Miedel

Schneider CPC: Dateiverwaltung

Eine Software-Sammlung

Franzis'

Miedel
Schneider CPC: Dateiverwaltung

In der Reihe
Franzis Computer-Bibliothek
sind erschienen:

Busch, Basic: Dateien, Listen und Verzeichnisse
Busch, Basic: Matrix-Operationen
Busch, Basic: Sortierprogramme
Requardt, Basic: Alles über Peek und Poke
Busch, Basic: Mathematik per Computer
Busch, Basic: Die perfekte Behandlung von Zeichenketten
Busch, Basic: Zahlen-Umwandlungen
Vogelsang, Basic: Betriebsprogramme für den Funkamateur
Nolte, Basic: HF-Rechenprogramme
Busch, Basic: Programme für Kaufleute
Busch, Basic: Wir machen Musik
Busch, Basic: Dateien auf Disketten verwalten
Busch, Basic: Der wahlfreie Zugriff auf Dateien
Smode, MS-DOS für Insider
Busch, Basic: Grafik mit Homecomputern

Franzis Computer-Bibliothek

Lothar Miedel

Schneider CPC: Dateiverwaltung

Eine Software-Sammlung

Mit 11 Abbildungen

Franzis'

Franzis Computer-Bibliothek, Band 16

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Miedel, Lothar:
Schneider CPC: Dateiverwaltung : e. Software-Sammlung / Lothar Miedel. — München : Franzis, 1987.
(Franzis-Computer-Bibliothek ; Bd. 16)
ISBN 3-7723-8691-1

NE: GT

Druck: Wiener Verlag, A-2325 Himberg
Printed in Austria · Imprimé en Autriche

ISB N 3-7723-8691-1

Vorwort

Mit diesem Buch haben Sie kein Dateiverwaltungsprogramm erworben, obwohl mehrere Programme dieser Art enthalten sind. Dieses Buch soll, trotzdem es sehr leistungsstarke Programme enthält, vielmehr Anregungen und Tips geben, wie Dateiverwaltungsprogramme aufgebaut werden können. Es ist ein sehr praxisbezogenes Anleitungsbuch für die Erstellung eigener Dateiverwaltungsprogramme.

Dabei habe ich größtenteils auf sonst übliche "Dateiverarbeiter" verzichtet. Diese kleinen Programme dienen mehr der Einführung in die Dateiverwaltung, als echten Aufgaben. Falls aber ganz schnell mal ein simples Datenverarbeitungsprogramm benötigt wird, sind auch diese Beispiele direkt einzusetzen.

Auch auf den BANKMAN, dem Hilfsprogramm zur Bank-Bedienung beim CPC 6128, habe ich vollkommen verzichtet, denn es geht viel einfacher und dazu noch mit mehr Speicherplatz für Daten!

Ich habe versucht, alle wichtigen Programme so zu beschreiben und zu dokumentieren, daß sie sowohl die Programme verstehen, als auch an Ihre Wünsche anpassen können. Außerdem ist dieses Buch so gegliedert, daß Sie dann, wenn Sie die Programme nur benutzen wollen, Sie größtenteils nur im Abschnitt über die Bedienung des jeweiligen Programmes nachschlagen müssen.

Wenn Sie aber die Arbeitsweise der Programme verstehen wollen (wer will das nicht?), erhalten Sie außerdem ein Wissen, das über das Handbuch-Niveau weit hinausgeht und, was mir ebenfalls wichtig erscheint, viele kleine Maschinensprache-Routinen, die Sie auch bei anderen Programmen sinnvoll einsetzen können.

Sie erfahren also nicht nur viele Dinge über die Dateiverwaltung, sondern erhalten auch sehr viele Tips,

Kniffe und Tricks für alle CPCs von Schneider.

Ein gutgemeinter Ratschlag gleich zu Beginn und für Beginner.

Man lernt durch Abtippen von Programmen (durch die dann auftretenden Fehler) zwar sehr viel, aber bevor Sie tage- oder wochenlang nach Fehler suchen, sollten Sie sich wirklich überlegen ob es nicht sinnvoller ist, die Diskette mit allen Programmen beim Franzis-Software-Service zu bestellen.

München Lothar Miedel

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwekke bestimmt und dürfen nicht gewerblich genutzt werden*).
Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag sieht sich deshalb gezwungen, darauf hinzuweisen, daß er weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann. Für die Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

*) Bei gewerblicher Nutzung ist vorher die Genehmigung des möglichen Lizenzinhabers einzuholen.

Inhaltsverzeichnis

1 Einführung in die Datenverarbeitung

Vorbemerkungen zur Datenverwaltung

Wer kennt ihn nicht, den guten alten Karteikasten, in dem sich - nach irgendwelchen Kriterien geordnet - viele Karteikarten befinden?

Auf diesen Karten wiederum sind Daten, also Informationen aufgeschrieben, die von Wichtigkeit sind. Je nachdem, um welche Daten es sich handelt, stellt der gesamte Karteikasten eine Lager-, Kunden-, Umsatz-, Adressen- oder auch eine andere spezielle Datei dar.

Während Programme zur Lagerverwaltung oder zur Verwaltung von Kunden-Dateien bereits "genau zugeschnittene" Programme sind, ist eine allgemeine Dateiverwaltung noch sehr universell. Erst die Art der Daten und auch deren Verarbeitung bestimmt, um welche "Verwaltung" es sich handelt.

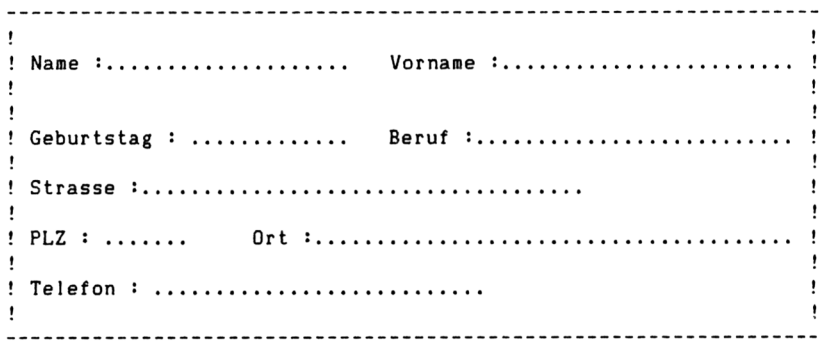
Das bedeutet, ein allgemeines Dateiverwaltungsprogramm kann zwar beispielsweise sowohl zur Adressen-, als auch zur Lagerverwaltung eingesetzt werden, ein Programm zur Lagerverwaltung kann aber höchstwahrscheinlich nur schlecht zur Adreßverwaltung dienen; dazu ist es zu speziell.

In vielen Fällen ist die Gestaltung der Karteikarte relativ frei. Dies gilt auch für die elektronische Dateiverwaltung. Auch bei dieser kann - wenn das Programm es zuläßt - die Bildschirmdarstellung oder auch die Ausdrucksform durch den Anwender bestimmt werden. Bei verschiedenen Programmen aber ist es nicht möglich. Hierzu wären dann Programmveränderungen nötig.

Ein allgemeines Dateiverwaltungsprogramm kann zu einer speziellen Dateiverwaltung werden, wenn bestimmte Kriterien starr festgelegt werden. Welche Punkte dies sein können, soll später betrachtet werden. Vorher

noch einmal zurück zu ganz allgemeinen Fakten bezüglich einer Dateiverwaltung.

Um nun einige wichtige Punkte bezüglich einer Datei zu klären, werfen wir einen Blick in eine gedachte Kartei und greifen uns wahllos irgendeine Karteikarte heraus. Zur besseren Veranschaulichung nehmen wir dabei eine Karte des guten alten Karteikastens:



Wie Sie sehen ist die Karteikarte leer, das heißt es sind keinerlei Einträge vorgenommen worden, also keine Daten vorhanden. Üblicherweise befinden sich in einer Kartei oder Datei aber ausgefüllte Karten. Es sind Informationen enthalten, die für bestimmte Zwecke benötigt werden.

Obwohl die obige Kartenskizze leer ist, können Sie genau erkennen, daß es sich um eine Karteikarte handelt. Dies liegt an den Randinformationen, die Sie unbewußt verarbeitet haben. Aus Ihrer Erfahrung heraus wissen Sie, daß ein "Informationsblatt" mit Angaben, wie sie oben gezeigt sind, zu einer Datei gehört.

Die Eintragsfelder auf der Karteikarte beziehen sich immer auf bestimmte Informationen. Im obigen Beispiel ist ein Platz für den Namen, ein anderer für den Vornamen usw. reserviert. Das ist wichtig, damit die Informationen auch immer richtig zugeordnet werden. Diese Karten haben eine "Eintragsschablone"; eine Maske. Bei käuflichen Karten ist diese sehr oft aufgedruckt.

Bei kleinen privaten Dateien werden oft Karten ohne Maske verwendet, die Eintragungen aber trotzdem immer in etwa an den gleichen Positionen plaziert. Dabei kann es sehr leicht vorkommen, daß wichtige Informationen vergessen werden, weil die Maske fehlt. Trägt man nach keinem bestimmten Schema ein, dann kann es ebenfalls leicht vorkommen, daß Einträge vergessen werden. Auch die Übersichtlichkeit der gesamten Datei leidet sehr.

Damit in einer Datei eine bestimmte Karte mit den gewünschten Informationen schnell gefunden werden kann, werden die Karten im Karteikasten nach irgendwelchen Sortierkriterien geordnet abgelegt. Bei einer normalen Kartei wird dies meist in alphabetischer Reihenfolge sein.

Bei Kundendateien von Firmen erfolgt aber zur schnelleren Unterscheidung von Kunden oft auch die Vergabe von Kunden-Nummern und die Sortierung nach diesen. Vor allem bei Großfirmen, mit vielen Abteilungen und Geschäftssitzen an verschiedenen Orten, bieten die Kunden-Nummern enorme Vorteile zur Unterscheidung.

Sollen nun bestimmte Daten gesucht werden, so ist in jedem Falle nach dem "Namen" der bestimmten Karte zu suchen. Das bedeutet für den Suchenden, daß er dann, wenn er die Datei und deren Struktur noch nicht kennt, bei der ersten Karte beginnt und dann mit den folgenden Karten weitermacht, bis er die entsprechende Karte gefunden hat.

Sind aber nähere Details über die "Ordnungsstruktur" bekannt, dann ist in den meisten Fällen die Suche wesentlich schneller abgeschlossen.

Bei alphabetisch geordneten Dateien werden auf die Karteikarten Reiter aufgesetzt oder auch Trennkarten mit einsortiert, die den Beginn einer neuen Gruppe von Karten anzeigen. Das "herausragende" Merkmal dieser Trenner ist ein Buchstabe des Alphabetes. Diese Unterteilung erleichtert sowohl die Suche, als auch das Einordnen.

Bei umfangreichen Dateien sind eventuell weitere Unterteilungen in einer Buchstabengruppe erforderlich. So werden dann beispielsweise für den Buchstaben "S" die Untergruppen "S", "SCH" und "ST" geschaffen. Auch

diese Unterteilungen helfen beim - um es einmal sehr modern auszudrücken - "handling" der Datei. Also bei eigentlich allen Arbeiten, die mit der Datei durchzuführen sind.

Wehe einem Suchenden aber, wenn eine Karteikarte nicht an "Ort und Stelle" ist. Wenn sie sich also nicht an dem Platz befindet, an dem sie eigentlich sein sollte! Diese Karte wird nur noch durch Zufall oder durch intensive Suche danach gefunden. Ordnung zu halten ist deshalb unbedingt geboten.

Ein Nachteil bei einer Datei dieser Art ist aber nicht zu übersehen: Ein schneller Zugriff kann nur über das Einordnungskriterium erfolgen.

Falls dieses (wie im Kartenbeispiel) der Name ist, dann ist die Karte oder sind die Karten mit den zutreffenden Suchkriterien schnell gefunden. Falls aber alle Karten bei denen der Vorname oder die Postleitzahl übereinstimmt, herausgesucht werden sollen, dann müssen die gesamten Karten der Datei gesichtet werden.

Diesen Nachteil erkannte auch Herman Hollerith und fand eine Möglichkeit, diesen zu umgehen. Mit dem von ihm entwickelten Selektionsverfahren wurde im Jahre 1890 die Volkszählung in den USA durchgeführt und deren Auswertung wesentlich erleichtert.

Das Prinzip war ebenso genial, wie einfach. Stellen Sie sich bitte Karteikarten vor, die in einer ganz bestimmten Anordnungsweise - jede an den gleichen Stellen - Löcher haben. Diese Löcher müssen so angeordnet sein, daß man mit einem "Stift" durch alle Karten hindurchfahren kann.

Wenn nun die Karten zusätzlich noch so präpariert sind, daß immer dann, wenn ein Suchkriterium nicht zutrifft vom Loch zum Kartenrand eingeschlitzt ist, dann bleiben am Stift immer nur die Karten hängen, für die das gesuchte Kriterium stimmt. Die "Trefferkarten" können deshalb dann mittels des durchgehenden Stiftes aus der Gesamtkartei herausgehoben werden.

Diese simple Idee von Herman Hollerith revolutionierte die damaligen Dateiverwaltungen und Karteikästen.

Heute im Zeitalter der elektronischen Datenverwaltungen ist diese Methode schon fast wieder in Vergessenheit geraten. Computer erlauben bei ausreichendem Speicherplatz und bei hoher Arbeitsgeschwindigkeit ein schnelles Sortieren von Daten nach allen möglichen Sortierkriterien und das auch noch mit mehreren gewünschten Sortiertiefen. Ein Sortieren also, bei dem nicht nur nach einem, sondern gleich nach mehreren Einträgen sortiert wird.

Ja, es kann nicht nur einfach sortiert werden, sondern die Sortierkriterien können auch noch - wenn es sein soll - berechnet werden, wie es beispielsweise für Umsatzstatistiken oder für die sogenannte ABC-Analyse erforderlich ist.

Es war aber ein weiter Weg von den Karteikästen mit ihren Karten bis zu Dateiverwaltungen mit Computern. Erst die Entwicklung des elektronischen Schaltkreises durch Jack Kilby (Texas Instruments) ermöglichte es, Computer so zu bauen, daß sie vernünftige Ausmaße hatten und nicht ganze Büroetagen belegten.

Aber das war noch nicht alles, was noch geschehen mußte, um elektronische Dateiverwaltungen für jedermann zu ermöglichen. Erst durch die Fort- und Weiterentwicklungen der integrierten Schaltkreise, die Entwicklung der Mikroprozessoren usw., wurden elektronische Dateiverarbeitungs-Systeme sinnvoll aber anfangs leider nur für große Firmen preislich akzeptabel.

