere Kanäle gleichzeitig offen zu halten (Drucker, Floppylaufwerk), erhält jeder Kanal eine Nummer (fn, in der Fachsprache Filenummer genannt). Diese Nummer wird dann bei den Übertragungsbefehlen angegeben, die dann den entsprechenden Kanal mit der Nummer identifizieren. Insgesamt können 255 Kanalnummern vergeben werden (1 bis 255).

Nun muß bestimmt werden, zu welchem Gerät der Kanal geöffnet werden soll. Jedes Gerät besitzt eine Geräteadresse (ga), die im OPEN-Befehl als zweiter Parameter gekennzeichnet wird. Typische Gerätenummer (ab Werk) sind:

```
Datassette :          Adresse 1
Drucker :             Adresse 4
Diskettenlaufwerk :   Adresse 8
```

Bleiben wir bei der Datassette. Soll auf dieses Gerät zugegriffen werden, so muß die Geräteadresse 1 angegeben werden.

Nun zu dem letzten Parameter, die Sekundäradresse (sa). Man redet hier zwar von Adresse, dieser Parameter hat jedoch andere Aufgaben, als irgendwas zu adressieren. Beim Drucker hat die Sekundäradresse z.B. eine andere Funktion als beim Diskettenlaufwerk oder bei der Datassette. Bleiben wir beim zuletzt genannten Gerät. Für die Datenverwaltung gibt es drei Sekundäradressen:

```
0 - KANAL ZUM LESEN ÖFFEN
1 - KANAL ZUM SCHREIBEN ÖFFNEN
2 - KANAL ZUM SCHREIBEN MIT ENDEMARKE ÖFFNEN
```

Nun zu dem dritten Parameter, der in Anführungszeichen eingeschlossen wird ("name"). Bei der Datassette entspricht dies dem Namen, unter dem die Datei geführt werden soll. Es ist somit möglich, jedoch nicht unbedingt sinnvoll, mehrere Dateieb auf einer Kassette anzulegen und diese beginnend beim Kassettenanfang zu suchen. Nehmen wir z.B. an, auf einer Kassette sind die drei Dateien mit den Namen "DATEI A", DATEI B", "DATEI C" gespeichert. Nun wollen wir die "DATEI C" lesen. Wir spulen die Kassette zurück und geben den entsprechenden OPEN-Befehl ein. Nun sucht die Datassette solange, bis die Datei gefunden ist. Wenn sie z.B. im letzen Drittel einer 60-Minuten Kassette angelegt wurde, so müssen Sie ca. 20 Minuten warten, bis die Datei gefunden ist und das

Programm fortfahren kann. Wäre Ihnen das angenehm?

Mit diesen Informationen sind wir nun in der Lage, z.B. Daten auf eine Kassette zu speichern. Wie sieht der entsprechende Befehl aus, wenn der Kanal die Nummer 1 und die Datei den Namen "TESTDATEI" erhalten soll?

```
OPEN 1,1,1,"TESTDATEI"
```

Der Kanal ist geöffnet und die Daten können übermittelt werden. Eine Datei muß sicher auch einmal gelesen werden. Dazu öffnen wir ebenfalls den Kanal, jedoch mit der Sekundäradresse 0.

```
OPEN 1,1,0,"TESTDATEI"
```

Was halten Sie vom folgenden OPEN-Befehl?

```
OPEN 1,1
```

Der 128er reagiert nicht etwa mit einer Fehlermeldung, weil die Sekundäradresse fehlt, sondern setzt intern einfach die Sekundäradresse 0 ein.

Haben Sie bemerkt, daß wir immer die Kanalnummer 1 verwendet haben? Wie reagiert der 128er darauf, wenn Sie versuchen mehrere Kanäle mit der gleichen Nummer zu öffen? Ganz einfach, er klopft Ihnen auf die Finger. Dies ist bitte nicht wörtlich zu nehmen, denn er gibt natürlich eine Fehlermeldung aus:

```
?FILE OPEN ERROR
```

Ein FILE ("fail") ist die englische Bezeichnung für Datei. Doch dies ist nicht immer der Fall. Auch ein Programm auf der Diskette wird als FILE bezeichnet. Aber wenn man genau überlegt, ist ein Programm auch nichts anderes als eine Datei. Befehle sind durchaus als Daten zu bezeichnen.

ERROR bedeuted Fehler, somit wird diese Fehlermeldung als "Datei bereits offen" interpretiert.

Öffnet man einen Kanal zum Schreiben, so muß dieser erst wieder geschlossen werden bevor man ihn zum Lesen öffnet. Doch wie schließt man einen Kanal? Wer etwas Englisch kann, ahnt sicher schon, wie der entsprechende Befehl heißt. Es ist die Anweisung CLOSE. Doch auch dieser Befehl besitzt einen Parameter.

```
CLOSE fn
```

'fn' ist wieder die Nummer des zu schließenden Kanals.

Bevor wir zu den Übertragungsbefehlen kommen, sollten Sie folgendes noch wissen: Wir haben bei den Parametern stets konstante Zahlen verwendet. Hier können Sie jedoch auch Variablen einsetzen. Folgende BASIC-Zeilen entsprechen somit dem Befehl 'OPEN 1,1,0':

```
10 KN=1:GA=1:SA=0
20 OPEN KN,GA,SA
```

Dieser Kanal kann auch mit 'CLOSE KN' geschlossen werden, vorausgesetzt daß die Variable KN noch den Wert 1 enthält.

Kommen wir nun zu den Übertragungsbefehlen. Hier gibt es drei verschiedene:

```
PRINT#fn,...        zum Schreiben der Daten
INPUT#fn,var        zum Lesen der Daten
GET#fn,...          zum Lesen der Daten
                    Zeichen für Zeichen
```

Alles was dem Komma der Befehle folgt, entspricht dem einfachen PRINT-, INPUT- oder GET-Befehl. Beim PRINT-Befehl können Zahlen, Strings, numerische Variablen und Stringvariablen eingesetzt werden. Der INPUT-Befehl läßt selbstverständlich nur die beiden zuletztgenannten Variablen zu, da die eingelesenen Daten hier gespeichert werden müssen. Beim GET-Befehl kann nur eine

Stringvariable angegeben werden, die das eingelesene Zeichen festhält. Alle folgenden Beispiele sind somit einsetzbar:

```
PRINT#1,"DUESSELDORF"
PRINT#4,A$
PRINT#2,235
PRINT#8,LA
INPUT#1,NA$
INPUT#4,SA
GET#9,X$
```

Natürlich muß die Kanalnummer (Filenummer) des OPEN-Befehls mit der des entsprechenden Übertragungsbefehls übereinstimmen.

Beim OPEN-Befehl können zwei Sekundäradressen zum Schreiben einer Datei angegeben werden.

```
1 - Schreiben ohne Endemarke
2 - Schreiben mit Endemarke
```

Was ist nun eine Endemarke? In der Fachsprache nennt man dieses Kennzeichen auch end-of-file Marke (EOF). Die Daten werden hintereinander (sequentiell) auf die Kassette geschrieben. Mit der Sekundäradresse 2 wird zusätzlich hinter dem letzten Datensatz eine Marke gesetzt, die dem lesenden Programm später mitteilt "die Datei ist zu Ende". Das Programm reagiert reagiert dann und schließt die Datei. Doch wie teilt die Datassette dem Rechner das Ende mit? Dazu gibt es beim 128er eine sogenannte Statusvariable mit dem Namen 'ST'. Diese Variable wird auf den Wert 64 gebracht, falls die EOF-Marke erreicht ist. Wann wird die Sekundäradresse 2 eingesetzt? Wenn der Programmteil, der für das Schreiben der Daten zuständig ist, vorher nicht weiß, wieviel Daten abgelegt werden, so muß die Sekundäradresse 2 verwendet werden. Wäre die Anzahl der zu übertragenden Daten bekannt, so könnte diese Zahl als Datensatzzähler am Anfang der Datei gespeichert werden. Der lesende Programmteil liest dann zuerst die Anzahl der gespeicherten Datensätze und kann das Auslesen der Datei

nach dem letzen Datensatz gezielt abbrechen. Die folgenden Beispiele bringen hier mehr Licht hinein.

Spielen wir zuerst die Methode mit der Sekundäradresse 1 durch. Große Dateien mit einheitlichem Datensatzaufbau werden wie bei der Adressenverwaltung rechnerintern in einem Array (Tabelle) gespeichert. Zu Anfang des Programms muß dieses Array aufgebaut werden (DIM-Anweisung).

```
10 REM *** PROGRAMMVORLAUF ***
20 DIM D$(100)
```

Dieses Array soll in unserem Beispiel eindimensional sein. Nun schreiben wir den Programmabschnitt zum Füllen des Arrays über die Tastatur.

```
100 REM *** DATENEINGABE A***
110 Z=1: REM DATENSATZZÄHLER
120 PRINT"EINGABE MIT '***' ABSCHLIESSEN!"
120 PRINT"DATENSATZ NR.";Z;": "
130 INPUT D$
140 IF D$="***"THEN 140
150 D$(Z)=D$
160 Z=Z+1: REM SATZZÄHLER ERHÖHEN
170 GOTO 120
180 END: REM PROGRAMM BEENDEN
```

Wir haben die Eingabe so einfach wie möglich gestaltet. Es soll nur demonstrieren, wie man die Datei im Rechner aufbaut. Soll jeder Datensatz unmittelbar nach der Eingabe abgespeichert werden, so sieht dieser Teil wie folgt aus:

```
100 REM *** DATENEINGABE B ***
110 PRINT"BITTE DATASETTE VORBEREITEN"
120 PRINT"UND RETURN DRUECKEN"
130 INPUT X$
140 PRINT"EINGABE MIT '***' ABSCHLIESSEN!"
150 OPEN 1,1,2,"TESTDATEI": REM OPEN MIT EOF
160 INPUT"DATENSATZ: ";D$
170 IF D$="***"THEN 200
```

```
180 PRINT#1,D$
190 GOTO 160
200 CLOSE 1
200 END
```

Hier wird das Array nicht benötigt. Beim späteren Einlesen wird es jedoch wieder eingesetzt. Die Datei muß mit Sekundäradresse 2 geöffnet werden, da das einlesende Programm unbedingt die Endemarke benötigt.