Blieb in der Anfangszeit die elektronische Datenverarbeitung (EDV) auch nur den Firmen vorbehalten, die sich die immer noch riesigen "Computer-Dinosauriere" leisten konnten, so ist heute die elektronische Datenverarbeitung fast überall im geschäftlichen wie im privaten Bereich anzutreffen.

Mit beigetragen zu dieser hohen Verbreitung der sogenannten Personalcomputer hat unter anderem auch Chuck Peddle, der den mittlerweise schon legendären PET (Personal Electronic Transacter) von Commodore entwickelte. Aber auch der Apple II von Steve Jobs und Stephen Wozniak soll nicht unerwähnt bleiben. Dieser Computer trug mit dazu bei, daß der weltweite Boom der Personalcomputer begann.

Nicht vergessen werden soll auch der C-64 von Commodore, der durch ein gewaltiges Angebot von Spielesoftware vor allem auch in Deutschland ein Computerfieber auslöste. Da aber nicht nur immer gespielt wird, wurde auch dieser Computer anderen Aufgaben zugeführt. Viele davon waren Dateiverwaltungsaufgaben.

Es ist auch kein Wunder, daß die Schneider CPCs in vielen Fällen Dateiverwaltung betreiben dürfen. Sie besitzen einen weitaus schnelleren und besseren Basic-Interpreter und eine schnellere Diskettenstation.

Aber nicht nur die Hardware-Voraussetzungen mußten geschaffen werden, sondern auch die Firm- und Software. Sie sorgt dafür, daß alle Komponenten richtig zusammenspielen und die Elektronik auch das tut, was man eigentlich will.

Die von Bill Gates gegründete Software-Firma Microsoft lieferte den ersten Basic-Interpreter, der es auch einem Laien ermöglichte mit einem Homecomputer umzugehen.

Viele Dinge liefen also parallel und so blieb es auch nicht aus, daß verschiedene Lösungswege für Dateiverwaltungsprobleme beschritten wurden. Dies betrifft sowohl die Hard- als auch die Software. So gibt es heute viele Computer und natürlich auch viele unterschiedliche Computertypen, mit denen Dateiverwaltungsaufgaben erledigt werden und es gibt noch viel mehr Programme hierzu.

Aber ein Hauptproblem ist bei der ganzen Vielfalt doch noch vorhanden:

> Jeder hat seine eigenen Vorstellungen von dem was eine Dateiverwaltung können soll und muß, denn in vielen Details liegen die Interessen doch sehr unterschiedlich.

Zwar gibt es viele "Quasi"-Standard-Programme für die verschiedensten Computer, aber gerade diese Vielfalt führt dann zu weiteren Problemen und Schwierigkeiten.

Oft ist ein einfacher Datenaustausch zwischen zwei verschiedenen Computern nur deshalb nicht möglich, weil die Speichermedien des einen, vom anderen gelesen werden können (Unterschiedliche Diskettenformate!).

Zwar können Daten über entsprechende Schnittstellen (z.B. RS 232) ausgetauscht werden, dies ist aber oft auch nur dann möglich, wenn im Programm eine entsprechende "Ausgabe" und "Eingabe" vorgesehen ist.

Besonders schlimm ist es aber, wenn auf ein und demselben Computer die Daten nicht zwischen verschiedenen Programmen ausgetauscht werden können, weil die Datenstruktur des "Erzeugerprogrammes" völlig anders ist, als es ein Programm zur weiteren Bearbeitung benötigt. Vor allem bei käuflichen Programmen kann man sehr leicht auf dieses Problem stoßen, wenn man verschiedene Programme von unterschiedlichen Herstellern einsetzen will, die auf die gleichen Datensätze zugreifen sollen.

Oft kann man sich dann nur sehr schwer weiterhelfen, weil die Informationen über die Struktur der Daten nicht erhältlich sind. Hier kann nur versucht werden, die Datenstruktur zu analysieren um sie dann mit kleinen Hilfsprogrammen umzubauen.

Diese zuletzt geschilderten Schwierigkeiten sind zwar für große Firmen oder für Computerbesitzer mit großem Geldbeutel kein Problem, aber für die anderen?

Deshalb ist es nicht nur sinnvoll, dann mehr über die elektronische Dateiverwaltung zu wissen, wenn es bereits "brennt", sondern frühzeitig. Dann sind alle Probleme nur halb so groß.

Wie Dateien aufgebaut sein können, wie Dateiverwaltungsprogramme arbeiten, wie sie selbst geschrieben werden können, welche Überlegungen gemacht werden sollen, und vor allem über die Dateiverwaltung selbst, soll Ihnen dieses Buch Antworten geben.

Es beginnt mit sehr einfachen Programmen und steigert sich dann von Abschnitt zu Abschnitt. Die vielen Beispielprogramme sollen zeigen, wie Lösungen mit den Schneider CPCs realisiert werden können.

2 Grundlagen zur elektronischen Dateiverwaltung

Die Grundbegriffe

Damit Sie sich etwas leichter tun, die speziellen Begriffe der elektronischen Datenverwaltung zu verstehen, folgt hier eine kurze Gegenüberstellung von Begriffen der elektronischen Datenverarbeitung zu solchen der Karteikästen.

Auch weitere Erklärungen sollen helfen, Dinge die im Zusammenhang mit der Verwaltung von Daten stehen, zu erläutern. Denn nicht immer ist klar, was mit Record, mit sequentiell, mit random, mit Datensatz usw. gemeint ist.

Datei/Kartei - hier handelt es sich im Regelfall um den kompletten Karteikasten, also um die Gesamtheit aller gespeicherten bzw. erforderlichen Daten.

Datensatz - auch Eintrag oder Record genannt. Er entspricht einer Karteikarte, also zusammengehörigen Daten. Ein Datensatz kann ein festgelegtes Schema (Struktur) haben. Die Unterteilung erfolgt in sogenannte Felder.

DOS - Disk Operating System. Das Betriebssystem für die Diskettenstation. In dieser Firmware liegt fest, wie die Zugriffe auf die Diskette erfolgen können.
Das Disketten-Betriebs-System ist

für den Daten- und Befehlsaustausch zwischen dem Computer und dem Diskettenlaufwerk zuständig. Es erledigt also die Steuerung der Diskettenstation, sowie die Verwaltung auf den Disketten selbst. Komfortable Versionen erlauben dabei auch relative Dateien aufzubauen oder bieten die Möglichkeit des Direkt-Zugriffes auf einzelne Sektoren.

Feld - oder auch Datenfeld; ein einzelner Eintrag auf einer Karteikarte.

Random-Datei - auch als relative Datei bezeichnet. Im ursprünglichen Sinne aber als Datei mit wahlfreiem Zugriff bezeichnet. Ähnlich wie relative Datei, aber keine Verweisdateien vorhanden. Die Daten liegen nur nach der Eintragsnummer geordnet vor. Die Stelle zum Auffinden des Eintrages wird berechnet.

relative Datei - bei dieser kann ein Datensatz direkt geholt werden, ohne andere lesen zu müssen. Allerdings muß hierzu bekannt sein, wo er zu finden ist. Diese Information kann beispielsweise in einer Kundennummer stecken. Manchmal gibt es auch sogenannte Verweisdateien. Bei den Datensätzen ist eine festgelegte Struktur zwingend. Der Name beispielsweise muß sich bei jedem Eintrag an der gleichen Position oder Stelle befinden. Der Feldaufbau liegt also fest.

sequentielle Datei - diese ist dadurch gekennzeichnet, daß die Daten nur nacheinander gelesen und geschrieben werden können. Wenn 1ØØ Datensätze vorhanden sind dann müssen evtl. 99 "überlesen" werden, bis der gewünschte Satz erreicht ist und bearbeitet werden kann. Dies entspricht also in etwa dem Durchblättern und Lesen der gesamten Datei, obwohl nur Informationen über den letzten Datensatz gewünscht werden!

Verweisdatei - eine Hilfsdatei, die beispielsweise alphabetisch geordnet ist und die Information beinhaltet, wo die Einträge, Datensätze etc. stehen.

Zugriffsart - man unterscheidet zwischen direktem und seriellem Zugriff. Die Zugriffsart ist vor allem vom Speichermedium abhängig. RAM-Speicher ermöglichen den direkten Zugriff, Kassetten-Rekorder den seriellen. Welche Zugriffsart bei einer Diskettenstation möglich ist, bestimmt das DOS.

Egal, ob Sie nun einen Karteikasten oder eine elektronische Dateiverwaltung sinnvoll einsetzen wollen, immer gilt:

Alle Datensätze müssen laufend aktualisiert werden, denn mit ungültigen Daten ist kaum etwas anzufangen, bzw. die Auswertungen sind für die "Katz"!

In diesem Zusammenhang auch gleich noch ein wirklich sehr, sehr wichtiger Ratschlag an alle, die elektronische Dateiverarbeitung betreiben:

BACKUP....BACKUP....BACKUP

Speichern Sie so oft wie nur (sinnvoll) möglich die Daten ab und verfahren Sie am besten nach dem Prinzip:

Großvater, Vater, Sohn!

Denn, wenn die Sohn-Diskettendaten nicht mehr gelesen werden können, dann haben Sie trotzdem noch zwei Reserven, die unter Umständen dann zwar nicht dem aktuellsten Datenstand entsprechen, aber es müssen nicht alle Daten neu eingegeben werden!

Bei 737280 Datenbytes, wie sie bei dem in diesem Buch enthaltenen Programm RANDOMD2 möglich sind, wäre das eine Menge Arbeit!

3 Einfache Dateiverwaltungs-Programme

3.1 Überlegungen für eigene Dateiverwaltungsprogramme

Wer wegen seiner speziellen Wünsche oder aufgrund von Unzulänglichkeiten verfügbarer Dateiverwaltungs-Programme die Entscheidung gefällt hat, für seinen Schneider CPC ein eigenes Programm zur Verwaltung von Daten zu schreiben, wird dies in den meisten Fällen mittels der Programmiersprache Basic tun. Nur dort, wo schnelle Abläufe erforderlich sind, oder auch dann, wenn das Basic keine Möglichkeiten bietet, wird auf die Ebene der Maschinensprache ausgewichen. Prinzipiell gilt dies auch für die Programme dieses Buches. Allerdings gab es noch einen weiteren Grund für die Maschinenprogramme dieses Buches: Komfort!

Auf Maschinensprache-Ebene sind aber auch nur dann erhebliche Geschwindigkeitssteigerungen möglich, wenn man die Eigen- und Besonderheiten der CPCs berücksichtigt und sie ausnützt. Die CPCs haben nämlich einen sehr guten und schnellen Basic-Interpreter eingebaut, der einige Interpreter anderer Computer weit hinter sich läßt.

Bevor aber mit den eigentlichen Programmierarbeiten begonnen wird, sollte sich jeder erst einmal Gedanken über seine zu lösenden Probleme machen! Einfach nur so "drauflosprogrammieren" hat keinen Sinn und führt außerdem in den meisten Fällen zu so chaotischen Programmen, daß spätere Verbesserungen und Erweiterungen fast unmöglich, bzw. nur unter hohem Zeitaufwand durchführbar sind.

Vor Eingabe der ersten Programmzeile sollte eine möglichst genaue Analyse des zu lösenden Problems erfolgen. Aus dieser ergibt sich dann auch, welche Voraussetzungen gegeben sein müssen. Beispielsweise auch, welche Hardware erforderlich ist:

Reicht der im CPC 464 eingebaute Kassettenrekorder? Ist eine Diskettenstation mit ca. 180 KByte ausreichend oder muß ein Laufwerk mit 720 KByte oder gar eine Harddisc eingesetzt werden, usw. ... usw.

Sind all diese Überlegungen abgeschlossen, dann sollte ein Programm-Ablaufschema erstellt werden, damit sinnvoll und übersichtlich programmiert werden kann.

An den folgenden kleinen Beispielen will ich aufzeigen, warum diese Überlegungen so wichtig sind.

Vor allem sind sie auch wichtig, wenn bei einem Programm größeren Umfanges nachträglich noch ergänzt werden muß. Sonst verliert man leicht den Überblick, vor allem wenn die Zeit des Erstellens schon etwas zurückliegt.

Aus einem ursprünglich vielleicht sehr übersichtlichen Programm wird dann im Laufe der Zeit ein "Programm-Verhau", der von Ergänzung zu Ergänzung immer schlimmer wird. Und das Schlimmste: Irgendwann verliert man den Überblick total und fängt wieder von vorne an!

3.2 Beispiel einer ersten kleinen Datei

Eine kleine Telefondatei soll programmiert werden. Es ist keine Ausgabe über einen Drucker gewünscht. Das Programm soll bei Eingabe eines Familiennamens die zugehörige Telefon-Nummer ausgeben. Aufzunehmen sind etwa 20 Einträge, es ist aber zu erwarten, daß die Anzahl höher werden kann.

Weitere Ansprüche bestehen für dieses erste Programm nicht!

Eine Datei mit einem derartig geringen Datenumfang kann inklusive Programm direkt im Speicher der CPCs gehalten werden.

Die Daten werden zweckmäßigerweise als sogenannte Datastatements abgelegt. Dadurch können - falls erforderlich - sehr schnell Ergänzungen und Änderungen erfolgen. Außerdem hat es den Vorteil, daß das Gesamtprogramm (mit Daten) nur aus einem einzigen Teil besteht. Es muß also nichts nachgeladen werden. Auch Er-

weiterungen, die nur den Programmablauf betreffen, sind schnell erledigt.

Als Daten sollen

der Familienname des Teilnehmers
der Vorname und
die Telefonnummer

aufgenommen werden.

Ohne nun bereits weiteres über das Programm selbst festzulegen, ist dadurch die Reihenfolge der Ablage in den Datazeilen definiert, nämlich in der Reihenfolge wie sie oben stehen. Das dürfte die sinnvollste sein. Eine entsprechende Datazeile könnte also beispielsweise folgendermaßen aussehen:

```
10000 DATA Meier,Peter,0711/356879
```

Alle weiteren Einträge sollten sinnvollerweise eine eigene Zeile erhalten, denn dann sind Änderungen sehr schnell zu machen und auch eine gewisse Übersichtlichkeit ist gegeben. Eine derartige Liste von Einträgen könnte nun beispielsweise so aussehen:

```
10000 DATA Meier,Peter,0711/356879
10010 DATA Huber,Michael,089/7765776
10020 DATA Gernhuber,Emma,0617/663421
10030 DATA Zeidler,Lothar,08122/344
10040 DATA Brack,Sieglinde,0211/8890943
10050 DATA Strobl,Hans,05344/7656
10060 DATA Faber,Richard,03221/8973
10070 DATA Rinderle,Heinrich,089/443434
10080 DATA Axtner,Wilhelm,0433/9824376
10090 DATA Dutschke,Franz,0211/49823
10100 DATA Schmid,Ernst,0422/3216
10110 DATA Schmidt,Egon,0911/54223
10120 DATA Mayer,Reinhard,632/6546
10130 DATA Meier,Alfons,0882/66532
10140 DATA Schneider,Peter,08365/3333
10150 DATA Mahrbach,Willi,07131/33435
10160 DATA Heusenbach,Susanne,04563/59837
10170 DATA Roehrig,Elfriede,09873/355
10180 DATA Schuster,Leonhard,0876/9983
10190 DATA Preiss,Walter,04134/79833
```

Diese Programmzeilen enthalten - wie gefordert - die zwanzig Einträge. Eine Erweiterung der Datei erfolgt durch Hinzufügen weiterer Datazeilen mit den entsprechenden Inhalten.