Übrigens: Wenn Sie diese Programmteile als Unterprogramme (Routinen) verwenden möchten, so ändern Sie den Befehl 'END um in 'RETURN'

Das mit dem ersten Abschnitt erfasste Array soll nun als Datei abgelegt werden.

```
200 REM *** SPEICHERN ***
210 PRINT"BITTE DATASETTE VORBEREITEN"
220 PRINT"UND RETURN DRUECKEN"
230 INPUT X$
240 OPEN 1,1,2,"TESTDATEI"
250 FOR I=1 TO Z: REM Z AUS EINGABE UEBERNOMMEN
260 PRINT#1,D$(Z)
270 NEXT I
280 CLOSE 1
290 END
```

Zunächst wurde die Datei zum Schreiben geöffnet. Die anschließende Schleife überträgt das Array Satz für Satz (sequentiell) auf die Kassette. Auch hier wird die Endemarke hinter dem letzen Datensatz geschrieben. Wie kann den nun ohne Endemarke, also mit Sekundäradresse 1 geschrieben werden? Dazu ist es notwendig, die Anzahl der zu speichernden Datensätze VOR den Daten aufzuzeichnen. Das lesende Programm liest dann die Anzahl und kann die Datensätze in einer gezielten Schleife einlesen.

Codieren wir nun den Teil eines Programmes, der die Daten mit Sekundäradresse 1, also ohne Endemarke und mit führender Datensatzanzahl speichert.

```
200 REM *** SPEICHERN OHNE ENDEMARKE ***
210 PRINT"BITTE DATASETTE VORBEREITEN"
220 PRINT"UND RETURN DRUCKEN"
230 INPUT X$
240 OPEN1,1,1,"TESTDATEI"
250 PRINT#1,Z: REM SPEICHERN SATZZAEHLER
260 FOR I=1 TO Z
270 PRINT#1,D$(Z)
280 NEXT
290 CLOSE1
300 END
```

Entgegen dem vorherigen Beispiel wurde hier mit Sekundäradresse 1 geöffnet und vor dem ersten Datensatz, also vor der Schleife die Anzahl der Datensätze gespeichert. Der Programmteil zum Auslesen dieser Datei ist folgender:

```
300 REM *** LESEN OHNE ENDEMARKE ***
310 PRINT"BITTE DATASETTE VORBEREITEN"
320 PRINT"UND RETURN DRUECKEN"
330 INPUT X$
340 OPEN 1,1,0,"TESTDATEI"
350 INPUT#1,Z: REM LESEN SATZZAHL
360 FOR I=1 TO Z
370 INPUT#1,D$(Z)
380 NEXT I
390 CLOSE1
400 END
```

Da in Zeile 350 die Anzahl der Datensätze eingelesen wird, kann die folgende Schleife zum Lesen der Datensätze gezielt bis zum Dateiende gesteuert werden.

Nun wollen wir eine Datei mit Endemarke lesen. Erinnern Sie sich: Das Dateiende wird an der Statusvariablen ST erkannt, die dann den Wert 64 enthält.

```
300 REM *** LESEN MIT ENDEMARKE ***
310 PRINT"BITTE DATASETTE VORBEREITEN"
320 PRINT"UND RETURN DRUECKEN"
330 INPUT X$
340 Z=1
350 OPEN 1,1,0,"TESTDATEI"
360 INPUT#1,D$(Z)
370 IF ST=64 THEN 400
380 Z=Z+1
390 GOTO 360
400 CLOSE1
410 END
```

Auch hier werden die Datensätze in einer Schleife eingelesen. Nur ist dies keine gezielte Schleife, sondern eine unbestimmte. Die Schleife wird verlassen, sobald die Endemarke erreicht ist. Dann nämlich wird ST auf 64 gesetzt und Zeile 370 führt zum Programmende.

Zum Schluß noch ein Hinweis: Dokumentieren Sie in Ihrem Programm, wie die Datei geschrieben wurde (REM-Zeilen). Abhängig von der Sekundäradresse beim Schreiben muß die Leseroutine gestaltet werden. Versucht z.B. ein Leseprogramm den Satzzähler am Anfang der Datei zu lesen, die keinen enthält (mit Sekundäradresse 2 erstellt), so gibt es Probleme. Solche Fehler sind nur schwer zu analysieren.

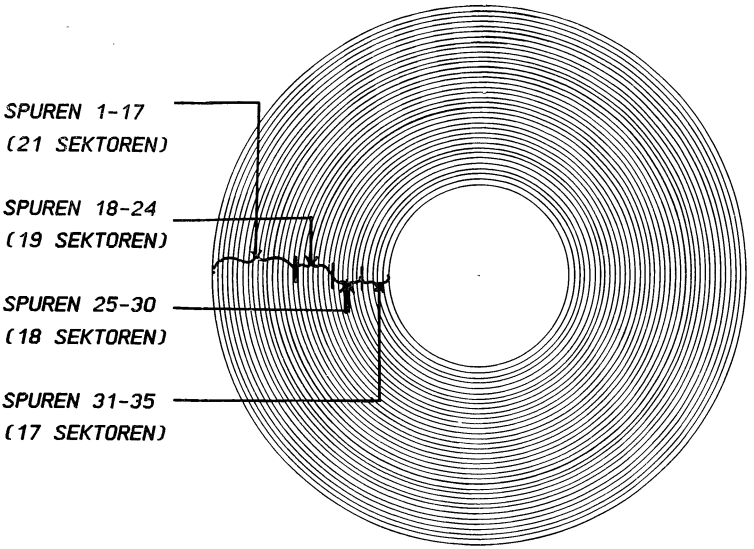
Mit Hilfe dieser Informationen werden Sie schon bald in der Lage sein, kleine Dateiverwaltungen für Schallplatten, Bücher oder ähnliches zu entwickeln.

Das Diskettenlaufwerk VC-1571

Diskettenlaufwerke sind für einen Mikrocomputer das Tüpfelchen auf dem "i". Der einfache Grund dafür ist, daß es keine andere externe Speichermöglichkeit gibt, die zu diesem Preis derartiges leistet. Die wesentlichen Vorteile der Diskettenspeicherung sind:

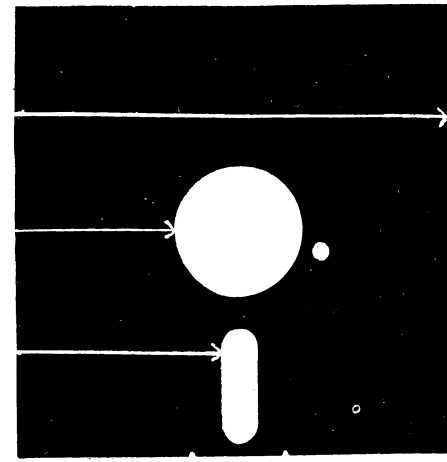
- **große Speicherkapazität**
- **direkter Zugriff auf die Daten**
- **schnelle Übertragungsgeschwindigkeit**

Disketten sind kleine, in quadratischen schwarzen Hüllen verstaute Magnetfolien in Kreisform. Diese Diskette wird von dem Laufwerk zur Speicherung vorbereitet. Sie wird dazu in Spuren aufgeteilt, die wiederum in Sektoren gegliedert werden. Die Anzahl der Spuren und der Sektoren auf einer Spur ist von dem Typ des Laufwerks abhängig. Meist werden 40 oder 80 Spuren verwendet. Bei dem COMMODORE-Laufwerk 1571 sind es 35 Spuren auf jeder Diskettenseite. Das folgende Bild verdeutlicht die Aufteilung der Diskette in Spuren und Sektoren bei der VC-1571:



Je weiter die Spur zum Mittelloch liegt, desto geringer ist ihr Umfang und die Anzahl der Sektoren. Die unterschiedliche Spurenanzahl ist notwendig, damit die Datendichte auf allen Spuren nahezu gleich ist. Sonst wäre die Fehlerquote auf den inneren Spuren größer als auf den äußeren. Den Vorgang, bei dem die gesamte Diskette in Spuren und Sektoren aufgeteilt wird, nennt man "Formatieren". Erst nach dem Formatieren können auf der Diskette Daten abgelegt werden.

Schauen wir uns nun die Öffnungen der schwarzen Diskettenhülle an:



Das Mittelloch wird im Laufwerk auf die Achse des Motors gelegt, der die Diskette ständig dreht. Durch das Mittelloch erreicht der Schreib/Lesekopf die magnetische Folie. Durch einen zweiten Motor, den sogenannten Schrittmotor, kann der Kopf auf jede beliebige Spur positioniert werden. Ein Wechsel der Spur wird also durch den Schrittmotor erreicht. Doch wie wird nun der gewünschte Sektor dieser Spur gelesen? Da die Diskette sich beim Datenzugriff ständig dreht, braucht dazu nur der Magnetkopf zur richtigen Zeit aktiviert werden. Der Kopf erkennt den adressierten Sektor an einem Vermerk, der auf jedem Sektor enthalten ist. Somit kann jeder beliebige Sektor der Diskette erreicht werden.

Die Kerbe für den Schreibschutz wird von einer Lichtschranke optisch abgetastet. Bleibt diese Kerbe offen, so kann das Licht hier passieren und die Diskette wird als nicht schreibgeschützt erkannt. Klebt man diese Kerbe jedoch mit einem Schreibschutz zu, so wird der Lichtstrahl unterbrochen und das Laufwerk erkennt die geschützte Diskette. Was bewirkt nun der Schreibschutz? Ist er angebracht, so wird jeder Schreibzugriff vom Laufwerk abgewiesen. Es kann somit nicht mehr gespeichert oder formatiert werden. Das Anbringen des Schreibschutzes ist besonders bei wertvollen Disketten wichtig. Einmal

sollte jede Programmdiskette geschützt werden, des anderen auch jede Datendiskette, von denen meistens nur gelesen wird.

Die Systembefehle der VC-1571

Die Floppy VC-1571 ist ein intelligentes Speichermedium mit einem eigenen Prozessor und einem eigenen Betriebssystem. Dieses eigene Betriebssystem, das DOS (Disk Operating System) belegt k e i n e n Platz im Speicher Ihres COMMODORE 128 und bietet trotzdem eine Reihe sehr leistungsfähiger Befehle, die den Befehlssatz Ihres COMMODORE Computers wesentlich erweitern. Eine weitere Besonderheit neben der Speicherplatzersparnis (bei fast allen anderen Computern wird das DOS in den Hauptspeicher geladen und belegt dort wertvollen Platz) ist die Tatsache, daß die Befehle des Floppy DOS von der Floppy völlig selbstständig ausgeführt werden, ohne daß Ihr Computer hiermit belastet wird.

Formatieren von Disketten

Der Befehl zum Formatieren lautet "HEADER". Der NEW-Befehl hat folgendes Format:

```
HEADER "Diskettenname",Iid
```

Der Diskettename umfaßt maximal 16 Zeichen und ist im Kopf des Directorys enthalten. Das Identifikationsmerkmal (ID) der Diskette besteht aus zwei beliebigen Zeichen, an denen das Laufwerk erkennt, ob eine andere Diskette eingelegt wurde. Da Sie dieses Idenfikationsmerkmal frei wählen können, bietet es sich gut für die Unterscheidung sonst völlig identischer Disketten an, oder aber für eine allgemeine Klassifizierung Ihrer Disketten. Wer nicht mehr als 99 Disketten hat, kann seine Disketten sehr schön an Hand des Identifikationsmerkmals ordnen.
Nun aber ein Beispiel zum Formatieren einer Diskette:

```
HEADER "TESTDISKETTE",IKL
```

Geben Sie diesen Befehl nun einmal ein, nachdem Sie eine "rohe" Diskette eingelegt haben. Da alle eventuell auf der Diskette befindlichen Daten und Programme unwiederbringlich durch das Formatieren gelöscht werden, fragt Ihr 128er Sie, ob Sie die Diskette wirklich formatieren wollen - "ARE YOU SURE?". Wenn Sie sicher sind, so beantworten Sie diese Sicherheitsabfrage mit "Y" für YES. Sie werden feststellen, daß das Laufwerk nun mit dem Formatieren beginnt. Dieser Vorgang dauert ca. 40 Sekunden.

Auslesen des Fehlerkanals

Wie Ihnen sicher bekannt ist, gibt der Rechner bei nicht ordnungsgemäßer Programmierung Fehlermeldungen aus. Da die Diskettenbefehle aber nicht von dem Prozessor des Rechners, sondern von dem des Laufwerks überprüft und ausgeführt werden, kann der Rechner die Fehlermeldungen des Laufwerks nicht anzeigen. Fehlermeldungen werden vom Anwender an der aufblinkenden roten Leuchtdiode am

Laufwerk erkannt. Um jedoch festzustellen, welcher Fehler aufgetreten ist, muß die Systemvariable DS$ gelesen werden. Diese Systemvariable kann nur gelesen und nicht beschrieben werden. Sollten Sie versuchen DS$ einen String zuzuordnen, so werden Sie die Fehlermeldung: SYNTAX ERROR erhalten.
Geben Sie einmal den Befehl:

```
PRINT DS$
```

Wenn die rote Leuchtdiode an der 1571 nicht geblinkt hat, also kein Fehler vorlag, so erhalten Sie die "Fehlermeldung":

```
00, OK,00,00
```

Die vier durch Komata getrennten Felder haben folgende Bedeutung:

1. Feld: Nummer des Fehlers (numerisch)
2. Feld: Bezeichnung des Fehlers (alphanumerisch)
3. Feld: Spur (numerisch)
4. Feld: Sektor (numerisch)

Die Spur- und Sektorangabe bezeichnet, wo der Fehler lokalisiert wurde.
Um die Wirkungsweise dieser Systemvariablen zu erkennen, verursachen Sie bitte folgenden Fehler:

```
DLOAD "XYZ"
```

Wenn Sie diese Befehlsfolge eingegeben haben, blinkt die rote Leuchtdiode an dem Floppy-Laufwerk, wenn Sie nicht zufällig ein Programm mit dem Namen "XYZ" auf Ihrer Diskette gespeichert haben. Lesen Sie nun die Systemvariablen DS$ mit den Befehl: PRINT DS$. Auf dem Bildschirm erscheint dann die Meldung:

```
62, FILE NOT FOUND,00,00
```

Die 62 ist die Nummer des Fehlers, dessen Klartext dann folgt. Das Feld Spur und Sektor ist 0, weil dieser Fehler

diese Angaben nicht benötigt.
Sollte ohne daß ein Fehler aufgetreten ist, die Systemvariable DS$ ausgelesen werden, so wird die Meldung

```
00 OK 00 00
```

ausgegeben.

Falls während der Arbeit mit der Floppy-Station die rote Leuchtdiode blinken sollte, so überprüfen Sie erst Ihren Befehl, denn meistens ist der Fehler wie beim o.g. Beispiel leicht zu erkennen. Andernfalls lesen Sie einfach den Fehlerkanal aus.

Laden des Directory

Das Directory ist das Inhaltsverzeichnis der Diskette. Hier sind alle Files (Programme und Dateien) der Diskette katalogisiert.
Das Directory wird mit

```
DIRECTORY
```

auf den Bildschirm ausgegeben. Probieren Sie es nun einmal mit der dem Laufwerk beigefügten Test/Demo-Diskette aus. Legen Sie diese Diskette in das Laufwerk und geben Sie den o.g. Befehl zum Laden der Directory ein. Es erscheint dann wie folgt auf dem Bildschirm: (Bitte beachten Sie, daß nicht alle VC-1571 mit derselben Test/Demo-Diskette geliefert werden, da COMMODORE auch hier manchmal nicht angekündigte Änderungen vornimmt).

```
0    "1541test/demo   " zx 2a
13   "how to use"      prg
5    "how part two"    prg
4    "vic-20 wedge"    prg
1    "c-64 wedge"      prg
4    "dos 5.1"         prg
11   "copy/all"        prg
4    "disk addr change" prg
4    "dir"             prg
```

```
6    "view bam"          prg
4    "check disk"        prg
14   "display t&s"       prg
9    "performance test"  prg
5    "sequential file"   prg
13   "random fial"       prg
```

Diesem Directory sind viele Informationen zu entnehmen. Sehen wir uns die 1. Zeile, den Kopf des Directory, einmal an. Das Zeichen '0' in dieser Zeile hat keine besondere Bedeutung. Daneben ist der Name und die ID der Diskette angegeben, wie es bei der Formatierung vereinbart wurde. Die Zeichen '2A' symbolisieren das Diskettenformat. Ist dieses Format nicht '2A', so ist diese Diskette auch nicht auf dieser Art Laufwerk formatiert worden.

Nun folgen die einzelnen Files mit Ihrer Blocklänge am Anfang und dem Filetyp am Ende der Zeile. Auf dieser Diskette erkennen Sie 3 verschiedene Filetypen die im folgenden erklärt werden. Auf die restlichen Filetypen wird später noch eingegangen.

PRG — Dies sind PROGRAM-FILES, d.h. Programme in BASIC oder Maschinensprache

SEQ — So werden sequentielle Dateien gekennzeichnet, die später beschrieben werden

REL — Dies ist eine andere Form der Datenspeicherung, deren Beschreibung den Rahmen dieses Buches sprengen würde. (siehe "Das große Floppy-Buch")

Die Länge der Files ist in Blöcken angegeben, von denen jeder 256 Bytes umfasst.So kann man leicht die Größe eines Programms ermitteln. Man muß lediglich von den 256 Bytes eines jeden Blocks 2 Bytes abrechnen, die für interne Zwecke benötigt werden.

Am Ende des Directory ist dann noch die Anzahl der noch freien Blöcke der Diskette ersichtlich.

Löschen von Files

Natürlich muß die Möglichkeit bestehen, nicht mehr benötigte Files zu löschen. Dazu ist der Befehl 'SCRATCH' vorgesehen. Bevor dieser Befehl angewandt wird, sollte man sich stets überzeugen, daß der im SCRATCH-Befehl angegebene Name auch mit dem des zu löschenden Files übereinstimmt. Ein unabsichtlich gelöschtes File kann die Arbeit von mehrerern Stunden oder sogar Tagen zunichte machen.

Zum Löschen eines Files muß das folgende Format des Befehls beachtet werden:

```
SCRATCH "Filename"
```

Wichtig ist die Tatsache, daß dem Floppy-Befehlskanal innerhalb der Anführungszeichen nicht mehr als 16 Zeichen mit einem Befehl übermittelt werden können!
Um z.B. ein File mit dem Namen 'TEST' zu löschen, wird folgender Befehl eingegeben:

```
SCRATCH "TEST"
```

Der Systemvariablen DS$ wird die Meldung

```
01 FILES SCRATCHED nn 00
```

übergeben. 'nn' ist die Anzahl der gelöschten Files.