Nun muß noch ein kleiner Programmteil geschrieben werden, der sowohl die Eingabe des Suchkriteriums zuläßt, als auch die Suche durchführt. Selbstverständlich soll durch diesen auch eine Bildschirmausgabe erfolgen.

Dieser Teil kann beispielsweise so aussehen, wie das nachfolgende Programmlisting. Bitte diesen Programmteil nicht abtippen, er dient nur zur Erklärung wie derartige Programme arbeiten können. Besser ist dann schon PRG2.BAS.

3.2.1 Basic-Listing: PRG1.BAS

```
100 'Dateiverwaltung 1
110 '
120 MODE 2
130 INPUT"Familien-Name";such$
140 PRINT
150 READ familienname$
160 IF familienname$="ENDE" THEN PRINT"Keinen Eintrag gefunden":END
170 READ vorname$,telefon$
180 IF familienname$<>such$ THEN 250
190 PRINT"Familien-Name:  "familienname$
200 PRINT"Vorname      :  "vorname$
210 PRINT"Telefon-Nummer: "telefon$
220 PRINT
230 PRINT"Bitte eine Taste druecken":CALL &BB06
240 PRINT
250 GOTO 150
```

Vor den Zeilen mit den Datastatements müssen also lediglich diese Programmzeilen eingegeben werden, damit Sie ein kleines Programm zum Auffinden von Telefon-Teilnehmern haben. Da in Zeile 160 die Abfrage nach dem Wort "ENDE" erfolgt, muß zu den Datazeilen noch eine mit dem Inhalt ENDE angefügt werden.

Also: 10200 DATA ENDE

3.2.2 Programmbeschreibung: Dateiverwaltung 1 (PRG1.BAS)

Zeile 100 Das Apostroph-Zeichen kennzeichnet alles in dieser Zeile Folgende nur als Bemerkung. Es dient demselben Zweck wie der Befehl REM (Abkürzung von Remark).

Zeile 110 Ebenfalls nur eine Bemerkung. Diese Zeile dient nur als Trenner, um den Programmkopf, also zum Beispiel um den in Zeile 100 stehenden Programmnamen vom übrigen Programm abzutrennen um dadurch die Übersichtlichkeit zu erhöhen. Das Apostroph-Zeichen ist schneller einzugeben als der Befehl REM, belegt im Speicher allerdings zwei Bytes, REM nur ein Byte.

Zeile 120 MODE-Festlegung auf 80 Zeichen pro Zeile.

Zeile 130 Diese fordert die Eingabe des Suchkriteriums und übernimmt die Eingabe als Stringvariable (such$).

Zeile 140 Macht einen Zeilenvorschub; eine Leerzeile wird ausgegeben.

Zeile 150 Ein Datastatement (das erste jeder Datazeile) wird gelesen und der Stringvariablen familienname$ zugewiesen.

Zeile 160 Prüfung ob Statement-Ende erreicht. Dies trifft dann zu, wenn der Inhalt von familienname$ die Zeichenfolge "ENDE" ist. Ist dies der Fall, dann beendet das Programm seinen Lauf mit der Bemerkung: "Keinen Eintrag gefunden".

Zeile 170 Ist der Inhalt von familienname$ ungleich der Zeichenfolge "ENDE", dann werden durch diese Zeile der Vorname und die Telefonangaben in die Stringvariablen vorname$ und telefon$ eingelesen.

Zeile 180 Falls der aus den Datastatements gelesene Familienname nicht mit dem gesuchten Namen übereinstimmt, dann mit dem Programmlauf weitermachen, also weitersuchen.

Zeile 190 Die Zeilen 190 bis 240 werden nur dann abgearbeitet, wenn eine Suche erfolgreich war. In dieser Zeile erfolgt die Ausgabe des Familiennamens.

Zeile 200 Ausgabe des Vornamens.

Zeile 210 Ausgabe der Telefonnummer.

Zeile 220 bewirkt einen Zeilenvorschub.

Zeile 230 fordert zu einem Tastendruck auf und wartet bis dieser auch ausgeführt wurde.

Zeile 240 bewirkt einen weiteren Zeilenvorschub.

Zeile 250 springt die Zeile 150 an, dort macht das Programm also wieder weiter.

Eine kurze Anmerkung zum Programmablauf:

Nach Eingabe des zu suchenden Strings wird das erste Datastatement in der ersten Datazeile gelesen. Danach wird geprüft, ob "ENDE" gelesen wurde. Wenn ja, dann beendet das Programm mit dem Hinweis "Keinen Eintrag gefunden" seinen Lauf. Da es aber vorkommen kann, daß in den Datastatement-Zeilen mehrere Einträge mit dem gleichen Familien-Namen enthalten sind und auch alle gefunden werden sollen, kann die Meldung auch dann erfolgen, wenn das Programm bereits schon "fündig" war und nun lediglich am Ende der Statements angelangt ist.

"Unschön" an diesem kleinen Programm ist, daß die Datastatements nicht in alphabetischer Reihenfolge stehen. Die CPCs erlauben es aber - in äußerst komfortabler Weise - Zeilen einzuschieben und auch Zeilennummern zu ändern, wodurch Sie sehr leicht Ordnung in diese Folge bringen können.

Durch den Befehl

RENUMBER 10000,10000,10

kann bei Bedarf der Data-Teil auch immer wieder neu durchnumeriert werden.

Doch berücksichtigen wir noch einen weiteren Punkt!

Wem ist es nicht schon einmal so ergangen, den Vornamen eines Bekannten zwar zu wissen, aber der Familienname war "weg"? Selbst wenn dies nur deshalb passierte, weil man sich immer nur mit dem Vornamen ansprach, das Programm kann in der jetzigen Form nicht weiterhelfen.

Eine kleine Änderung und die Suche kann auch mittels des Vornamens erfolgen. Gleichzeitig wurden an dem nun folgenden Programm aber noch weitere kleine Änderungen vorgenommen. Das nachstehende Programm kann deshalb PRG1.BAS ersetzen und ist außerdem auch noch komfortabler und übersichtlicher.

3.3.1 Basic-Listing: Dateiverwaltung 2 (PRG2.BAS)

```
100 'Dateiverwaltung 2
110 '
120 MODE 1
130 PRINT"Telefon-Verzeichnis":PRINT
140 INPUT"Gesucht werden soll:";such$
150 PRINT
160 PRINT CHR$(24);:PRINT"F";:PRINT CHR$(24);:PRINT"amilienname    oder
170 PRINT
180 PRINT CHR$(24);:PRINT"V";:PRINT CHR$(24);:PRINT"orname
190 PRINT
200 PRINT"(Bitte F oder V druecken!)
210 x$=LOWER$(INKEY$)
220 IF x$="" THEN 210
230 PRINT
240 READ familienname$
250 IF familienname$="ENDE" THEN PRINT"Keinen Eintrag gefunden":END
260 READ vorname$,telefon$
270 IF x$="f" THEN vergleich$=familienname$ ELSE vergleich$=vorname$
280 IF vergleich$<>such$ THEN 350
290 PRINT"Familien-Name:  "familienname$
300 PRINT"Vorname      :  "vorname$
310 PRINT"Telefon-Nummer: "telefon$
320 PRINT
330 PRINT"Bitte eine Taste druecken":CALL &BB06
340 PRINT
350 GOTO 240
360 '------------------------------------------------------------
10000 DATA Axtner,Wilhelm,0433/9824376
10010 '
```

```
10020 DATA Brack,Sieglinde,0211/8890943
10030 '
10040 DATA Dutschke,Franz,0211/49823
10050 '
10060 DATA Faber,Richard,03221/8973
10070 '
10080 DATA Gernhuber,Emma,0617/663421
10090 '
10100 DATA Heusenbach,Susanne,04563/59837
10110 DATA Huber,Michael,089/7765776
10120 '
10130 DATA Mahrbach,Willi,07131/33435
10140 DATA Mayer,Reinhard,632/6546
10150 DATA Meier,Alfons,0882/66532
10160 DATA Meier,Peter,0711/356879
10170 '
10180 DATA Preiss,Walter,04134/79833
10190 '
10200 DATA Rinderle,Heinrich,089/443434
10210 DATA Roehrig,Elfriede,09873/355
10220 '
10230 DATA Schmid,Ernst,0422/3216
10240 DATA Schmidt,Egon,0911/54223
10250 DATA Schneider,Peter,08365/3333
10260 DATA Schuster,Leonhard,0876/9983
10270 '
10280 DATA Strobl,Hans,05344/7656
10290 '
10300 DATA Zeidler,Lothar,08122/344
10310 '
10320 DATA ENDE
```

Durch die Änderungen der Zeilen bis 10000 entstand nun ein Programm, bei dem sowohl der Vorname, als auch der Familienname als Suchkriterium dienen kann. Außerdem wurde noch der MODE und die Ausgabe etwas geändert.

Daß dieses Programm nun komplett abgedruckt wurde, hat den Grund, um vor allem Einsteigern etwas über die ersten Klippen hinwegzuhelfen. Die Änderungen zum vorherigen Programm sind nur geringfügig. Wie Sie bemerken konnten, wurden die Datastatements auch in eine übersichtlichere Form gebracht.

Beide bisherigen Programme haben aber einen großen Nachteil. Sie können nur dann Übereinstimmung feststellen, also "fündig" werden, wenn der Suchstring in

genau der gleichen Weise geschrieben wurde, wie er in den Statements steht.

Die Namen "MEIER", "MEier", "MeieR" würden bisher nicht gefunden werden. Falls also die genaue Schreibweise nicht bekannt ist, erfolgen auch keine Ausgaben.

Bei so einem kleinen Datenumfang kein Problem, der Befehl

LIST

zeigt das komplette Programm bzw.

LIST 10000 -

zeigt alle Datazeilen und dann kann nachgelesen oder auch geändert werden.

Aber erstens soll es nicht Sinn eines Programmes sein, selbst "zu Fuß" zu suchen und zweitens ist dies außerdem bei vielen Daten-Einträgen auch nicht gerade das "Gelbe vom Ei".

Deshalb soll dieses Programm nun so abgeändert werden, daß dieser Nachteil nicht mehr vorhanden ist.

```
145 such$=UPPER$(such$)
245 familienname$=UPPER$(familienname$)
265 vorname$=UPPER$(vorname$):telefon$=UPPER$(telefon$)
```

Fügen Sie bitte diese Zeilen in das Programm ein!

Nun können die Eingaben sowohl in Groß-, als auch in Kleinschreibweise gemacht werden und die Einträge "MEIER" werden immer dann gefunden, wenn die Zeichenfolge stimmt. Die Eingabe von "MEIer", "meiER" usw. führt also auch immer zum Erfolg.

Diese simplen Beispiele von Dateiverwaltung sind aber, wie Sie sehen konnten, ein genau festgelegtes Programm nämlich eine Telefondatei.

Adreßverwaltungen sind mit diesen Programmen aber auch möglich, nur muß dann alles was mit der Telefonnummer zu tun hat, geändert werden. Auf diese Weise können Sie kleine Programme für verschiedene Zwecke programmieren.

Wie Sie sich überzeugen können, arbeitet das Programm für den normalen Bedarf schnell genug, solange die Anzahl der Daten nicht zu hoch ist.

Wieviele Datensätze Sie mit einem Programm dieser Art verwalten können, läßt sich nicht ohne weiteres berechnen, denn bisher sind noch keinerlei Festlegungen der Eintragslängen erfolgt.

3.4 Abschätzen der Eintragsanzahl

Damit Sie aber zumindest eine Vorstellung über die Größenordnung haben, nehmen wir folgendes an:

Der Familienname ist maximal 15 Zeichen lang, der Vorname 1Ø und die Telefonnummer hat ebenfalls ein Länge von 15 Zeichen. Das ergibt pro Eintrag 4Ø Zeichen.

Pro DATA-Zeile muß im CPC auch noch die Zeilen-Nummer, die Länge der Zeile, das Wort DATA und die Einträge selbst - inklusive der Kommata - usw. gespeichert werden. Das ergibt pro Eintrag rund 5Ø Zeichen.

Das restliche Programm nimmt auch noch Speicherplatz weg. Deshalb stehen dann je nach Länge dieses Programmteiles noch ca. 3Ø KByte zur Verfügung. Teilt man diesen Wert durch die 5Ø Zeichen, so erhält man in etwa die Anzahl der Einträge.

Mit dem angenommenen Werten ergeben sich etwa 6ØØ Einträge, das sollte eigentlich für eine Telefondatei ausreichen, wenn nicht gerade ein ganzes Telefonbuch aufgenommen werden soll.

Nun haben Sie also ein kleines Programm in zwei Versionen kennengelernt, das den gewünschten Zweck erfüllt. Allerdings ist es vom Bildaufbau her und auch seitens der Bedienung nicht gerade zu den "schönsten" zu zählen.

Um dieses zu verbessern, ist noch einiges zu tun und gerade dieses "Drumherum" wird - wie Sie sehen werden - einigen Programmieraufwand und darum auch Speicherplatz erfordern.

Da sich derartige, kleine Programme zur Demonstration wesentlich besser eignen als umfangreichere und auch

die kleinen Programmiertricks schneller zu erkennen sind, benütze ich diese wieder als Ausgangsbasis für eine weitere Version. Sie bietet auch dem Auge etwas mehr.

Zunächst legen wir einmal fest, daß das nächste Programm im MODE 1 - also mit 40 Zeichen pro Zeile - arbeiten soll. Die Anzahl der Informationen die in diesem Programm bzw. den Daten stecken, sind nicht so umfangreich, daß die Darstellung in diesem MODE nicht mehr möglich wäre.

3.5 Eine komfortablere Variante einer einfachen Datei

Betrachten Sie nun einmal das Listing des Programmes PRG3.BAS.