Umbenennen von Files

Um Files einen anderen Namen zu geben wird der Filename im Fileeintrag der Directory geändert. Der Befehl 'RENAME' ist dafür zuständig. Er hat das folgende Format:

```
RENAME "alter Name" TO "neuer Name"
```

Wenn z.B. das File mit dem Namen "TEST" umbenannt werden soll in "TEST.01", so verwenden Sie den Befehl

```
RENAME "TEST" TO "TEST.01"
```

Ein File, das eröffnet, aber noch nicht abgeschlossen wurde, kann nicht umbenannt werden!

Verbinden von Files

Aus mehreren sequentiellen Files kann ein neues File gebildet werden. Wenn Sie z.B. jeden Monat eine sequentielle Datei der Ausgaben in Ihrem Haushalt erstellt haben und diese mit den Namen AUSG.01, AUSG.02 usw. gekennzeichnet sind, so kann mit einigen Befehlen eine Datei der Ausgaben im ersten Quartal (z.B. AUSG.Q1) des Jahres gebildet werden. Da der Befehl das Format

```
CONCAT "Name1" TO "Name2"
```

hat, kann die Zusammensetzung der genannten Dateien mit folgenden Befehlen erfolgen:

```
CONCAT "AUSG.01" TO "AUSG.Q1"
CONCAT "AUSG.02" TO "AUSG.Q1"
CONCAT "AUSG.03" TO "AUSG.Q1"
```

Diese Methode des Mischens von Dateien kann bei Programmen nicht angewendet werden. Hier kann nur ein Programm innerhalb der Diskette kopiert werden. Der Name des neuen Files darf nicht schon auf der Diskette enthalten sein.

Kopieren von Files

Dieser COPY-Befehl findet selten Anwendung. Der Grund dafür ist, daß das Kopieren eines Files auf dieselbe Diskette eigentlich keinen Sinn hat. Wenn Sie allerdings eine Diskettenstation mit einem Doppellaufwerk besitzen, so ist dieser Befehl sehr nützlich. Das Befehlsformat lautet:

```
COPY "Name1" TO "Name2"
```

Die Jokerzeichen

Es gibt zwei Jokerzeichen: Den Stern (*) und das Fragezeichen (?). Der Stern an einer bestimmten Stelle des Filenamens symbolisiert, daß das erste File auf der Diskette relevant ist, das mit den Zeichen vor dem Stern beginnt. Ein Beispiel:

```
DLOAD "TEST*"
```

Dieser Befehl lädt das erste Programm, dessen erste vier Buchstaben "TEST" beinhalten. Der Befehl

```
DLOAD "*"
```

lädt das erste Programm der Diskette, da kein Zeichen vor dem Stern angegeben ist. Der Stern in einem SCRATCH-Befehl hat eine andere Funktion. Hier wird nicht das erste File gelöscht, sondern ALLE. Z.B. löscht der Befehl

```
SCRATCH "TEST*"
```

alle Files, die mit den Buchstaben "TEST" beginnen. Dies ist unbedingt zu beachten! Auch das Laden der Directory kann mit dem Stern selektiert erfolgen. Ein Beispiel:

```
DIRECTORY "A*"
```

zeigt nur das Directory mit den Files, die mit dem Buchstaben "A" beginnen.
Das DOS bietet eine weitere Einsatzmöglichkeit des Sterns: Es können auch Filetypen selektiert werden, wenn nach dem Stern ein Gleichheitszeichen mit anschließendem ersten Buchstaben des gewünschten Filetyps angegeben wird. Hier eine Übersicht:

```
*=S     selektiert nur sequentielle Files
*=P     selektiert Programmfiles
*=R     selektiert relative Files
*=U     selektiert User-Files
```

Geben Sie z.B.

```
DIRECTORY "*=P"
```

ein, so werden nur die Programme auf der Diskette ausgegeben. Auch können mit dem SCRATCH-Befehl z.B. alle sequentiellen Files auf der Diskette mit folgendem Befehl gelöscht werden:

```
SCRATCH "*=S"
```

Natürlich kann vor diesem Stern auch noch eine Zeichenfolge angegeben werden, sodaß dann nur die sequentiellen Files gelöscht werden, deren Namen mit dieser Zeichenfolge beginnen.

Mit dem Fragezeichen können im Filenamen Buchstaben an beliebigen Stellen als "nicht relevant" gekennzeichnet werden. Um die Funktion des Fragezeichens zu erläutern, folgen nun zwei Beispiele von abgekürzten Filenamen und ihren Auswirkungen:

```
A????       - fünfstellige Filenamen, deren erster
              Buchstabe "A" ist, sind angesprochen
????TEST    - achtstellige Filenamen, deren letzte vier
              Buchstaben "TEST" beinhalten, sind
              angesprochen
```

Eine Kombination von Stern und Fragezeichen ist erlaubt. Jedoch sollte beachtet werden, daß nach dem Stern weder Buchstaben, noch Fragezeichen folgen, da diese Kombinationen keinen Sinn ergeben. Zwei Beispiele zur Kombination von Stern und Fragezeichen:

```
????.*      - alle Filenamen, die vor dem Punkt vier
              Buchstaben besitzen, sind angesprochen
TEST.??*    - alle mindestens 7-stellige Filenamen,
              deren erste fünf Zeichen "TEST."
              beinhalten, sind angesprochen.
TEST-??01*=S - alle mindestens 9-stelligen,
              sequentiellen Files, deren Namen in den
              ersten 5 Stellen "TEST-" und in den
```

Stellen 6 bis 7 "01" enthalten, sind angesprochen

Sequentielle Datenverwaltung mit der Floppy

Ein Diskettenlaufwerk sollte nicht ausschließlich zur Programmspeicherung genutzt werden. Spätestens dann, wenn Sie eigene Programme schreiben, die eine große Datenmenge zu verwalten haben, werden Sie eine schnelle Datenorganisation benötigen. Die sequentielle Datenspeicherung ist zwar nicht die schnellste, aber die einfachste Methode, Daten zu verwalten, was gerade für Anfänger wichtig sein dürfte. Diese Datenorganisation ist vergleichbar mit der sequentiellen Datenspeicherung auf Kassette.

Es ist selbstverständlich, daß die maximale Datenmenge von der Größe des Speichers im Rechner abhängig ist, da ein Datensatz in einer sequentiellen Datei nicht direkt auf der Diskette oder Kassette geändert oder gelöscht werden kann. Dazu muß die gesamte Datei eingelesen, geändert und wieder abgespeichert werden. Das Laden und Speichern der Datei geschieht bei Einsatz eines Diskettenlaufwerkes wesentlich schneller als bei einem Kassettenlaufwerk. Dies ist der erste Vorteil der Datenspeicherung mit Diskette.
Der zweite Vorteil ist, daß zum Anfügen eines Datensatzes an eine sequentielle Diskettendatei nicht die gesamte Datei eingelesen werden muß. Hierzu wird der APPEND #x-Befehl eingesetzt. Dies ist bei der Speicherung auf Kassette nicht möglich.

Bei der Floppystation wird jede sequentielle Datei grundsätzlich mit der Endemarke abgespeichert. Sie können aus diesem Grunde beide Methoden zur Datenübermittlung aus dem Abschnitt "Die Datasette" verwenden.

Das Speichern und Laden von sequentiellen Dateien läuft genauso ab wie bei der Datasette. Lediglich der OPEN-Befehl der Floppy weicht erheblich von der Datasette ab.

Das Eröffnen einer sequentiellen Datei

Um eine Datei zu erstellen, muß sie vorher geöffnet werden. Beim Öffnen zum Beschreiben wird Folgendes durchgeführt:

1. Es wird geprüft, ob auf der Diskette bereits ein File mit diesem Namen existiert. Wenn ja, wird die Fehlermeldung "FILE EXISTS" in DS$ ausgegegeben.

2. Der entsprechende Fileeintrag in der Directory wird angelegt. Dabei wird im Filetyp gekennzeichnet, daß dieses File noch nicht geschlossen ist, was dann in der aufgelisteten Directory durch einen Stern vor dem Filetyp ersichtlich ist.

3. Es wird ein freier Block gesucht, auf dem die ersten Daten gespeichert werden. Die Adresse (Spur und Sektor) wird im Fileeintrag gespeichert.

4. Die Anzahl der Blocks im File wird auf 0 gesetzt, da noch kein Block dieses Files beschrieben ist.

Nach dem Erstellen der Datei kann diese dann geändert oder erweitert werden. Im OPEN-Befehl wird festgelegt, zu welchem Zweck die Datei geöffnet werden soll. Das Format des OPEN-Befehls sieht folgendermaßen aus:

```
DOPEN #fn,"Filename",Modus
```

Die Filenummer liegt zwischen 1 und 255. Der Modus bezeichnet um welche Art von Datei es sich handelt und ob die Datei zum Lesen oder Schreiben eröffnet werden soll.
Sollten mehrere Dateien gleichzeitig geöffnet sein, so muß die Filenummer unbedingt unterschiedlich sein, da immer nur ein Kanal für eine Datei zuständig sein kann. Andernfalls wird die Fehlermeldung 'FILE ALREADY OPEN' ausgegeben. Es können auch nur maximal 3 Kanäle mit jeweils einer Datei geöffnet werden.

Bei der Angabe des Filenamens ist darauf zu achten, daß dieser Filename nicht bereits auf der Diskette existiert.

Soll eine Datei zum Schreiben geöffnet werden, die bereits auf der Diskette existiert, so muß wie bei dem Befehl 'DSAVE' dem Filenamen der Klammeraffe mit anschließendem Doppelpunkt vorangestellt werden!. Ein Beispiel:

```
DOPEN #1," :ADRESSEN",W
```

Bei der Eröffnung der Datei muß der Modus 'W' für eine sequentielle Datei, in welche geschrieben werden soll, angegeben werden.