3.5.1. Basic-Listing: PRG3.BAS

```
100 'Telefondatei
110 '
120 'MODE und BORDER festlegen
130 MODE 1:BORDER 23
140 '
150 inv$=CHR$(24)'                                  String fuer Zeicheninvertierung
160 '
170 'Fenster festlegen
180 WINDOW #1,2,39,2,4'                                Fenster fuer Programm-Namen
190 WINDOW #2,2,39,22,25'                            Fenster fuer Bedienungs-Ausgaben
200 WINDOW #3,2,39,10,15'                 Fenster fuer Suchstring-Eingabe und Ausgabe
210 '
220 READ anzahl'                     Anzahl der Eintraege aus erster Datazeile lesen
230 '
240 PRINT #1,"  Telefon-Datei mit"anzahl"Eintraegen
250 '
260 'Trennstriche ausgeben
270 LOCATE 1,3:PRINT STRING$(40,"-"):LOCATE 1,21:PRINT STRING$(40,"-")
280 '
290 CLS #3'                                                  Fenster 3 loeschen
300 INPUT #3,"Gesucht werden soll:";such$
310 such$=UPPER$(such$)
320 IF LEN(such$)=0 THEN BORDER 0:END
330 PRINT #2,inv$"F";inv$;:PRINT #2,"amilien- oder "inv$"V";inv$;"orname
340 PRINT #2:PRINT #2,"(Bitte F oder V oder ENTER druecken!)
350 '
360 x$=LOWER$(INKEY$):IF x$="" THEN 360'                      Auf Taste warten
```

```
370 IF x$ <> "f" AND x$ <> "v" AND x$<> CHR$(13) THEN 360
380 '
390 CLS #2'                                              Fenster 2 loeschen
400 READ familienname$'                                  Familiennamen lesen
410 IF familienname$="ENDE" THEN 540'                    Eintragsende erreicht
420 READ vorname$,telefon$
430 IF x$="v" THEN vergleich$=UPPER$(vorname$):GOTO 450
440 vergleich$=UPPER$(familienname$)
450 IF vergleich$<>such$ THEN 510'                       Keine Uebereinstimmung
460 '
470 CLS #3:PRINT #3,"Familien-Name:  "familienname$
480 PRINT #3,"Vorname       :  "vorname$
490 PRINT #3,"Telefon-Nummer: "telefon$
500 CLS #2:PRINT #2,"Bitte eine Taste druecken":CALL &BB06
510 GOTO 400
520 '
530 '-------------------------------------------------------------------------
540 RESTORE 10000:CLS #2
550 PRINT #2,"Eintrag "inv$ such$ inv$ " nicht gefunden"
560 PRINT #2,"Bitte eine Taste druecken !
570 CALL &BB06:CLS #2:GOTO 290
580 '-----------------------------------------------------------
590 Anzahl der Eintraege
600 DATA 20
610 '
10000 DATA Axtner,Wilhelm,0433/9824376
10010 '
10020 DATA Brack,Sieglinde,0211/8890943
10030 '
10040 DATA Dutschke,Franz,0211/49823
10050 '
10060 DATA Faber,Richard,03221/8973
10070 '
10080 DATA Gernhuber,Emma,0617/663421
10090 '
10100 DATA Heusenbach,Susanne,04563/59837
10110 DATA Huber,Michael,089/7765776
10120 '
10130 DATA Mahrbach,Willi,07131/33435
10140 DATA Mayer,Reinhard,632/6546
10150 DATA Meier,Alfons,0882/66532

usw. bis....

10320 DATA ENDE
```

In Zeile 13Ø wird der MODE 1 und zum Zwecke einer etwas schöneren Darstellung eine hellere Border-Farbe gewählt.

In Zeile 15Ø wird die Stringvariable inv$ (= Invertierungsstring) definiert. Dies ist kürzer als chr$(24), aber trotzdem ist verständlich was durch diesen geschieht: Die Zeichen werden invertiert.

Da die CPCs die hervorragende Eigenschaft haben, Fensterdefinitionen zuzulassen, machen wir in dieser Programmversion hiervon auch Gebrauch (Zeilen 17Ø bis 2ØØ). Dann erfolgen Bildschirmausgaben ganz gezielt und auch Teilbereiche des Bildschirmes können einfach gelöscht werden.

Um nicht jedesmal wenn die Anzahl der Einträge festgestellt werden soll, die Datazeilen abzählen zu müssen, wurde eine weitere Datazeile hinzugefügt. Sie enthält die Anzahl der enthaltenen Einträge und muß laufend aktualisiert werden. Durch Zeile 22Ø wird die Anzahl gelesen und in Zeile 24Ø ausgegeben.

Durch Zeile 27Ø werden zwei Trennstriche ausgegeben, um die Übersichtlichkeit zu erhöhen.

Durch Zeile 29Ø wird das Ein-/Ausgabe-Fenster gelöscht und in Zeile 3ØØ wird die Eingabe gefordert. Der zu suchende Name wird der Variablen such$ übergeben.

Falls nur die ENTER-Taste betätigt wird, ist die Länge von such$ null und der Programmlauf wird nach Änderung der Border-Farbe beendet (Zeile 32Ø).

Erfolgte aber eine Eingabe, die mit ENTER abgeschlossen wurde, dann wird durch die Zeilen 33Ø/34Ø die Aufforderung, die Taste "F" für den Familien-Namen oder die Taste "V" für den Vornamen zu drücken, ausgegeben. Anstelle der Taste "F" kann auch die ENTER-Taste benutzt werden.

Durch Zeile 36Ø wird auf die entsprechende Eingabe gewartet. Zeile 37Ø "filtert" Fehleingaben aus.

Zeile 39Ø löscht das Fenster für die Ausgabe der Bedienungsmeldungen und in der folgenden Zeile wird der erste Teil der Datas jeder "Eintragszeile" gelesen: Der Familienname. Falls dies das Wort "ENDE" ist

(Zeile 41Ø), wird über den Sprung nach 54Ø der DATA-Zeiger wieder auf den Anfang der Einträge gestellt, die Meldung, daß der Eintrag nicht gefunden wurde ausgegeben etc. und das Programm verzweigt zu Zeile 29Ø. Dort fährt es fort.

Enthält die Variable familienname$ nicht "ENDE", dann wird in Zeile 42Ø der Vorname und die Telefonnummer gelesen.

In Zeile 43Ø wird - je nach vorheriger Wahl - der Stringvariablen vergleich$ entweder der Inhalt von familienname$ oder vorname$ zugewiesen. In der folgenden Zeile erfolgt dann der Vergleich mit dem zu suchenden Kriterium. Ist dieser negativ, dann erfolgt der Sprung zu Zeile 51Ø und von dort aus dann zu 4ØØ um die nächsten Statements einzulesen und der Vergleich wiederholt sich bis zum Ende oder bis die Suche erfolgreich ist.

Ist der Suchvergleich positiv ausgefallen, dann erfolgt durch die Zeilen 47Ø bis 49Ø die Ausgabe. Die Zeile 5ØØ bewirkt, daß das Programm erst nach einem Tastendruck weitermacht.

Obwohl es den eigentlichen Zweck (Ausgabe des Familiennamens, des Vornamens und der Telefonnummer) mehr als erfüllt, zeigt sich auch bei Ihnen wahrscheinlich sehr bald, daß es doch zu weiteren Wünschen Anlaß gibt.

Eine Programmeigenart ist es ja, daß alle Namen komplett (in der gesamten Länge) und auch korrekt eingegeben werden müssen. Falls die genaue Schreibweise des Namens nicht bekannt ist, kann dieser nicht gefunden werden.

Bei dieser sehr geringen Eintragsanzahl kein Problem, mit LIST kann Einsicht in das Programm genommen werden. Nur mit Komfort hat das dann wenig zu tun. Wenn die Anzahl der Einträge hoch ist, dann kann die Suche auch sehr zeitraubend sein.

Bevor wir uns nun anderen Dateiverwaltungsprogrammen zuwenden, wollen wir das bestehende Programm noch um ein paar Funktionen erweitern. Beispielsweise um auch die Suche bei unvollständiger Eingabe zu ermöglichen. Außerdem soll auch noch die Eintrags-Nummer bei gefun-

denen Übereinstimmungen ausgegeben werden. Der Programmaufwand hierfür ist gering. Sie brauchen nur die nachstehenden Basic-Zeilen in das Programm einzufügen.

Listing: ERGZ.BAS

```
415 nummer=nummer+1:IF nummer > anzahl THEN nummer = Ø
445 IF such$="?" THEN 47Ø
45Ø IF INSTR(vergleich$,such$) THEN 47Ø
455 GOTO 39Ø
495 PRINT #3:PRINT #3:PRINT #3,"Eintragsnummer: "nummer
51Ø GOTO 4ØØ
```

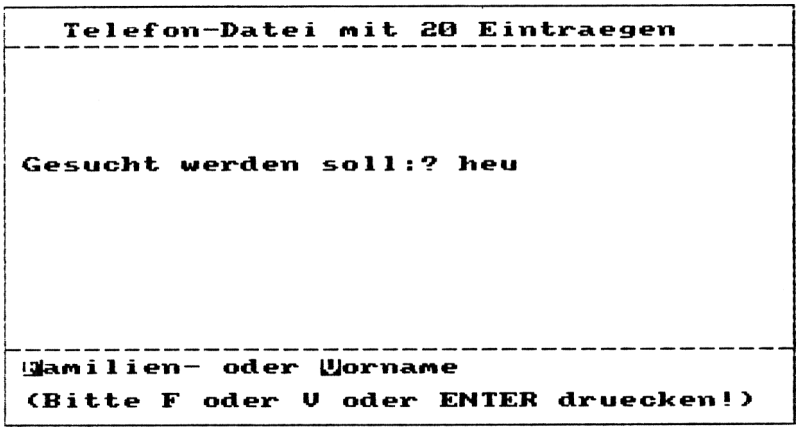


Abbildung 1: Durch die Ergänzungen kann - unabhängig von der Groß-/Kleinschreibweise - auch nach Teil-Namen gesucht werden.

Wie Sie sich erinnern werden, stand zu Beginn dieses Kapitels eigentlich nur der Wunsch nach einer einfachen Telefon-Datei, aber sehr schnell hatte sich gezeigt, daß doch zusätzliche Funktionen programmiert werden mußten.

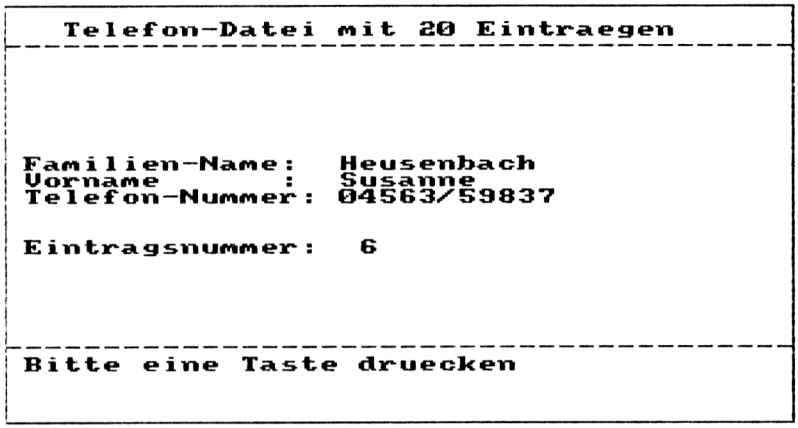


Abbildung 2: So sieht es aus, wenn ein übereinstimmender Eintrag gefunden wurde.

Damit sollte nur aufgezeigt werden, daß vor der eigentlichen Programm-Entwicklung wirklich genau überlegt werden soll und muß, welche Funktionen ein Programm erfüllen soll und auch wie!

Zum Abschluß dieser doch noch sehr einfachen Programme zur Dateien-Verwaltung nun ein weiteres kleines Programm, das aber etwas aus dem Rahmen des üblichen "Datei-Denkens" fällt. Es ist ein kleines Wörterbuch-Programm. Allerdings werden bei diesem nun schon ganz spezielle Eigenheiten der CPCs ausgenutzt und diese auch für den jeweiligen CPC-Typ automatisch angepaßt.

3.6 Wörterbuch in Dateiform

3.6.1 Basic-Listing: Wörterbuch (WOERT.BAS)

```
100 REM Einfaches Woerterbuch Deutsch/Englisch (c) by LM
110 '
120 'CPC-Typ feststellen um Adresse fuer aktive Fehler-Routine zuzuweisen
130 IF PEEK(&AC01)=&C9 THEN fcodeadr = &ADB1 ELSE fcodeadr=&AD98
140 IF fcodeadr=&ADB1 THEN datazeiladr=&AE2E ELSE datazeiladr=&AE15
150 '
160 MODE 1:ON ERROR GOTO 550
170 inv$=CHR$(24)
180 WINDOW #1,11,39,7,7:WINDOW #2,11,39,11,11
190 WINDOW #3,25,29,3,3:WINDOW #7,1,39,23,25
200 '
210 PRINT inv$"Einfaches Woerterbuch (c) by LM"inv$:PRINT
211 LOCATE 1,3:PRINT"Eintrags-Nummer:"
220 LOCATE 1,7:PRINT inv$"Deutsch :":LOCATE 1,11:PRINT"Englisch:"inv$
240 CLS #1:CLS #2:CLS #7
250 RESTORE:zaehler=0:PRINT #3,zaehler
260 LOCATE 1,18
270 PRINT "Suchbegriff in "inv$"D"CHR$(24)"eutsch oder ";
280 PRINT inv$"E"inv$"nglisch ":PRINT inv$"P"inv$" = Programmende
290 x$="":WHILE x$="":x$= LOWER$(INKEY$):WEND
300 LOCATE 1,18
310 IF x$="d" THEN PRINT"Suchbegriff in Deutsch";:flag=1:GOTO 360
320 IF x$="e" THEN PRINT"Suchbegriff in Englisch";:flag=2:GOTO 360
330 IF x$="p" THEN CLS:END
340 GOTO 290
350 '
360 PRINT CHR$(18):PRINT CHR$(18):INPUT #7,"Begriff";begriff$
370 such$=UPPER$(begriff$)
380 LOCATE 25,18:PRINT ":"such$:CLS #7
390 '
400 READ deutsch$,englisch$
410 zaehler=zaehler+1:PRINT #3,zaehler
420 IF flag = 1 THEN vergleichs$=UPPER$(deutsch$)
430 IF flag = 2 THEN vergleichs$=UPPER$(englisch$)
440 vergleichs$=UPPER$(vergleichs$)
450 position=INSTR(vergleichs$,such$)
460 IF position <>1 THEN 390:' Evtl. ersetzen durch:IF position=0 THEN 350
470 PRINT #1,deutsch$:PRINT #2,englisch$
480 PRINT #7,"Neustart= "inv$"N"inv$
490 PRINT #7,"Weitermachen= irgendeine andere Taste
500 x$=LOWER$(INKEY$):IF x$="" THEN 500
510 IF x$="n" THEN 240
520 CLS #7:GOTO 390
530 '
```

```
540 'ON ERROR-ROUTINE deaktivieren
550 POKE fcodeadr,0
560 PRINT #7,"Ende der Eintraege erreicht!"
570 CLS #1:CLS #2:RESTORE
580 PRINT #7,"Neuen Eintrag aufnehmen? (J=ja)"
590 x$=LOWER$(INKEY$):IF x$="" THEN 590
600 IF x$ <> "j" THEN 240
610 '
620 MODE 2
630 PRINT inv$"Neuaufnahme in's Woerterbuch:"inv$
640 PRINT:PRINT"Hierzu mit der COPY-Taste die ausgegebene Zeilen-Nummer,
650 PRINT"sowie das Wort DATA":PRINT"uebernehmen.":PRINT
660 PRINT"Dann das deutsche Wort eingeben, ";
670 PRINT"anschliessend ein Komma, dann das englische "
680 PRINT"Wort und abschliessend die ENTER-Taste betaetigen.":PRINT
690 PRINT"Programm nach Eingabe der neuen Datazeilen mit RUN starten!
700 '
710 spadldatz=PEEK(datazeiladr)+PEEK(datazeiladr+1)*256
720 adresse1$=HEX$(spadldatz,4)
730 LOCATE 1,15
740 PRINT"Informationen*
750 PRINT"Speicheradresse der letzten Datazeile :"adresse1$
760 adresse$="&"+adresse1$:adresse=VAL(adresse$)
770 highbyte=PEEK(adresse):lowbyte=PEEK(adresse+1)
780 zeilennr=highbyte+256*lowbyte
790 zeilennr=zeilennr+10
800 PRINT"Naechste Data-Zeilennummer :"zeilennr
810 LOCATE 1,20:PRINT zeilennr "DATA "
820 LOCATE 1,19
830 END
840 '
850 'Deutsch - English
860 DATA Kueche,kitchen
870 DATA Maedchen,girl
880 DATA Hilfe,help
890 DATA Klingel,bell
900 DATA laden,load
910 DATA speichern,save
```

Nun folgt eine nur kurze Programmbeschreibung, denn aufgrund der vorherigen Beschreibungen dürften kaum Verständnisprobleme auftreten.

Das Programm enthält ein paar kleine Tricks, die auch sehr gut bei anderen Programmen verwendet werden können. So beispielsweise die CPC-Typ-Feststellung und die daraufhin zu erfolgende Reaktion. Außerdem ist

auch die Übernahme von neuen Daten nicht gerade auf normale Weise programmiert.

Versuchen Sie einmal herauszufinden, woher der CPC weiß, welche Datazeilen-Nummer er bei einem neuen Eintrag ausgeben muß. Die Zeilen von 71Ø bis 8ØØ sind hierfür zuständig. Die Adressen aus denen ein CPC dies "abholen" kann sind zwischen dem CPC 464 und seinen Nachfolgern unterschiedlich. Deshalb die Zeilen 12Ø bis 14Ø.

Die Fehlercode-Adresse (fcodeadr) ist die Adresse, die anzeigt, daß ein Fehler aufgetreten ist. Die Datazeilen-Adresse (datazeiladr) ist ebenfalls unterschiedlich bei den CPCs.

Weshalb die Zeile 55Ø enthalten ist, können Sie leicht feststellen, indem Sie nach dieser Zeilen-Nummer REM schreiben und dann mit dem Programm ein paar Durchgänge mit nicht enthaltenen Datas machen!

```
Einfaches Woerterbuch (c) by LM
Eintrags-Nummer:              6

Deutsch : speichern

Englisch: save


Suchbegriff in Deutsch   :SPEICHERN


Neustart= J
Weitermachen= irgendeine andere Taste
```

Abbildung 3: So sieht die Ausgabe beim Wörterbuch-Programm aus.

Ein gut gemeinter Ratschlag noch zu diesem Programm:

> Vergessen Sie nicht, nach Neueingaben das Programm wieder abzuspeichern, denn sonst waren die Neueinträge umsonst!

Damit wollen wir diese Art der einfachen Dateiverwalter abschließen und uns anderen Techniken der Dateiverwaltung zuwenden. Da die folgenden Programme auch wesentlich leistungsfähiger sind, wird es auch schwieriger, sie zu verstehen.

Trotzdem habe ich versucht, auch diese Programme so zu strukturieren und so zu kommentieren, daß Sie auch diese durchschauen.

4 Vorüberlegungen zu komfortablen Programmen

Während bei den vorangegangenen Programmen die Daten im Programm selbst standen, wollen wir nun eine andere Technik benutzen. Die Daten sollen als separate Files geladen und auch wieder abgespeichert werden können. Dies hat den Vorteil, daß ein und dasselbe Datei-Programm, Daten verschiedener Art (aber von gleicher Struktur) be- und verarbeiten kann.

Als Datenspeicher könnte in einfachen Fällen zwar ein Kassettenrekorder dienen, aber es sollte eigentlich immer mindestens eine Diskettenstation eingesetzt werden. Wenn es - wegen enorm hohen Datenanfalles - sein sollte, mittels einer Harddisc die Dateiverwaltung durchführen zu müssen, dann kann dieses Buch zwar jede Menge an Informationen geben, aber ein Programm für den Harddisc-Betrieb ist nicht enthalten.

Da in Computer-Zeischriften viele Dateiverwaltungs-Programme abgedruckt wurden und werden, die mit Stringvariablen arbeiten, will ich in diesem Buch soweit wie möglich darauf verzichten und nun einen Weg aufzeigen, der viele Vorteile hat:

Wir richten eine sogenannte RAM-Disk ein.

Gleich vorweg soll aber auch nicht verschwiegen werden, daß eine derartige Programmtechnik auch Nachteile hat. Diese sind meiner Meinung nach aber nicht so schwerwiegend, daß sie die Vorteile übertreffen.

4.1 Was ist eine RAM-Disk?

Ganz simpel ausgedrückt ist dies ein RAM-Speicherbereich, der von Basic-Programmen nicht benutzt werden kann oder darf.

Es kann auch beim CPC 464 oder 664 RAM-Disk-Betrieb gemacht werden, wenn der Speicher für Basic begrenzt wird und ein großer Teil von RAM-Stellen als "schneller" Speicher zum Einsatz kommt. Beim CPC 6128 mit seinen zusätzlichen 64 KByte ist es sowieso klar, daß RAM-Disk-Betrieb möglich ist.

Für komfortable Dateiverwaltungs-Programme gehe ich davon aus, daß alle wichtigen Informationen von Datenfiles geliefert werden und sowenig Eingaben wie möglich vom Programmbediener gefordert werden. Das bedeutet aber auch, daß von einem Datenfile die Anzahl der Einträge, die Anzahl der Felder pro Eintrag, aber auch die Namen der Felder und deren Bildschirm-Positionen geliefert werden.

Ein Datenfile, das solche Informationen liefern soll, wird auch mit einem Programm erstellt. Ein derartiges Programm kann als Unterroutine in einem Dateiverwaltungsprogramm stehen, sinnvoller ist aber sicherlich ein getrenntes Programm hierfür, da bei dem ohnehin meist sehr knappen Bereich für Daten das Programm für die eigentliche Dateiverwaltung sehr kurz sein sollte.

Ein Programm, mit dem das eben genannte File erstellt wird und mit dem auch die Werte für die Maske, also für die spätere Bildschirmdarstellung der Einträge, geschrieben werden, nennt man Maskengenerator.

4.2 Zweck von Maskengeneratoren

Sinn eines Maskengenerators ist es, den Bildschirmaufbau für ein beliebiges Dateiprogramm zu generieren und so aufzubereiten, daß die Werte vom eigentlichen Dateiverwalter oder auch von Zusatz- und Hilfsprogrammen übernommen werden können. Dadurch müssen diese Werte nicht immer wieder in den verschiedenen Programmen festgelegt werden.

Diesen Zweck erfüllt beispielsweise das in diesem Buch enthaltene Programm MASKGEN.BAS für die später noch

folgenden Dateiverwaltungen. Das Programm finden Sie im Kapitel der Programmbeschreibungen. (Dies trifft für alle größeren Programme zu, denn ich wollte den "fließenden" Text nicht zu sehr mit den Listings "stören".)

Der Maskengenerator ist so geschrieben, daß er für die vier komfortablen "Dateiverwalter" dieses Buches eingesetzt werden kann.

Da auch meine Überlegungen anfangs nicht weit genug gingen, hatte jedes Programm einen eigenen Maskengenerator. Nachdem ich aber später erkannt hatte, daß alle Programme mit den gleichen Daten auskommen können, wurde aus den ürsprünglich vier Programmen eines erstellt, das alle vier Dateiverarbeitungsprogramme "beliefern" kann.

Ursprünglich war von mir auch geplant, eigentlich nur ein einziges Grundprogramm für die Dateiverwaltung zu schreiben und die Versionen lediglich durch hinzuladen von Programmteilen zur gewünschten Version zu machen. Von diesem Grundgedanken bin ich dann aber wieder abgegangen, weil die Unterschiede doch so umfangreich sind und auch weil ich aus der Erfahrung heraus weiß, daß es vielen CPC-Besitzern Schwierigkeiten bereitet, wenn sie nicht die kompletten Programme abschreiben können. Vor allem Einsteiger haben oft gewaltige Schwierigkeiten beim Ergänzen durch "MERGE".

Deshalb können Sie feststellen, daß alle Programme zwar immer die gleiche Grundstruktur haben, aber in den Details und auch in den Zeilen-Nummern variieren. Fortgeschrittene CPC-Besitzer sind bestimmt in der Lage - wenn Sie die verschiedenen Programme haben wollen - aus einem der Programme auch die anderen Versionen aufzubauen. Die anderen bekommen die fertigen Listings "serviert".

Es wiederholen sich also bestimmte Grundprogrammteile mehrfach, aber ich kann ja niemanden von Ihnen über die Schulter sehen und ggf. helfen wenn Probleme auftreten. Das ist auch mit ein Grund, weshalb die Programme komplett abgedruckt sind.

Aber, ich wiederhole es, die verschiedenen Dateiverwaltungsprogramme benötigen nur einen einzigen Maskengenerator.

Hintergedanke beim Maskengenerator war aber nicht nur der Wunsch nach dem Maskenaufbau, sondern auch die Lösung eines Problemes das häufig auftritt: Der Bildaufbau wird durch zu lange Eingaben "gestört". Eine Methode um dies zu verhindern ist der Einsatz von programmierbaren Fenstern im Programm. Dadurch wird das "Bild" dann zwar nicht mehr zerstört, aber bei zu langen Eingaben "rutschen" diese wieder aus den Fenstern heraus und sind dann verschwunden.

Viele Programmierer zeigen deshalb die Grenzen der Fenster durch irgendwelche Kennzeichen an, oder stellen die Fenster invertiert dar. Wird aber "blind" eingegeben, dann kann es doch wieder zu Überschreitungen der Fenstergrenzen kommen. Aus diesem Grunde habe ich mir etwas völlig anderes überlegt.

4.3 Der Window-Editor

Durch eine kleine Maschinenroutine kann die Eingabe innerhalb eines Fensters (Window) ganz ausgezeichnet unterstützt werden. Dieses Programm habe ich WINDOW-EDITOR getauft. Die Kurzbezeichnung (wegen der Abspeicherung auf der Diskette) lautet: WED.

Sinn und Zweck dieser Routine ist es, Eingaben nur innerhalb des Fensters zuzulassen, ohne daß bereits eingegebene Zeichen verschwinden. Das Verlassen des Eingabefensters soll nur mittels ganz bestimmter Tasten möglich sein. Eigentlich nur mittels der Taste ENTER, da ja nur diese normalerweise eine Eingabe abschließt. Aus Komfortgründen wird bei den Programmen aber dann noch die COPY-Taste zugelassen.

Diese Routine kann für viele Programme eingesetzt werden und ist auch Bestandteil der Dateiverwaltungsprogramme.

Für diejenigen, die genau wissen wollen wie dieses Programm arbeitet, folgt nun der Quellcode. Alle Quellprogramme dieses Buches tragen die Erweiterungs-Kennzeichnung (Extension) ".EDI". Der Grund ist: Sie wurden mit einem Text-EDItor erstellt. Dadurch sind sie leicht von anderen Programm-Kennzeichnungen wie ".BAS" für Basicprogramm, "*.BAK" für Backup usw. im Inhaltsverzeichnis (Directory) der Diskette zu erkennen.

Dieses Assembler-Listing ist - wie alle dieses Buches - ausreichend mit Kommentaren versehen, wodurch Interessierte das Programm leichter verstehen können.