Für den Modus gibt es im Zusammenhang mit sequentiellen Dateien folgende Möglichkeiten:

```
keine Angabe - Sequentielle Datei zum Lesen
           W - Schreiben in eine sequentielle Datei
```

Öffnen Sie nun einmal eine sequentielle Datei mit dem Namen "SEQU.TEST" zum Schreiben:

```
DOPEN #1,"SEQU.TEST",W
```

Wenn Sie anschließend mit 'DIRECTORY' das Directory ansehen, werden Sie feststellen, daß dieses File mit einem Stern vor dem Filetypen als geöffnet gekennzeichnet ist:

```
0   SEQU.TEST          *SEQ
```

Diese Datei läßt sich nun aber nicht mehr schließen! Bevor also nach dem Eröffnen und Beschreiben einer Datei das Directory geladen wird, muß unbedingt das File mit dem CLOSE x-Befehl geschlossen werden!
Der Befehl 'DCLOSE' hat zur Folge, daß alle Files geschlossen werden.

Der Drucker

Daß ein Drucker früher oder später zu jedem professionellen Computersystem gehört, braucht nicht mehr diskutiert zu werden. Wir wollen uns nun der Anwendung eines Druckers zuwenden. Dieser Einstieg ist auf jedem Drucker praktizierbar. Nähere Beschreibungen über die Möglichkeiten, die alle an den 128er anschließbare Drucker bieten, würden den Rahmen dieses Buches sprengen.

Da das Druckerangebot sehr groß ist, sollten Sie schon vor dem Kauf wissen, was IHR Drucker leisten sollte. Grundsätzlich gibt es zwei verschiedene Druckerarten:

- **Typenraddrucker**
- **Matrixdrucker**

Typenraddrucker haben ein sehr sauberes Schriftbild und sind aus diesem Grund sehr gut für die Korrespondenz geeignet. Ihr Nachteil ist die relativ langsame Geschwindigkeit von ca. 10 bis 40 Zeichen in der Sekunde. Erfahrene Sekretärinnen werden zwar auch diese Geschwindigkeit bewundern, jedoch bringt es ein Matrixdrucker auf immerhin 200 Zeichen je Sekunde. Außerdem kann ein Typenraddrucker nicht alle Sonderzeichen die sich auf der Computer-Tastatur befinden ausdrucken.

Das Schriftbild eines Matrixdruckers ist sicher jedem bekannt, der schon einmal Computerabrechnungen wie z.B. die eines Elektrizitätswerkes näher betrachtet hat. Wie auf dem Bildschirm Ihres 128ers setzen sich die Zeichen eines Matrixdruckers aus einzelnen Punkten zusammen. Wie schon gesagt, arbeitet der Matrixdrucker mit enormen Geschwindigkeiten von bis zu 200 Zeichen pro Sekunde. Daher werden sie gerne überall dort eingesetzt, wo die Qualität des Schriftbilds keine allzu große Rolle spielt. Moderne Matrixdrucker erreichen allerdings mit sogenannter NLQ (Near Letter Quality - Schönschrift) ein vom Typenraddrucker oft kaum zu unterscheidendes Schriftbild.

Kommen wir nun zu der Bedienung des Matrixdruckers, nachdem er an den 128er angeschlossen ist.
Wie sie bereits der Adressenverwaltung entnehmen konnten, ist zunächst ein OPEN-Befehl erforderlich.

```
OPEN fn,4,sa
```

Es muß also eine Filenummer, die Geräteadresse 4 und eine Sekundäradresse eingegeben werden. Die Sekundäradresse ist es, die schon viele Anwender eines Druckers zum Verzweifeln gebracht hat. Der Grund dafür ist, daß fast sämtliche Hersteller andere Sekundäradressen für verschiedene Betriebsarten verwenden. Welche Betriebsarten gibt es?. Sehen Sie selbst:

groß / Grafik
groß / klein
druckereigener Kanal

Die beiden erstgenannten Betriebsarten sind uns bereits bekannt. Über den druckereigenen Kanal kann man Befehle an den Drucker übermitteln. So gibt es Drucker die verschiedene Schriften, hochauflösende Grafik usw. beherrschen.

Die Betriebsarten groß/Grafik und groß/klein sollten Mindestvoraussetzung für einen Drucker sein. Meist wird dies erst über ein sogenanntes INTERFACE ermöglicht. Dies ist eine elektronische Schaltung, die einen Drucker mit einem besonderen Anschluß an den 128er anpasst.

Nachdem Sie nun den Kanal zum Drucker wunschgemäß geöffnet haben, können Sie mit dem Befehl 'PRINT#fn,....' ausdrucken, was immer Sie wollen. Dieser Befehl ist genau wie der Befehl PRINT für den Bildschirm einzusetzen.

Nach der Ausgabe muß der Kanal wieder geschlossen werden. Der Befehl 'CLOSE fn' ist Ihnen nicht mehr unbekannt.

Sie können nun Daten auf den Drucker ausgeben. Aber was ist z.B. mit dem Programmlisting? Die Übersicht auf dem Bildschirm ist etwas umständlich. Um alle Ausgaben, die

auf den Bildschirm gehen, auf den Drucker zu lenken, gibt es einen speziellen Befehl.

```
CMD fn
```

Dieser Befehl leitet die Bildschirmausgabe auf das mit 'fn' im OPEN-Befehl bestimmte Gerät um. Ein Beispiel: Laden Sie ein BASIC-Programm (z.B. die Adressenverwaltung). Die folgenden Befehle werden im Direktmodus eingegeben und jedesmal mit RETURN abgeschlosssen.

```
OPEN 1,4
CMD 1
LIST
PRINT#1
CLOSE 1
```

Leichter ist es, wenn alle Befehle in einer Zeile eingegeben werden:

```
OPEN 1,4:CMD 1:LIST:PRINT#1:CLOSE 1
```

Hier gibt man mit einem RETURN die gesamte Zeile zur Verarbeitung frei.

Es gibt jedoch noch weitere Möglichkeiten, den Befehl CMD einzusetzen. So läßt sich z.B. das Directory der Diskette auf den Drucker ausgeben, um es evtl. auf die Diskettenhülle zu kleben. Sie müssen nur vor der gerade beschriebenen Befehlsfolge das Directory mit dem Befehl 'LOAD "$",8' in den Rechner laden.

Auch beim Einsatz des Druckers ist noch kein Meister vom Himmel gefallen. Je mehr Sie sich Ihren Geräten zuwenden, desto leichter fällt Ihnen die Bedienung.

Der Joystick

Wer kennt diese "Spaßstöcke" nicht, die bei Videospielen erstmals eingesetzt wurden. Heute gibt es kaum ein Spiel für den Homecomputer, das nicht über den Joystick gesteuert wird. Die Auswahl ist groß und reicht vom leichten COMMODORE-Joystick über Ausführungen mit Pistolengriff bis zu Exemplaren mit Quecksilberschaltern.

Wer mit dem 128er spielen möchte, kommt bis auf wenige Ausnahmen nicht um einen Joystick herum. Wer diesen Knüppel einmal in seinen eigenen Programmen einsetzen möchte, muß mit dem 'JOY'-Befehl arbeiten. Das nachfolgende Programm dürfte diesen Befehl ausreichend erklären:

```
10 A=JOY(2)
20 IF A=0 THEN 10
30 IF A>127 THEN A=A-128:PRINT"FEUERKNOPF",
40 IF A=1 THEN PRINT"NACH VORN"
50 IF A=2 THEN PRINT"NACH RECHTS VORN"
60 IF A=3 THEN PRINT"NACH RECHTS"
70 IF A=4 THEN PRINT"NACH RECHTS HINTEN"
80 IF A=5 THEN PRINT"NACH HINTEN"
90 IF A=6 THEN PRINT"NACH LINKS HINTEN"
100 IF A=7 THEN PRINT"NACH LINKS"
110 IF A=8 THEN PRINT"NACH LINKS VORN"
120 PRINT:GOTO 10
```

In Zeile 10 wird der Variablen 'A' der Wert der Funktion 'JOY' zugeordnet. Die '2' in den Klammern des JOY-Befehls zeigt an, daß der Joystick in Port 2 abgefragt werden soll. In Zeile 20 wird solange nach Zeile 10 zurückgesprungen, wie der Joystick nicht bedient wird (JOY(2)=0). Wurde der Feuerknopf betätigt, so wird zu dem Wert von JOY 128 hinzuaddiert. Das wird in Zeile 30 entsprechend berücksichtigt. Ab Zeile 40 wird dann die vollzogene Joystickbewegung ausgegeben.

Sie müssen als Einsteiger dieses Programm nicht bis ins Letzte verstehen. Es soll Ihnen nur als Experimentiermaterial dienen. So können Sie z.B. die

PRINT-Befehle gegen andere Anweisungen ersetzen.

INTERNES

Der Hauptspeicher

Wie der Name des 128ers schon sagt, besitzt er 128 KByte Speicher. Da ein KByte 1024 Bytes entspricht, hat der 128er einen Speicher mit 131072 Speicherstellen. Was ist nun hier alles untergebracht? Die Begriffe Betriebssystem und BASIC-Interpreter sind in diesem Buch schon gefallen. Doch was haben diese beiden Programme für eine Aufgabe?

Das Betriebssystem des 128er umfaßt 8 KByte. Es ist z.B. für die Ein- und Ausgaben des 128ers zuständig. Das Betriebssystem sorgt auch dafür, daß der Cursor blinkt und daß mit der Taste CLR/HOME der Bildschirm gelöscht wird.

Erst wenn Sie BASIC-Programme oder Anweisungen einsetzen, arbeitet der BASIC-Interpreter, der 32 Kbyte umfaßt. Er erkennt die Befehle und führt die entsprechenden Routinen aus. Für jeden Befehl gibt es intern eine Reihe von Maschinenbefehlen. Der Prozessor, das "Gehirn" des 128ers versteht nur diese Maschinenbefehle. So kann es sein, daß eine mathematische Berechnung intern ca. 200 Maschinenbefehle abarbeiten läßt.

Dem BASIC-Programmierer stehen 58109 Bytes für Programme und ohne Erweiterung 64256 Bytes für Variablen (Daten) zur Verfügung. Doch wer nutzt diesen Bereich schon aus?

PEEK und POKE

Sicher haben Sie diese Befehle bereits in Programmen gesehen, wie sie z.B. in den Fachzeitschriften enthalten sind. Je mehr dieser Befehle eingesetzt werden, desto komplizierter und undurchsichtiger wird das Programm. Obwohl es BASIC-Befehle sind, stehen Sie in keinem unmitelbaren Zusammenhang mit dieser doch so einfachen

Programmiersprache. Es sind auch die einzigen Befehle, die nicht auf einem anderen Rechner mit den gleichen Werten übernommen werden können.

Die Vorgänger des COMMODORE 128 (VC 20, C 64 usw.) mußten oft mit diesen Befehlen programmiert werden um die Grafik- und Soundmöglichkeiten nutzen zu können. Bei dem umfangreichen BASIC des COMMODORE 128 ist das nicht mehr erforderlich. Es wird daher hier auch nicht näher darauf eingegangen, da es Sie nur unnötig belasten würde.

Der SYS-Befehl

Dieser Befehl ermöglicht das Einbinden von Maschinenprogrammen in BASIC. Ein Maschinenprogramm ist ein in der Sprache des Prozessors geschriebenes Programm, das höllisch schnell arbeitet. Mit SYS kann man solche Programme von BASIC aus starten. Ist das Maschinenprogramm abgelaufen, so läuft das BASIC-Programm weiter.

Machen Sie sich zunächst keine Gedanken über diesen Befehl. Sie sollten ihn nur kennen, falls Sie ihn einmal in einem BASIC-Programm entdecken sollten.

ANWENDUNGSBEISPIELE

Dieses Kapitel soll Sie bei der Auswahl im Handel erhältlicher Programme unterstützen. Viele Gebiete des Softwaremarktes werden beschrieben und die wichtigsten Kriterien erläutert. Kurzum ist dieses Kapitel eine echte Hilfe vor dem Kauf.

Dateiverwaltung

Die Anwender des Commodore 128 haben oft die unterschiedlichsten Interessen. Es gibt jedoch eine Art von Programmen, die jeder einsetzen kann. Dies ist die Dateiverwaltung, die es ermöglicht, beliebiges zu katalogisieren. Hier nur wenige Beispiele:

- **die Schallplattensammlung eines Musikfreundes**
- **die Kunden eines Versicherungsvertreters**
- **die Autos eines Gebrauchtwagenhändlers**
- **die Kochrezepte der Hausfrau**

Bei einer Datenverwaltung sind die zu verwaltenden Daten vollkommen variabel. Sowohl die Länge eines Datensatzes, als auch die Anzahl und Länge der einzelnen Felder kann vom Anwender individuell angepaßt werden. So wie eine Karteikarte per Hand entworfen wird, können Sie Ihre "Datenmaske" auf dem Bildschirm aufbauen. Als Maske bezeichnet man den Inhalt des gesamten Bildschirms. Ein Beispiel:

```
NAME: .....      VORNAME: .............

STRASSE: ..........................
PLZ/ORT: .... ....................

OFFENE RECHNUNGEN: ...............
ZULETZT BESTELLT:  ........
UMSATZ JAHR:       ........
```

Diese Maske wird einmal erstellt und dann auf einer Diskette festgehalten. Wenn Sie nun später z.B. Daten eingeben möchten, so erscheint immer diese Maske, die dann nur noch ausgefüllt wird.

Der Clou an einer Dateiverwaltung ist eine ausgeklügelte Datenauswertung. So ist es möglich, die Datei nach beliebigen Kriterien zu duchsuchen und die gefundenen Datensätze in Listenform auf dem Drucker auszugeben. Auch dazu erscheint wieder die Maske, die Sie mit den gewünschten Kriterien füllen. So ist es z.B. möglich, alle Adressen im Postleitzahlengebiet 4 zu suchen, die gleichzeitig am 2. März Geburtstag haben.

Bei großen Datenmengen werden Dateiverwaltungen auf dem 128er oft langsam. Die relativ langsame Übertragung des Diskettenlaufwerkes ist mit der Grund dafür. Der Zeitaufwand beim Verwalten der Daten auf Diskette steigt mit der Datenmenge auf der Diskette. Doch wer nicht seine 10.000 Kunden verwalten möchte, sondern "nur" 500 Schallplatten, der ist mit einer Dateiverwaltung auf dem 128er bestens bedient. Das Programm DATAMAT z.B. kann die Kapazität einer gesamten Diskette für eine Datei nutzen! Neben den beschriebenen Eigenschaften verfügt DATAMAT über eine Reihe weiterer komfortabler Programmfunktionen. So kann die Datei z.B. nach jedem beliebigen Feld sortiert und auf dem Drucker ausgegeben werden.

Textverarbeitung

Welcher 128er Anwender träumt nicht davon, seine Briefe oder seine Hausaufgaben mit dem 128er zu schreiben? Die Texte werden einfach auf dem Bildschirm erfaßt, korrigiert und können dann auf einer Diskette abgespeichert werden. Zum beliebigen Zeitpunkt kann man diesen Text wieder einlesen, drucken, ändern, usw. Das Erstellen von Standardbriefen, die in gleicher oder ähnlicher Form anfallen, bietet sich hier an.

Eine Textverarbeitung und natürlich ein Drucker sind dafür erforderlich. Die Auswahl der Textverarbeitungen

für den 128er wird in Kürze sehr umfangreich sein. Die Entscheidung zum Kauf fällt somit nicht leicht. Wie unterscheiden sich die verschiedenen Textverarbeitungsprogramme für den 128er?

Ein wichtiger Punkt ist der Zeichensatz. Amerikanische Textverarbeitungsprogramme verfügen z.B. nicht über deutsche Umlaute. Da es beim 128er kein Problem ist, den Zeichensatz umzustellen, gibt es auch Programme mit deutschem Zeichensatz. Hier gibt es jedoch ein Problem: Die deutschen Zeichen müssen auch auf dem Drucker ausgegeben werden. Viele Drucker für den 128er haben keinen deutschen Zeichensatz. Was nun? Das Programm TEXTOMAT besitzt für jeden handelsüblichen Drucker für den 128er intern ein Programm, das es ermöglicht, auch auf einem amerikanischen Matrixdrucker deutsche Zeichen auszugeben.

Ein weiteres Kriterium ist die Zeilenbreite. Die meisten Textprogramme arbeiten mit dem normalen 40-Zeichen-Bildschirm des 128er. Dies hat sich aber als unpraktisch erwiesen, da eine Druckerzeile in zwei Bildschirmzeilen dargestellt wird. So können Tabellen, die breiter als 40 Zeichen sind, nur sehr umständlich auf dem Bildschirm dargestellt werden. TEXTOMAT bietet hier eine elegante Lösung. Es kann alternativ auf dem 80- oder dem 40-Zeichen Bildschirm gearbeitet werden.

Bei soviel programmtechnischen Voraussetzungen stellt sich natürlich die Frage der Bedienerfreundlichkeit. Nimmt man z.B. einen einfachen Befehl zum Löschen des Textes. Da gibt es Textverarbeitungen, bei denen dazu eine Kombination von 5 Tasten eingegeben werden muß, die leicht in Vergessenheit gerät. TEXTOMAT z.B. ist menügesteuert. Das heißt, der Anwender erreicht Schritt für Schritt das Menü, in welchem er den Text löschen kann. Befehle werden hier nicht mit Tasten symbolisiert oder abgekürzt, sondern im Menü ausgeschrieben. So kann man TEXTOMAT fast ohne Handbuch "erforschen".

Sehr komfortabel ist die Verbindung einer Dateiverwaltung mit einer Textverarbeitung. So können mit der

Dateiverwaltung z.B. Adressen ausgewertet werden, die dann von der Textverarbeitung übernommen werden können. So ist es z.B. möglich, Rundschreiben zu erstellen, die nur an die vorher mit der Dateiverwaltung ausgewählten Adressen gesendet werden. Mit TEXTOMAT und DATAMAT ist dies möglich.

Wie gesagt, ist ein Drucker unbedingt zum Einsatz einer Textverarbeitung erforderlich. Doch gibt es so viele Drucker, die zwar an den 128er anschließbar sind, jedoch bei vielen Programmen versagen. Dazu sollte eine Textverarbeitung ein Programm zur Druckeranpassung enthalten. Dieses Programm legt dann z.B. die Geräte- und Sekundäradresse fest. Doch dies reicht nicht aus. So können manche Drucker ganz andere Zeichen als die gewünschten ausgeben. TEXTOMAT besitzt hier eine Tabelle, mit der die Zeichen des Programms an die des Druckers angepasst werden können.

Beim Kauf einer Textverarbeitung sollten Sie unbedingt mit den zuvor beschriebenen Kriterien vertraut sein. Die Kaufentscheidung wird dann wesentlich vereinfacht.