4.3.1 Assembler-Listing: WED.EDI

```
                ;****************************************
                ;*              WED.EDI                 *
                ;*          Window-Editor               *
                ;*______________________________________*
                ;*                                      *
                ;*     Copyright by Lothar Miedel       *
                ;*--------------------------------------*
                ;* ermoeglicht die Eingabe von Zeichen  *
                ;*  und Steuerung mittels Cursortasten  *
                ;*      innerhalb eines Fensters        *
                ;****************************************

                ;Firmwareroutinen

waitch  equ   &bbØ6       ;auf Zeichen warten
tout    equ   &bb5a       ;Text output
setcurs equ   &bb75       ;Cursor setzen
curen   equ   &bb7b       ;Cursordarstellung ein
curdis  equ   &bb7e       ;Cursordarstellung aus
valid   equ   &bb87       ;Cursor im Window ?
placur  equ   &bb8a       ;Stelle Cursor auf Schirm dar
remcurs equ   &bb8d       ;Loesche Cursor auf dem Schirm
strsel  equ   &bbb4       ;Selektiere Stream
getcur  equ   &bb78       ;aktuelle Cursorposition ?

        org   &aØØØ

AØØØ  3A 75 AØ          ld    a,(stream)   ;stream holen
AØØ3  CD B4 BB          call  strsel       ;und auswaehlen
                ;-------------------------------------------------
AØØ6  CD 78 BB   wloop  call  getcur       ;Cursorposition holen
AØØ9  CD 75 BB          call  setcurs      ;Cursor setzen
AØØC  CD 8A BB          call  placur       ;Cursor auf erste Position
AØØF  CD Ø6 BB          call  waitch       ;Auf Zeichen warten
AØ12  4F                ld    c,a          ;Zeichen retten
AØ13  CD 8D BB          call  remcurs      ;Cursorsymbol entfernen
AØ16  79                ld    a,c          ;Zeichen wieder holen
AØ17  FE ØD             cp    a,13         ;ist es return ?
AØ19  C8                ret   z            ;ja -> Basic
AØ1A  FE F2             cp    a,242        ;ist es Cursor links ?
```

```
A01C  CA 41 A0          jp    z,links      ;ja -> Routine links
A01F  FE F3             cp    a,243        ;ist es Cursor rechts ?
A021  CA 57 A0          jp    z,rechts     ;ja -> Routine rechts
A024  FE 7F             cp    a,&7f        ;Zeichen groesser &7f
A026  F2 06 A0          jp    p,wloop      ;ja weitermachen
A029  FE 20             cp    a,32         ;Zeichen kleiner 32
A02B  FA 06 A0          jp    m,wloop      ;ja weitermachen

A02E  CD 5A BB    outp  call  tout         ;und ausgeben
A031  CD 78 BB          call  getcur       ;pruefen ob im Fenster
A034  3A 77 A0          ld    a,(mrechts)  ;rechten Maximalwert
A037  3C                inc   a            ;erhoehen
A038  57                ld    d,a          ;und merken
A039  7C                ld    a,h          ;aktuelle Position holen
A03A  BA                cp    a,d          ;und vergleichen
A03B  CA 6D A0          jp    z,wrr        ;rechte Windowposition erreicht
A03E  C3 06 A0          jp    wloop        ;weitermachen

A041  3E 08       links ld    a,8          ;Cursor links laden
A043  4F                ld    c,a          ;und merken

A044  CD 78 BB          call  getcur       ;Cursorposition holen
A047  3A 76 A0          ld    a,(mlinks)   ;linken Grenzwert
A04A  57                ld    d,a          ;merken
A04B  7C                ld    a,h          ;mit aktuellem
A04C  BA                cp    a,d          ;Wert vergleichen
A04D  CA 06 A0          jp    z,wloop      ;Rand erreicht
A050  79                ld    a,c          ;Backspace holen
A051  CD 5A BB          call  tout         ;und ausfuehren
A054  C3 06 A0          jp    wloop        ;weitermachen

A057  3E 09       rechts ld   a,9          ;Cursor rechts laden
A059  4F                ld    c,a          ;und merken
A05A  CD 78 BB          call  getcur       ;aktuellen Wert holen
A05D  3A 77 A0          ld    a,(mrechts)  ;rechten Grenzwert
A060  57                ld    d,a          ;merken mit
A061  7C                ld    a,h          ;aktuellen Wert
A062  BA                cp    a,d          ;vergleichen
A063  CA 06 A0          jp    z,wloop      ;Rand erreicht
A066  79                ld    a,c          ;Cursor rechts
A067  CD 5A BB          call  tout         ;ausfuehren
A06A  C3 06 A0          jp    wloop        ;und weitermachen

A06D  3E 08       wrr   ld    a,08         ;Backspace
A06F  CD 5A BB          call  tout         ;ausgeben
A072  C3 06 A0          jp    wloop        ;und weiter
```

```
A075  07            stream  defb  &07
A076  05            mlinks  defb  &05
A077  06            mrechts defb  &06

Fehler: 00000  Warnungen: 00000

Symboltabelle:

BB7B CUREN     BB7E CURDIS     BB78 GETCUR     A041 LINKS     A076 MLINKS
A077 MRECHTS   A02E OUTP       BB8A PLACUR     BB8D REMCURS   A057 RECHTS
BB75 SETCURS   BBB4 STRSEL     A075 STREAM     BB5A TOUT      BB87 VALID
BB06 WAITCH    A006 WLOOP      A06D WRR
```

Diese Routine kann durchaus noch ergänzt werden, was für die späteren Programme auch noch geschieht. Aber dadurch, daß es an dieser Stelle für einen anderen Adressbereich assembliert wurde, können Sie diese Version auch für andere Programme besser einsetzen.

Die Erstellung des Programmes "WED" hat aber noch einen weiteren Grund. Denn meine Überlegungen gingen noch weiter, als ich es bisher geschrieben habe. Dieser Maschinenspracheteil ist nämlich bereits die Basis für das später noch beschriebene Umkopieren vom Bildschirm in die RAM-Disk und umgekehrt. Damit Sie aber den Sinn dieser Routine nicht nur verstehen, sondern auch in der Praxis erleben können, folgt nun ein kleines Demonstrationsprogramm.

4.3.2 Basic-Listing: WED.BAS

```
100 'WED.BAS
110 '
120 MEMORY &9FFF:MODE 2:PRINT"Demo des Window-Editors
130 '
140 IF PEEK(&A000)<>58 THEN GOSUB 240
150 stream=7:oben=10:unten=10:links=30:rechts=75
160 mstream=&A075:mrechts=&A077:mlinks=&A076
170 '
180 WINDOW #7,links+1,rechts+1,oben,unten
190 PRINT #7,CHR$(24):CLS #7:PRINT #7,CHR$(24):CLS #7
200 POKE mstream,stream:POKE mlinks,1:POKE mrechts,rechts-links
210 CALL &A000:oben=oben+1:unten =unten +1:IF oben >=20 THEN END ELSE 180
220 '
230 'Der nachfolgende Teil ist der Hexlader
240 a=&A000:e=&A074:zb=1000:e=e+1
```

```
250 FOR i =a TO e:READ d$:IF LEFT$(d$,1)="&" THEN flag =1
260 IF (flag AND ps<>VAL(d$)) THEN PRINT"Fehler in Zeile "zb+1:END
270 IF (flag AND i=e) THEN RETURN
280 IF flag THEN i=i-1:zb=zb+1:ps=0:d$="":flag = 0:GOTO 300
290 d$="&"+d$:POKE i,VAL(d$):ps=ps+VAL(d$):
300 IF i < e THEN NEXT i
310 '
1001 DATA 3A,75,A0,CD,B4,BB,CD,78,BB,CD,75,BB,CD,8A,BB,CD,&0A67
1002 DATA 06,BB,4F,CD,8D,BB,79,FE,0D,C8,FE,F2,CA,41,A0,FE,&0A0A
1003 DATA F3,CA,57,A0,FE,7F,F2,06,A0,FE,20,FA,06,A0,CD,5A,&09AE
1004 DATA BB,CD,78,BB,3A,77,A0,3C,57,7C,BA,CA,6D,A0,C3,06,&0875
1005 DATA A0,3E,08,4F,CD,78,BB,3A,76,A0,57,7C,BA,CA,06,A0,&0782
1006 DATA 79,CD,5A,BB,C3,06,A0,3E,09,4F,CD,78,BB,3A,77,A0,&07AB
1007 DATA 57,7C,BA,CA,06,A0,79,CD,5A,BB,C3,06,A0,3E,08,CD,&07D4
1008 DATA 5A,BB,C3,06,A0,&027E
```

Da das Programm nicht sehr umfangreich ist, sollten Sie sich wirklich die Mühe machen, es abzuschreiben und laufen zu lassen, um damit etwas zu experimentieren. Dadurch stellen Sie dann fest, daß Sie außer mit der Enter-Taste (oder RETURN) keine Möglichkeit haben, das Fenster zu verlassen. Das bedeutet auch, daß ein Verschwinden von Zeichen wenn Sie am rechten oder linken Window-Ende angekommen sind und weiterschreiben, nicht erfolgt.

"Bewegungen" im Fenster sind nur durch die beiden Cursor-Steuertasten "links" und "rechts" möglich. Die Delete-Funktion wurde ebenfalls verhindert. Löschungen erfolgen durch Überschreiben mit anderen Zeichen, wie z.B. dem Leerzeichen.

Wie Sie ebenfalls bemerken werden, kann die Größe und die Position eines beliebigen Fenstes während eines Programmlaufes jederzeit geändert werden. Im Beispiel wird hierzu Window #7 benutzt.

Allerdings habe ich in diesem Programm nur "einzeilige" Fenster berücksichtigt, denn für die Dateiverwaltungs-Programme dieses Buches reicht dies aus.

4.4 Schnell umkopiert

Nun wäre es noch sehr gut, wenn die Window-Eingaben direkt in den RAM-Speicher gelesen werden könnten. Der große Vorteil wäre: Es bräuchten keinerlei Zeichenumwandlungen zu erfolgen und vor allem entfiele dann jegliche Stringbehandlung für die Daten.

Dazu sollte dann aber auch die "Gegenroutine" zum Schreiben aus dem normalen RAM-Speicher-Bereich auf den Bildschirm vorhanden sein.

Auch das kann in Maschinensprache sehr leicht gelöst werden.

Wie dies funktioniert, können Sie dem nachfolgenden Assembler-Listing entnehmen, denn wie für alle Maschinencode-Routinen dieses Buches ist auch dieses ausführlich kommentiert. Für die späteren "echten" Programme sehen Sie bitte in den Abschnitten RAMWR und RAMRD nach. Diese sind nichts anderes als der Programmteil COPY und erledigen dort das Lesen von Zeichen, die sich auf dem Bildschirm befinden und auch das Schreiben in einen ganz genau festgelegten RAM-Bereich und umgekehrt. Daß dies auch noch sehr schnell geschieht, ist ein ebenfalls nicht zu unterschätzender Vorteil gegenüber Basic.

Das Programm wurde so assembliert, daß Sie es direkt hinter den Window-Editor "schreiben" können. Diesmal ist allerdings kein Lade-Programm abgedruckt. Das können Sie sich dadurch, daß beim Assembler-Listing alle Codes enthalten sind, selbst schreiben.

4.4.1 Assembler-Listing: COPY.EDI

```
;****************************************
;*              COPY.EDI                *
;* -------------------------------------*
;*     Copyright by Lothar Miedel       *
;*                                      *
;*                                      *
;* kopiert ASCII-Zeichen vom Bildschirm *
;* in jeden beliebigen Speicherbereich  *
;*              um umgekehrt!           *
;*--------------------------------------*
;* Frei waehlbar, bzw. von Basic aus zu *
```

```
                    ;* poken sind ->                          *
                    ;*                                       *
                    ;* Die Zeile, die Spalte, die Anzahl     *
                    ;* der zu kopierenden Zeichen, der       *
                    ;* Stream von dem gelesen werden soll    *
                    ;* und die RAM-Disk (der Bereich in den *
                    ;* geschrieben werden soll)!             *
                    ;*****************************************
                    ;
;Benutzte Firmware-Routinen mit Funktion (F), Einsprungbedingungen (E)
;und Aussprungbedingungen (A)

txt_set_column equ   &bb6f  ;F= Setze die horizontale Position des Cursors
                    ;E= a enthaelt die logische Spalte
                    ;A= af und hl zerstoert

txt_set_row    equ   &bb72   ;F= Setze die vertikale Position des Cursors
                    ;E= a enthaelt die logische Zeile
                    ;A= af und hl zerstoert

txt_str_select equ   &bbb4   ;F= Selektiere das Ein-/Ausgabegeraet (Stream)
                    ;E= a enthaelt Stream-Nummer
                    ;A= a enthaelt vorherigen Stream
                    ;   hl und Flags zerstoert

txt_rd_char    equ   &bb60        ;F= Lese Zeichen vom Bildschirm
                    ;E= keine
                    ;A=   wenn Zeichen erkannt wurde
                    ;         CARRY an a enthaelt Zeichen
                    ;     wenn kein Zeichen erkannt wurde
                    ;        CARRY aus a enthaelt Null
                    ;   immer alle anderen Flags zerstoert

txt_wr_char    equ   &bb5d        ;F= Schreibe Zeichen auf den Schirm
                    ;E= a enthaelt das Zeichen
                    ;A= af,bc,de,hl zerstoert
;-----------------------------------------------------

;Lesen von Bildschirmzeichen und schreiben in RAM-Disk

                    org   &a078

 A078  3A D1 A0     ramwr   ld    a,(stream)     ;Streamauswahl
 A07B  CD B4 BB             call  txt_str_select ;aufrufen
;-----------------------------------------------------
 A07E  3A CE A0             ld    a,(zeile)      ;Zeilenposition
 A081  F5                   push  af             ;retten
 A082  CD 72 BB             call  txt_set_row    ;positionieren
```

```
A085  F1                      pop   af              ;Position wieder
A086  32 CE A0                ld    (zeile),a       ;rueckspeichern
;-------------------------------------------------------
A089  3A CF A0      arbeite   ld    a,(spalte)      ;Spalte holen
A08C  F5                      push  af              ;und merken
A08D  CD 6F BB                call  txt_set_column  ;Spalte positionieren
A090  F1                      pop   af              ;wieder holen
A091  3C                      inc   a               ;erhoehen
A092  32 CF A0                ld    (spalte),a      ;neue Spalte merken
;-------------------------------------------------------
A095  2A D0 A0                ld    hl,(anzahl)     ;verbliebene Zeichenanzahl
A098  7D                      ld    a,l             ;Anzahl zum Vergleich in Akku
A099  FE 00                   cp    0               ;Ende ?
A09B  C8                      ret   z               ;ja -> Basic
;-------------------------------------------------------
A09C  2B                      dec   hl              ;Zeichenanzahl erniedrigen
A09D  22 D0 A0                ld    (anzahl),hl     ;neue Anzahl merken
;-------------------------------------------------------
A0A0  CD 60 BB                call  txt_rd_char     ;Lesen - Zeichen in den Akku
A0A3  2A D2 A0                ld    hl,(ramdisk)    ;RAM-Disk-Adresse holen und
A0A6  77                      ld    (hl),a          ;Akkuinhalt in RAM-Disk
A0A7  23                      inc   hl              ;RAM-Disk-Adresse erhoehen
A0A8  22 D2 A0                ld    (ramdisk),hl    ;und merken
A0AB  C3 89 A0                jp    arbeite         ;weitermachen
A0AE  00                      nop
;-------------------------------------------------------
;Lesen von Zeichen aus der RAM-Disk und schreiben auf den Bildschirm

A0AF  3A D1 A0      ramrd     ld    a,(stream)      ;Stream auswaehlen
A0B2  CD B4 BB                call  txt_str_select
;-------------------------------------------------------
A0B5  2A D2 A0      ramles    ld    hl,(ramdisk)    ;Adresse der RAM-Disk holen
A0B8  7E                      ld    a,(hl)          ;Zeichen aus RAM-Disk lesen
;-------------------------------------------------------
A0B9  23                      inc   hl              ;RAM-Diskadresse erhoehen
A0BA  22 D2 A0                ld    (ramdisk),hl    ;und merken
A0BD  CD 5D BB                call  txt_wr_char     ;Zeichen aus Akku schreiben
;-------------------------------------------------------
A0C0  2A D0 A0                ld    hl,(anzahl)     ;Zeichenanzahl holen
A0C3  7D                      ld    a,l             ;Anzahl in den Akku
A0C4  FE 00                   cp    a,0             ;vergleichen auf Null
A0C6  C8                      ret   z               ;zurueck ins Basic
;-------------------------------------------------------
A0C7  2B                      dec   hl              ;Zeichenanzahl erniedrigen
A0C8  22 D0 A0                ld    (anzahl),hl     ;und merken
A0CB  C3 B5 A0                jp    ramles
;=======================================================
```

```
;Speicherstellen um (von Basic aus) die gewuenschten
;Werte uebergeben zu koennen.

 AØCE  Ø1          zeile   defb  &Ø1
 AØCF  Ø1          spalte  defb  &Ø1
 AØDØ  Ø1          anzahl  defb  &Ø1
 AØD1  Ø7          stream  defb  &Ø7
 AØD2  ØØ 3Ø       ramdisk defw  &3ØØØ

Fehler: ØØØØØ  Warnungen: ØØØØØ

Symboltabelle:

AØ89 ARBEITE              AØDØ ANZAHL             AØ78 RAMWR
AØAF RAMRD                AØB5 RAMLES             AØD2 RAMDISK
AØCF SPALTE               AØD1 STREAM             BB6F TXT_SET_COLUMN
BB72 TXT_SET_ROW          BBB4 TXT_STR_SELECT     BB6Ø TXT_RD_CHAR
BB5D TXT_WR_CHAR          AØCE ZEILE
```

4.5 Hilfsprogramme für die Diskettendateien

Bisher sind wir mit den Überlegungen immer davon ausgegangen, daß die gesamten Daten irgendwo im CPC-Speicher stehen. Aber immer dann, wenn der Platz im "Schneider" zu gering ist, dann müssen andere Wege gefunden werden, um auch derartig große Datenmengen zu bewältigen.