Finanzbuchhaltung

Als die ersten Computer Ihren Einzug in die Verwaltungsetagen der Betriebe fanden, wurden sie in der Buchhaltung eingesetzt und auch dieser Abteilung allein zugeordnet. Dies ist auch verständlich, denn gerade in der Buchhaltung fallen große Mengen an Daten an, die ohne Computer nur schwer zu verarbeiten sind. Durch den Rechner erhält die Buchführung auch jederzeit einen Überblick über die Vermögens- und Finanzlage des Betriebes und das in wenigen Sekunden.

Wer jedoch die Buchhaltung eines Mittelbetriebes mit dem 128er lösen will, der sollte sich diese Illusion aus dem Kopf schlagen. Die Speicherkapazität des 128ers, sowohl intern, als auch extern auf der Diskette reicht für solche Zwecke nicht aus.

Zur Führung von Büchern ist jeder Unternehmer

verpflichet, wenn er eine bestimmte Vermögens-, Umsatz- oder Gewinngrenze überschreitet. Doch was ist mit denjenigen, die unterhalb dieser Grenze liegen? Hier hat das Einkommensteuergesetz die sogenannte Einnahme/Überschußrechnung vorgesehen. Bei der Überschußrechnung wird der Gewinn durch Vergleich der Betriebsausgaben mit den Betriebseinnahmen ermittelt. Dazu wird ein Kassenbuch geführt, das in der einen Spalte die Ausgaben und in der anderen die Einnahmen vorweist.

Mit dem 128er läßt sich eine Überschußrechnung problemlos durchführen, da hier keine große Datenmengen anfallen. Programme die diese Aufgabe erfüllen, sind bereits im Handel.

Fakturierung

In einem Handels- oder Produktionsbetrieb muß der Wareneingang und der Warenausgang ständig festgehalten werden. Doch nicht nur das, auch Rechnungen müssen erstellt werden. Warum soll man dazu nicht einen 128er einsetzen? Auch zu diesem Zweck werden bereits Programme angeboten.

Grafik

Hier sind nicht die Grafikzeichen der Tastatur gemeint, sondern die hochauflösende Grafik des 128ers mit einer Auflösung von 320 * 200 Bildpunkten. Da jeder dieser 64.000 Bildpunkte gesetzt werden kann, können die schönsten Bilder und Grafiken dargestellt werden. Durch das komfortable BASIC Ihres COMMODORE 128 ist es relativ leicht, eindrucksvolle Grafiken auf den Bildschirm zu zaubern.

Mit den folgenden Befehlen lassen sich Punkte, Linien und ganze geometrische Figuren zeichnen:

```
BOX     - zum Zeichnen eines Rechtecks
CHAR    - zum Schreiben von Text in die Grafik
CIRCLE  - zum Zeichnen von Kreisen, Ellipsen und Bögen
DRAW    - zum Zeichnen einer Linie
PAINT   - zum Ausmalen einer beliebigen Fläche
```

Zu jedem obigen Befehl werden einige Parameter hinzugefügt, um Position, Größe und Farbe der Figuren zu bestimmen. Wir werden hier auf die einzelnen Befehle und ihre Möglichkeiten nicht näher eingehen, da sich damit allein ein ganzes Buch füllen läßt. Um Ihnen dennoch eine Vorstellung von den Möglichkeiten zu geben, ist nachfolgend eine Routine mit Grafikbefehlen abgebildet:

```
40000 COLOR 2,9:COLOR 3,4
40010 GRAPHIC 3,1
40020 BOX 1,5,5,154,80
40030 BOX 1,10,10,149,75
40040 PAINT 3,6,6,1
40050 CHAR 1,11,5," ADRESSVERWALTUNG ",1
40060 CIRCLE 3,43,43,10,,180,360
40070 DRAW 3 TO 116,23
40080 CIRCLE 3,116,43,10,,0,180
40090 DRAW 3 TO 43,63
40100 PAINT 2,11,11,1
40110 Y=-3
40120 FOR X=70 TO 10 STEP -10
40130 Y=Y+6
```

```
40140 CIRCLE 1,80,140,X,Y
40150 NEXT X
40160 SLEEP 5
40170 GRAPHIC 0:RETURN
```

Haben Sie erkannt, daß diese Grafikroutine zur Erweiterung des Adressprogrammes gedacht ist? Laden Sie also erst das Adressprogramm in Ihren COMMODORE 128 und geben dann die obige Routine ein. Ändern Sie dann noch Zeile 1000 wie folgt:

```
1000 GOSUB 40000:GOSUB 100
```

Wenn Sie jetzt das Programm wie gewohnt mit 'RUN' starten, so baut sich ein ansprechendes Titelbild in hochauflösender Grafik auf.
Weiterführende Informationen zur Grafik-Programmierung finden Sie im Handbuch zum COMMODORE 128 und in den Literaturhinweisen im Anhang.

Sound

Der 128er enthält einen kompletten Synthesizer, mit dem er die schönsten Töne und die effektvollsten Geräusche erzeugen kann. Wie bei der Grafik sind auch zur Bedienung des Synthesizers komfortable BASIC-Befehle vorhanden. Da dies aber für einen Einsteiger zu kompliziert ist, werden wir auch die Soundmöglichkeiten des COMMODORE 128 nur kurz vorstellen. Genaueres entnehmen Sie bitte dem Handbuch und weiterführender Literatur (s. Anhang). Doch zunächst einige grundsätzliche Betrachtungen:

Sie wissen nicht was ein Synthesizer ist? Synthesizer im eigentlichen Sinne sind Musikinstrumente, die eine präzise Kontrolle über die elektronische Klangerzeugung ermöglichen. Der Synthesizer unterscheidet sich von anderen Musikinstrumenten dadurch, daß er keinen festgelegten klanglichen Charakter aufweist, wie etwa die Gitarre, die Flöte, oder jedes andere "natürliche" Musikinstrumet. Der musikalische Reiz eines Synhtesizers liegt in der Fähigkeit, sämtliche Klangeigenschaften eines Musikinstrumentes zu imitieren.

Folgende Befehle stellt Ihr COMMODORE 128 zur Bedienung des Synthesizer zur Verfügung:

```
ENVELOPE  - Einstellung der Klangfarbe
FILTER    - Einstellung eines elektronischen Filters
PLAY      - Übermittlung der Partitur
SOUND     - Ausgabe eines einfachen Tones
TEMPO     - Spieltempo
VOL       - Lautstärke
```

Um Ihnen die Leistungsfähigkeit dieser Befehle zu demonstrieren, finden Sie nachfolgend eine Routine die ein Musikstück (Beethoven möge mir verzeihen) nacheinander mit den zehn voreingestellten Instrumenten spielt.

```
10 FOR I=0 TO 9
20 PLAY"T"+RIGHT$(STR$(I),1)
30 VOL9
40 TEMPO 20
50 PLAY"HEQFGGFEDCCDEHEIDHD"
60 PLAY"HEQFGGFEDCCDEHDICHC"
70 PLAY"DQECDIEFQECDIEFQEDCDHO3GO4"
80 PLAY"EQFGGFEDCCDEHDICHC"
90 PLAY"DQECDIEFQECDIEFQEDCDHO3GO4"
100 PLAY"EQFGGFEDCCDEHDICHC"
110 NEXT I
```

Zeile 60 unterscheidet sich von Zeile 50 nur in den letzten 5 Zeichen. Zeile 70 ist mit Zeile 90 und Zeile 80 mit Zeile 100 identisch, wenn Sie daß bei der Eingabe berücksichtigen, so können Sie sich Arbeit ersparen.
Starten Sie die Routine dann mit 'RUN' und genießen das Stück.

Mit dem entsprechenden Programm können Sie Ihren COMMODORE 128 so durchaus zum Bühnenfähigen Synthesizer machen.

Sprites

Wenn Sie schon einmal ein Computerspiel gespielt haben, so war das Männchen, Auto, Flugzeug oder was immer Sie mit dem Joystick auf dem Bildschirm bewegt haben, mit großer Wahrscheinlichkeit ein Sprite. Ein Sprite kann also ganz verschiedene Formen darstellen.
Wie das funktioniert läßt sich am einfachsten in der Praxis zeigen. Geben Sie dazu einmal folgenden Befehl:

```
SPRDEF
```

Ihr COMMODORE 128 zeigt Ihnen daraufhin einen roten Bildschirm mit einem großen Fenster in der linken, oberen Bildschirmecke. Darunter wird die Frage nach der 'SPRITE NUMBER' gestellt. Beantworten Sie diese mit 1.
In dem Fenster erscheint daraufhin ein bedeutungsloses Streifenmuster. Rechts neben dem Fenster, also im roten Bildschirmbereich, erscheint dasselbe Streifenmuster noch einmal verkleinert. Sie haben richtig vermutet, das ist ein Sprite.
Betätigen Sie nun die Tasten 'SHIFT+CLR/HOME'. Das Streifenmuster im Fenster und ebenfalls das Sprite werden gelöscht. Jetzt fällt das '+' Zeichen in der linken oberen Ecke des Bildschirmes auf. Sie können dieses Zeichen mit den Cursor-Steuertasten innerhalb des Fensters bewegen. Versuchen Sie das einmal.
Wenn Sie die Taste '2' betätigen, so erscheint an der Stelle des '+'-Cursors ein dunkles Quadrat. Wenn Sie genau hinsehen, werden Sie im roten Bildschirm an der Stelle des Sprites ebenfalls das verkleinerte Quadrat erkennen.
Fahren Sie mit dem Cursor wieder auf das dunkle Quadrat und betätigen die '1'-Taste. Das Quadrat verschwindet einschließlich der Verkleinerung. Zeichnen Sie auf diese Weise einmal ein Männchen in das Fenster.
Wenn Sie fertig sind, betätigen Sie die Tasten 'SHIFT+RETURN'. Das verkleinerte Männchen verschwindet und Sie können wieder eine Sprite Nummer eingeben. Betätigen Sie die 'RETURN'-Taste ohne eine Nummer eingegeben zu haben. Sie befinden sich damit wieder im Textbildschirm.

Was haben wir jetzt genau gemacht? Zuerst wurde durch den Befehl 'SPRDEF' der Sprite-Editor aufgerufen. Dann haben wir festgelegt, daß das Sprite mit der Nummer 1 zu editieren ist. Das alte Sprite 1 wurde mit 'CLR' gelöscht und ein neues Sprite 1 (Männchen) gezeichnet. Daraufhin haben wir den Sprite-Editor wieder verlassen. Geben Sie jetzt einmal folgende Befehle:

```
MOVSPR 1,100,100
SPRITE 1,1,2
```

Ihr Männchen erscheint in der linken Bildschirmhälfte. Der Befehl 'MOVSPR' hat das Sprite positioniert und der 'SPRITE'-Befehl das Sprite eingeschaltet und die Farbe gesetzt. Diese beiden Befehle können noch wesentlich mehr, wenn sie mit den entsprechenden Parametern versorgt werden. Im Rahmen dieses Buches können wir auch hier nicht mehr als einen ersten Eindruck des Möglichen vermitteln. Folgende Befehle sind zur Spritesteuerung und Gestaltung einsetzbar:

```
COLLISION  - Stellt Kollisionen von Sprites fest
GSHAPE     - Sprite aus Zeichenkette in Grafik
MOVSPR     - Sprite bewegen
RSPCOLOR   - Spritefarben holen
RSPOS      - Spriteposition und Geschwindigkeit holen
RSPRITE    - Spriteparameter holen
SPRCOLOR   - Spritefarben
SPRDEF     - Sprite Editor einschalten
SPRSAV     - Sprite in Zeichenkette oder umgekehrt
SSHAPE     - Grafik aus Bildschirm in Zeichenkette
```

Zum Abschluß dieses Kapitels sollen Sie noch Gelegenheit bekommen ein wenig mit Ihrem Männchen-Sprite zu spielen. Geben Sie dazu das folgende Programm ein:

```
10 SPRITE 1,1,2
20 MOVSPR 1,100,100
30 A=JOY(2)
40 IF A=0 THEN 30
50 IF A>127 THEN A=A-128:SPRITE 1,1,3,1,1,1
60 IF A=1 THEN MOVSPR 1,+0,-5
70 IF A=2 THEN MOVSPR 1,+5,-5
80 IF A=3 THEN MOVSPR 1,+5,+0
90 IF A=4 THEN MOVSPR 1,+5,+5
100 IF A=5 THEN MOVSPR 1,+0,+5
110 IF A=6 THEN MOVSPR 1,-5,+5
120 IF A=7 THEN MOVSPR 1,-5,+0
130 IF A=8 THEN MOVSPR 1,-5,-5
140 SPRITE 1,1,2,0,0,0
150 PRINT:GOTO 30
```

Erkennen Sie das Listing wieder? Richtig, es handelt sich um das Demoprogramm aus dem Kapitel über den Joystick. Es wurden nur die 'PRINT' Anweisungen durch entsprechende Befehle zur Bewegung des Sprites ersetzt. Natürlich können Sie das Joystickprogramm einladen und entsprechend abändern.
Stecken Sie dann Ihren Joystick in Port 2 und starten das Programm mit 'RUN'.

ANHANG

Adressenverwaltung auf Kassette

```
5000 REM ====================
5010 REM   DATEI LADEN
5020 REM ====================
5030 GOSUB 300:PRINT:PRINT
5040 PRINT"BITTE DATENKASSETTE EINLEGEN UND ZU-"
5050 PRINT
5060 PRINT"RUECKSPULEN!"
5065 PRINT
5070 PRINT"DRUECKEN SIE DANACH RETURN";
5080 INPUT X$
5085 PRINT
5090 OPEN 1,1,0,"ADRESSEN"
5100 INPUT#1,Z
5110 FOR Y=1 TO Z
5120 FOR I=1 TO 7
5130 INPUT#1,D$(Y,I)
5140 NEXT I
5150 NEXT Y
5160 CLOSE 1
5170 PRINT:PRINT"DATEN SIND GELADEN!"
5180 SLEEP 2
5190 GOTO 1000

10000 REM ====================
10020 REM   DATEI SPEICHERN
10030 REM ====================
10040 GOSUB 300:PRINT
10045 GOSUB 500:IF Z=0 THEN 1000
10060 PRINT"BITTE DATENKASSETTE EINLEGEN UND ZU-"
10070 PRINT
10080 PRINT"RUECKSPULEN!"
10085 PRINT
10090 PRINT"DRUECKEN SIE DANACH RETURN";
10095 INPUT X$
10100 OPEN1,1,1,"ADRESSEN"
```

```
10110 PRINT#1,Z
10120 FOR Y=1 TO Z
10130 FOR I=1 TO 7
10140 PRINT#1,D$(Y,I)
10150 NEXT I
10160 NEXT Y
10165 CLOSE 1
10170 PRINT:PRINT"DATEN SIND GESICHERT!"
10180 SLEEP 2
10190 GOTO 1000
```

Stichwortregister

S

T

U

V

Z

Bücher zu COMMODORE 128

Ein Buch, das nicht nur absolut wichtig ist für jeden 64er-Besitzer, der die phantastischen Möglichkeiten des Nachfolgemodells kennenlernen will, sondern auch dem Kaufinteressierten Entscheidungshilfen bietet. Informieren Sie sich sachkundig über die wirklich herausragenden Leistungsmerkmale des C128: 64-aufwärts-kompatibel, 3 Betriebssysteme (eins davon CP/M), 128 KB RAM u.v.m.

Gerits/Kampow
Das Premierenbuch
Der neue Commodore 128
220 Seiten, DM 39,–
ISBN 3-89011-062-2

Ein Muß für jeden, der sich intensiver mit dem C-128 beschäftigt. Einführung in das System, Hardware- und Interfacebeschreibung, Erläuterung des VIC-Chips, des VDC, SID, detailliert und leichtverständliche Beschreibung der Memory-Management-Unit (MMU), ein sehr ausführlich kommentiertes ROM-Listing, Einführung: wie arbeite ich mit ROM-Listing und Zeropage, mit sehr vielen Programmbeispielen!

Gerits/Schieb/Thrun
128 INTERN
507 Seiten, DM 69,–
ISBN 3-89011-098-3

Eine Fundgrube für alle C-128 Besitzer! Ob man einen eigenen Zeichensatz erstellen, die doppelte Rechengeschwindigkeit im 64er Modus benutzen oder die vorhandenen ROM-Routinen verwenden will. Dieses Buch ist randvoll mit wichtigen Informationen; z.B.: Bank-Switching/ Speicherkonfiguration, Registererläuterungen zum Video-Controller und 640 x 200 Punkte Auflösung. Dieses Buch darf bei keinem 128er fehlen!

Hornig/Weltner/Trapp
128 TIPS & TRICKS
327 Seiten, DM 49,–
ISBN 3-89011-097-5

Bücher zu COMMODORE 128

Jetzt gibt es das große Floppybuch auch zur 1570/1571! Mit einer Einführung für Einsteiger, Arbeiten mit dem C-128 und BASIC 7.0, einer umfassenden Einführung in das Arbeiten mit sequentiellen und relativen Dateien, Programmierung für Fortgeschrittene: Nutzung der Direktzugriffsbefehle, Programme im DOS, wichtige DOS-Routinen und ihre Anwendung und natürlich ein ausführlich dokumentiertes DOS-Listing.

Ellinger
Das große Floppybuch zur 1570/1571
ca. 300 Seiten, DM 49,–
ISBN 3-89011-124-6

Erscheint ca. November

Falls Sie auf dem Commodore 128 das CP/M einsetzen wollen, sollten Sie dieses Buch lesen! Von grundsätzlichen Erklärungen zur Speicherung von Zahlen, Schreibschutz oder ASCII, Schnittstellen und Anwendung von CP/M-Hilfsprogrammen. Für Fortgeschrittene: CP/M und Commodore-Format, Erstellen von Submit-Dateien u.v.m. Nutzen Sie die vollen Möglichkeiten des Standard-Betriebssystems CP/M!

Weiler/Schieb
Das CP/M-Buch zum C-128
ca. 250 Seiten, DM 49,–
ISBN 3-89011-116-5

Erscheint ca. November

DAS STEHT DRIN:

128 für Einsteiger sollte Ihr erstes Buch zum Commodore 128 sein. Es bietet eine leichtverständliche Einführung in Handhabung, Einsatz und Programmierung Ihres neuen Rechners und setzt keinerlei Vorkenntnisse voraus.

Aus dem Inhalt:

- Die Geräte nach dem Auspacken
- Die Bedienung der Tastatur und des Editors
- Fertigprogramme laden und starten
- Der erste Befehl
- Das erste Programm
- BASIC-Einführung Schritt für Schritt
- Programmieren einer Adressenverwaltung
- Bedienen der Peripheriegeräte
- Anwendungsbeispiele
- Textverarbeitung
- Dateiverwaltung
- Finanzbuchhaltung
- Fakturierung
- Grafik
- Sound
- Sprites

UND GESCHRIEBEN HABEN DIESES BUCH:

Norbert Szczepanowski ist Datenverarbeitungs-Kaufmann bei DATA BECKER. Seine fundierten Kenntnisse in der Programmierung verschiedener Rechner haben ihn schnell zum Bestsellerautor werden lassen. Heribert Schmidt ist Redakteur bei der DATA WELT und hat sein Können schon mit den beiden Trainingsbüchern zu DATAMAT und ASSEMBLER unter Beweis gestellt.

ISBN 3-89011-099-1