Eine Möglichkeit ist es den RAM-Speicher zu erweitern. Zu diesem Zweck gibt es verschiedene Speicher-Erweiterungen auf dem Markt.

Wer etwas elektronik-bastlerisches Geschick hat, kann einen solchen Zusatz-Speicher leicht in einen CPC einbauen. Derjenige könnte aber Pech haben, daß das eine oder andere Programm dieses Buches nicht auf Anhieb läuft. Geringe Änderungen dürften aber auch dieses Problem lösen. Leider kann ich keine Tips hierzu geben, da es verschiedene Speicherkarten gibt.

Aber auch mit einer Speicherkapazität von 512 KByte kann es sein, daß es noch nicht reicht! Und was soll der tun, der noch Garantie auf seinen CPC hat und sie durch einen Eingriff ja verlieren würde? Aus diesem Grunde nun Überlegungen nach einem anderen Weg.

Die Diskettenlaufwerke der Schneider-Computer zeichnen sich durch relativ hohe Lese- und Schreibgeschwindigkeiten aus. Ein Zugriff lesender oder schreibender Art, nimmt deshalb wenig Zeit in Anspruch und ist für eine Dateiverwaltung noch gut zu gebrauchen. Die Daten werden so schnell geholt, daß man es "erwarten" kann.

Nur eines sollte auch bei Dateiverwaltungs-Programmen dieses Buches, die mit den Disketten arbeiten, nicht geschehen, nämlich die Daten über Strings zu handhaben. Aus diesem Grunde kommen nur direkte Zugriffe auf die Diskettensektoren in Frage. Genau dies lassen die CPCs aber nicht von Basic aus zu. Deshalb muß auch dafür eine Maschinensprache-Routine her.

Dabei könnten nun bei dem einen oder anderen CPC-Besitzer Probleme auftreten, weil er nicht mit dem AMSTRAD-Disketten-Betriebs-System arbeitet. Das hat mir etwas Kopfzerbrechen bereitet. Aber ich habe - glaube ich - eine Lösung gefunden.

Wenn die Forderungen, die seitens Amstrad an sogenannte Sideway-ROMS, und das DOS ist eines davon, eingehalten werden, dann müsste es mit einer der beiden Maschinenroutinen die ich geschrieben habe, auf alle Fälle klappen.

Das ist also meine Lösung: Zwei unterschiedliche Maschinen-Programme! Damit die Handhabung dieser Routinen klar wird, verwende ich beim Programm RANDOMD1 die eine und bei RANDOMD2 die andere.

Die erste Routine läuft ganz sicher unter AMSDOS (also dem Original-CPC-DOS) und die zweite arbeitet ganz sicher mit dem VORTEX-Laufwerk F1-X, also dem 5,25-Zoll-Laufwerk mit X-Modul.

Die zweite Version wurde in Verbindung mit einem CPC 6128 getestet, wobei das VDOS-ROM als Erweiterungs-ROM eins geschaltet war.

Die erste Disketten-Routine - Sie finden sie bei den Programmbeschreibungen - heißt DISC1 und wurde sehr gut durchdacht. Mit ihr ist es möglich, von Basic aus alle Diskettenbefehle aufzurufen. Also sowohl die Befehle &81 bis &89, als auch andere wie DISC, A, B und auch Direkt-Einsprünge in das DOS sind möglich.

Die neun Diskettenbefehle können direkt übergeben werden, für die anderen muß die Adresse an der sie stehen mitgeteilt werden, dann sucht das Programm durch Ansprung der Firmware-Routine KL FIND COMMAND die anzuspringende Adresse selbst.

Deshalb hat der erste Teil des Maschinenprogrammes auch zwei Einsprungmöglichkeiten. Will man direkt einspringen, dann muß sowohl die gewünschte Adresse mit POKE in den Speicher geschrieben werden, als auch die ROM-Nummer. Durch den Merker RUE kann sogar von Basic aus abgefragt werden, ob ein Diskettenbefehl korrekt ausgeführt wurde.

Diese letzten Hinweise waren nur für Spezialisten gedacht, die diese Routine ganz hervorragend für beispielsweise einen Disketten-Monitor einsetzen können. Die Dateiverwaltungsprogramme für den direkten Zugriff könnten übrigens bei nur ganz kleinen Modifikationen sogar als Disketten-Monitor-Programm dienen! Ist doch klar warum und wie, oder? (Falls nicht, tut es mir leid, aber in diesem Buch geht es um Dateien).

Am Ende des ersten Diskettenhilfsprogrammes ist dann noch eine kleine Tabelle enthalten, die zur Formatierung von Disketten dient. Diese Tabelle wird in Verbindung mit einem zusätzlichen kleinen Basic-Programm dann verwendet, wenn den Schneider-Disketten ein Spezial-Format verpaßt wird.

Sie haben es bestimmt schon selbst bemerkt: Mit diesem Buch erfahren Sie nicht nur einiges über Dateiverwaltungen, sondern erhalten gleichzeitig eine Menge Hilfs-Routinen und Tips auch für andere Programme.

Nun zur zweiten Hilfsroutine. Sie heißt DISC2 und ist ebenfalls bei der Programm-Beschreibung für RANDOMD1 zu finden. Zwar klappt mit dieser der Zugriff auf die Schneider-Laufwerke ebenfalls, aber sie wurde eben noch zusätzlich geschrieben um Ihnen evtl. bei anderen Betriebssystemen zu helfen.

Eine weitere Maschinensprache-Routine soll Ihnen nicht verheimlicht werden. Sie dient als Suchprogramm, um Zeichenfolgen in der Datei zu suchen. Hierüber erfahren Sie aber später noch einiges.

5 Erstellen von Datei-Masken

5.1 Vorbemerkungen zum Maskengenerator MASKGEN.BAS

Nach den vielen theorethischen Überlegungen nun zur Praxis. Bei allen nachfolgenden Programmen wird die oberste Speichergrenze von &9FFF nicht überschritten! Der Grund ist, daß die oberste benutzbare Speichergrenze nicht immer genau festliegt.

Sind nämlich Erweiterungs-ROMs (wie MAXAM, UTOPIA usw.) angeschlossen, dann benötigen diese Speicherplatz. Da ich nicht weiß unter welchen Bedingungen Ihr CPC betrieben wird, habe ich diese Grenze einfach so festgelegt.

Also nochmal: Die höchste Speicheradresse, die von den Dateiverwaltungsprogrammen

RAMDISK1
RAMDISK2
RANDOMD1
RANDOMD2

benutzt wird, liegt unter &A000!

Diese Obergrenze wird beispielsweise für das Programm RAMDISK1 im Maskengenerator MASKGEN.BAS festgelegt und könnte dort verändert werden. (Die maximale Obergrenze für die CPCs mit Diskettenlaufwerken wäre &A67b, wenn keine anderen Eprom-Erweiterungen angeschlossen sind.)

Wird der Wert verändert, dann muß gleichzeitig auch der in der Data-Zeile 2430 enthaltene Wert "27671" um den Betrag der Erhöhung heraufgesetzt werden.

Verwechseln Sie nun diese Hinweise auf die höchste Speicherstelle nicht mit der Grenze von HIMEM (wird durch MEMORY festgelegt). HIMEM wird in den Programmen noch viel tiefer gesetzt.

Der Bereich zwischen HIMEM und der obersten benutzbaren Speicherstelle wird für die Maschinensprache-Routinen und vor allem für die Daten verwendet.

Die Programme RAMDISK1 und RAMDISK2 könnten zwar prinzipiell mit einem Kassettenrekorder arbeiten, aber nicht nur aus Geschwindigkeits-, sondern vor allem aus Sicherheitsgründen sollte immer eine Diskettenstation eingesetzt werden.

Ich betone es ausdrücklich noch einmal:

> Ich rate dringend davon ab, bei Dateiverwaltungen mit Kassettenrekordern zu arbeiten!

Selbst wenn Sie bisher mit dem Kassettenrekorder zu Ihrem CPC nie Probleme hatten, ich verspreche Ihnen, Sie werden sie bekommen.

Dateien dieses Umfanges können nicht mit der ausreichenden Datensicherheit mittels eines Rekorders verwaltet werden. Hauptgrund hierfür sind "dropouts" also kleine Fehlerstellen im Bandmaterial, die sich so auswirken, daß es zu Leseproblemen kommt. Das Ergebnis ist: die Daten sind verloren!

Ich habe es selbst versucht, mit RAMDISK1 im Rekorderbetrieb zu arbeiten und bekam prompt Ärger! Also Finger weg vom Rekorder wenn Sie Daten verarbeiten wollen.

5.2 Erstellung der Datei-Maske für RAMDISK1

Setzen Sie den CPC durch den sogenannnten - Ihnen bestimmt bekannten - "Dreifinger-Griff" (CONTROL-SHIFT-ESC) oder durch Aus-/Einschalten zurück und starten Sie bitte den Maskengenerator, also das Programm MASKGEN.BAS.

Mit diesem wird nun die komfortable Erstellung der Datei-Maske durchgeführt. Möglich sind dabei pro "Karteikarte" bis zu 18 Felder.

Aber vor der Erstellung echter Masken sollten Sie sich Gedanken darüber gemacht haben, wieviele Felder, wieviele Zeichen pro Feld usw., für Ihre Datei erforderlich sind.

Als Beispiel nehmen wir nun an, daß 15 Felder mit den folgenden Feldbezeichnungen nötig sind:

Kunden-Nummer	(6 Zeichen)
Firma	(3Ø Zeichen)
Straße	(3Ø Zeichen)
Postleitzahl	(4 Zeichen)
Ort	(25 Zeichen)
Telefon	(16 Zeichen)
Einkauf	(2Ø Zeichen)
Technik	(2Ø Zeichen)
Artikelgruppe	(4 Zeichen)
Besuche 1	(4Ø Zeichen)
Besuche 2	(4Ø Zeichen)
nächster Besuchstermin	(8 Zeichen)
drei Felder für Bemerkungen	(je 4Ø Zeichen)

Jede "Karteikarte" belegt deshalb 363 Zeichen.

Für diese Festlegung zeige ich nun an einem kompletten Durchgang auf, was Sie zu tun haben, wenn Sie eine Maske erstellen wollen. Es ist bestimmt nicht schwer, denn sonst wäre ein Maskengenerator nicht sehr sinnvoll.

Auch diejenigen, die mit anderen Verwaltungsprogrammen dieses Buches arbeiten wollen, können diese kleine Bedienungsanleitung benutzen, denn sie ist immer gleich.

Nach dem Programmstart durch RUN "MASKGEN" meldet sich das Programm mit seinem "Titelbild", dessen grundsätzlichen Aufbau Sie in Abbildung 4 sehen.

Durch diese Darstellung fällt es Ihnen nun bestimmt leichter, der Beschreibung zu folgen. In der Titelzeile, also beim Programmnamen erscheint

"Maskengenerator (c) by L. Miedel"

und im Informationsfenster erscheint die Mitteilung

"Maskenerstellung - Fuer welches Dateiverwaltungsprogramm ?"

Etwa in Bildschirmmitte sehen Sie die Zahlen 1 bis 4 und die vier Namen der Programme, für die Masken erstellt werden können.

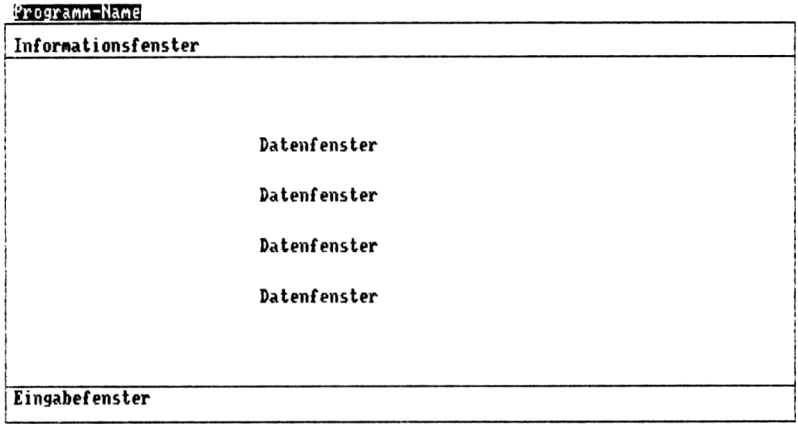


Abbildung 4: Die Einteilung des Bildschirmaufbaues beim Maskengenerator.

In der Eingabezeile erfolgt die Mitteilung:

"Bitte Ihre Eingabe (1 bis 4!)".

Die Abbildung 5 zeigt die Bildschirm-Darstellung zu diesem Zeitpunkt.

Da nun eine Maske für das erste Dateiverwaltungs-Programm erstellt werden soll, muß ganz einfach die Zifferntaste "1" gedrückt werden. Nach dieser Eingabe wird der Mittelteil mit dem Programmnamen gelöscht.

In der Informationszeile erscheint:

"Gewaehltes Dateiprogramm: RAMDISK1.BAS"

und im Eingabefenster werden Sie gefragt, ob eine bereits bestehende Maske geladen werden soll.

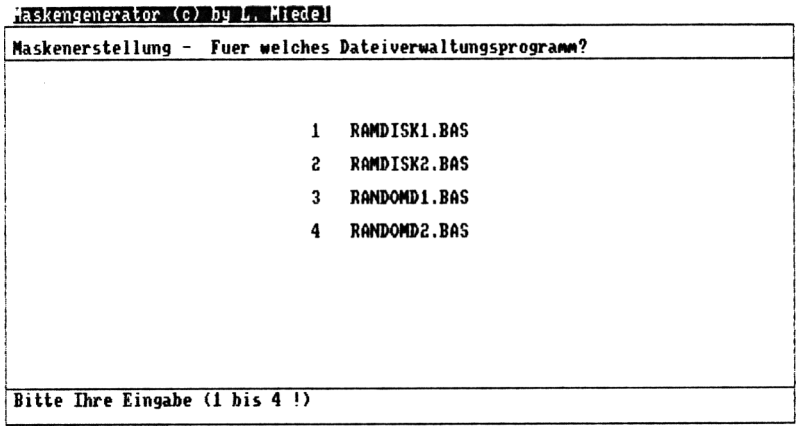


Abbildung 5: Nun muß die Wahl erfolgen, für welches Datei-Programm die Maske sein soll.

Da Sie noch keine Maske erstellt haben, kann auch keine geladen werden, deshalb beantworten Sie diese Frage mit "n" für "nein".

Als nächstes müssen Sie nun die Frage beantworten, wieviele Felder ein Dateieintrag (also eine Karteikarte) haben soll.

In der Eingabezeile erscheint:

"Wieviele Felder soll ein Eintrag haben (max 18!) ?"

Für die Demonstration haben wir insgesamt 15 Felder gewählt. Deshalb nun die Zahl 15 eingeben.

Daraufhin wird Ihnen sowohl in der obersten Zeile neben dem Programm-Namen, als auch im Datenfeld, die Anzahl der Felder angezeigt. Die gleiche Anzeige erfolgt auch im "Karteninhalt", also im eingerahmten

großen Feld (im großen Datenfenster).

Haben Sie sich davon überzeugt, daß dieser Wert korrekt ist, dann kann die Frage im Eingabefenster:

"Eingabe korrekt (j/n)"

mit "j" für ja beantwortet werden. Haben Sie sich bei der Eingabe vertan, dann kommen Sie durch Eingabe von "n" wieder zur vorherigen Frage nach der Feldanzahl.

Nachdem Sie "j" eingegeben haben, erscheint nun rechts neben dem Dateinamen der Job, also die Tätigkeit, die nun ausgeführt werden soll:

"Feldnameneingabe".

In der Informationszeile, also unterhalb des Programmnamens erscheint die Mitteilung:

"Maximale Laenge der Feldnamen 14 Zeichen."

Diese Länge kann bei der Eingabe zwar überschritten werden, übernommen werden aber nur 14 Zeichen.

Im Karteifeld (Datenfenster) sehen Sie nun eine 1, als Hinweis daß der erste Feldname eingegeben werden soll, sowie einen Hinweispfeil.

In der Eingabezeile werden Sie aufgefordert, den Feldnamen einzugeben. Nach jeder mit ENTER abgeschlossenen Eingabe wird der Name im Kartenfeld angezeigt und zur nächsten Eingabe aufgefordert. Geben Sie keinen Namen ein, so setzt das Programm die Namen mit

Feld-Nr. x

selbst. (x ist die laufende Nummer des Feldes.)

Sind Sie am Ende dieser Eingaben angelangt, dann müssen Sie die Frage beantworten, ob die Eingaben korrekt sind. Nehmen wir an, sie wären es nicht! Beantworten Sie deshalb die Frage durch Eingabe von "n"!

Daraufhin sehen Sie den Hinweispfeil beim Feldnamen mit der Nummer 1 stehen. Sie können nun durch Betätigen der ENTER-Taste die Namen solange übernehmen, bis Sie beim fehlerhaften Feldnamen angelangt sind.

Der Hinweispfeil zeigt immer auf den aktuellen Feldnamen. Nun brauchen Sie nur die Namenseingabe neu durchführen, dann wird dieser Feldname korrigiert. Falls keine weiteren Fehler mehr enthalten sind, brauchen Sie nun nur sooft die ENTER-Taste zu betätigen, bis Sie wieder gefragt werden, ob die Eingaben korrekt sind. Die bereits eingegebenen, vorherigen Feldnamen werden wieder automatisch übernommen. Sind Sie mit Ihren Eingaben zufrieden, geben Sie "j" für ja ein und das Programm fährt fort.

In der obersten Zeile wird nun die Anzahl der möglichen Einträge ausgegeben. Wenn im Programm die Buffergrenzen &3000 und &9FFF stehen, muß die Anzeige 28671 sein.

Da für das Programm RAMDISK1 diese Grenzen so festgelegt sind, muß bei Ihnen auch diese Zahl erscheinen. Sie nennt Ihnen die im Augenblick freie Anzahl der Datenbytes. Diese Anzahl von Zeichen stehen Ihnen also für die Daten zur Verfügung, denn Sie haben noch nicht festgelegt, wieviele Zeichen pro Eintrag reserviert werden.

Würden Sie jedem Feld nur ein Zeichen gönnen, dann könnten Sie 28671/15 = 1911,4 Einträge machen. Da aber nur ganze Einträge sinnvoll sind, wären es 1911 Einträge. Wenn Sie es zum Scherz einmal ausprobieren, werden Sie die Bestätigung finden. Es kann nichts dabei passieren, denn Sie können es ja wieder ändern.

Nun wird es Zeit, daß wir wirkliche Längen bei den Eintragszeilen eingeben. Betätigen Sie nur die ENTER-Taste, dann werden vom Programm automatisch 20 Stellen pro Feld vorgegeben.

Damit hätten Sie pro Karteikarte bei 15 Feldeinträgen (15*20) 300 Zeichen verbraucht. Die Kartenanzahl (=Anzahl der möglichen Einträge) wäre also 95 (Integerzahl aus 28671 dividiert durch 300).

Aber wir wollen ja andere Werte eingeben, nämlich so wie oben angegeben. Aus diesem Grund können nur 78 Karteikarten verwaltet werden. Dies sieht nun zwar nach sehr, sehr wenig aus, aber in Wirklichkeit ist es aufgrund der Tatsache, daß im Datenbereich nur Daten stehen, also ohne irgendwelche Variablenzeiger, doch sehr viel, denn jeder Eintrag umfaßt ja 363 Zeichen!

Wäre die Datei über Variable aufgebaut, dann hätten Sie wahrscheinlich, wenn alle Daten im Speicher gehalten werden sollen, auch nicht mehr mögliche Einträge. Außerdem würden Sie den großen Vorteil, auch mit anderen Programmen sehr einfach auf die Daten zugreifen zu können verspielen und auch noch andere Vorteile verlieren.

Bei Programmen mit Variablen ist nie so recht bekannt, wieviele Einträge möglich sind, bevor Ihnen der CPC

out of memory

oder

string space full

meldet.

Nun ein weiterer Punkt! Wie Sie sehen, hängt die maximale Eintragszahl von der Anzahl der benötigten Zeichen ab. Sie können also bei der Planung auch andersherum rechnen.

Als Datenspeicher stehen Ihnen 28671 Bytes zur Verfügung. Wenn Sie also 78 Karten brauchen, dann sehen Sie durch die Rechnung INT (28671/78) =367, daß Sie pro Karte 367 Bytes verwenden dürfen.

Bei 1ØØ "Kunden" wären es 286 usw. Die restlichen Bytes verschenken Sie bei dieser Technik. Rechnen Sie doch einmal nach, wieviele bei unseren Festlegungen verschenkt werden.

Für diejenigen, die geradeso oberhalb der Kapazitätsgrenze liegen, hier nochmals der kurze Hinweis:

> Die Obergrenze des Datenbereiches kann geändert werden, und zwar bis an die normale HIMEM-Grenze.

Nur aus den schon genannten, verschiedenen Gründen wurde diese obere Grenze "tiefergelegt" (Vorhandensein von EPROM-Erweiterungen, Wunsch nach Hardcopy-Programm u.ä.).

Doch wieder zurück zum laufenden Programm. Haben Sie nun alle Längenangaben gemacht, erfolgt wieder die Frage, ob alles korrekt ist.

Ähnlich wie bei der Eingabe der Feldnamen, können Sie - falls Sie es wünschen - nun wieder korrigieren (Siehe hierzu Abbildung 6).

Falls Sie aber "j" eingegeben haben, befinden Sie sich nun im Mode des Plazierens. Nun können Sie die Einträge mittels der Cursorsteuertasten auf der Karteikarte "umeinanderschieben" wie es Ihnen paßt.

Durch Betätigung der COPY-Taste wird das Feld dann auf dem Bildschirm fixiert und das nächste Feld kann plaziert werden. Etwas schwierig wird es vielleicht, wenn Sie einen Feldnamen über den anderen schieben, aber Sie schaffen es schon!

```
Maskengenerator (c) by L. Riedel Anzahl der moeglichen Eintraege:  78

 1.) Kunden-Nummer :......
 2.) Firma         :............................
 3.) Strasse       :............................
 4.) Postleitzahl  :....
 5.) Ort           :........................
 6.) Telefon       :..............
 7.) Einkauf       :..................
 8.) Technik       :..................
 9.) Artikelgruppe :....
10.) Besuche 1     :.....................................
11.) Besuche 2     :.....................................
12.) N. Besuch     :........
13.) Bemerkungen 1 :.....................................
14.) Bemerkungen 2 :.....................................
15.) Bemerkungen 3 :.....................................

Eingaben korrekt (j/n) ?
```

Abbildung 6: So sollte die Darstellung bei den angenommenen Daten sein!

Das Feld, das bewegt werden kann, wird immer invers dargestellt. Beim Bewegen wechselt die Invers- zur Normaldarstellung und dann wieder zurück. Bei vielen Feldern dauert der Bewegungsvorgang etwas, deshalb sollten Sie dann Geduld haben, denn die Maskenerstel-

lung für eine Datei geschieht ja nicht sehr oft. Aber der Aufwand für ein schnelles Maschinenprogramm erschien mir hierfür einfach zu hoch.

```
Maskengenerator (c) by L. Miedel Plazierung
Steuerung mit Cursortasten  -  fixieren =COPY     e= Ende der Plazierung

Kunden-Nummer :......          Firma          :.............................
                               Strasse        :.............................
Postleitzahl  :....            Ort            :.........................
                                              Telefon        :..............
Einkauf       :.....................      Technik        :.....................
Artikelgruppe :....
Besuche 1     :.....................................
Besuche 2     :.....................................
                                                   N. Besuch      :........
Bemerkungen 1 :.....................................
Bemerkungen 2 :.....................................
Bemerkungen 3 :.....................................
Ihre Eingaben bitte!
```

Abbildung 7: Beispiel für eine Plazierung

Wenn Ihnen die Plazierung gefällt, können Sie diese durch Betätigung der Taste "e" abbrechen, werden wiederum gefragt, ob die Eingaben korrekt sind und können - wie schon vorher auch -, wieder ändern oder die Plazierung beenden. Dann werden Ihnen noch zwei Werte mitgeteilt, nämlich wieviele Einträge mit dieser Maske möglich sind und welcher Offset von einem zum nächsten Eintrag besteht. Abbildung 7 zeigt mein Plazierungsbeispiel.

Die letzte Frage des Programmlaufes betrifft die Maskenausgabe. Geben Sie "j" ein, dann werden die Maskendaten auf Diskette gespeichert, andernfalls erfolgt nur eine Bildschirmausgabe. Damit ist dieser Programmteil abgeschlossen.

Wenn Sie die Maskendaten abgespeichert haben, können Sie diese nun bei einem neuerlichen Programmlauf einlesen, verändern, neu plazieren usw. Sie sehen, der Maskengenerator bietet trotz seiner langsamen Arbeitsweise beim Plazieren doch einigen Komfort. Ja Sie können sogar die Feldanzahl wieder ändern usw. Am besten spielen Sie etwas mit dem Programm, dann werden Sie sehen, es ist wirklich kinderleicht zu bedienen.

5.3 Erstellung der Datei-Masken für die anderen Dateiprogrammme

Falls Sie mit einem anderen Dateiverwaltungsprogramm dieses Buches arbeiten wollen, ist die Vorgehensweise ganz genauso! Nur die Daten sind anders, denn bei den anderen Programmen steht mehr Speicherplatz zur Datenspeicherung zur Verfügung.

Da die Daten hierfür im Programm des Maskengenrators stehen, haben Sie nichts zusätzlich zu tun, sondern auch nur das Fragen- und Antwortenspiel mit dem CPC durchzuführen.

Allerdings ist es für Besitzer des CPC 464 oder 664 sinnlos mit dem Programm RAMDISK2 arbeiten zu wollen, denn diesen fehlen die zusätzlichen 64 KByte Speicherplatz. Dagegen kann mit diesen beiden Computern selbstverständlich mit den Dateien für den Disketten-Random-Betrieb gearbeitet werden.

6 Mit den Programmen arbeiten

6.1 Arbeiten mit RAMDISK1

Um mit den Dateiverwaltungsprogrammen RAMDISK oder RANDOMD arbeiten zu können, muß bereits eine Dateimaske existieren. Selbst ein "Probelauf" ist ohne Maske nicht möglich!

Weitere Voraussetzungen sind, daß sich das Basic-Programm und das Maschinenprogramm auf der Diskette befinden.

Für RAMDISK1 gilt, daß Sie mit 2 Disketten arbeiten. Einmal die Programm-Diskette auf der sich die Programme:

RAMDISK1.BAS
RAMDISK1.MPG

befinden müssen.

Auf der Datendiskette muß

RAMDISK1.MSK

stehen.

Daten- und Programmdiskette können natürlich in einer einzigen vereinigt sein. Am besten die Seite A als "Programm-" und die Seite B als "Datendiskette" benutzen.

Den Teil RAMDISK1.MPG müssen Sie durch den Ablauf des Programmes RAMDISK1.HEX erzeugt haben. Der Teil RAMDISK1.MSK wird durch den Maskengenerator auf die Diskette geschrieben. Diese Programme finden Sie - genauso wie den Basic-Teil RAMDISK1.BAS - im Kapitel der Programmbeschreibungen.

Sind alle Teile vorhanden, dann starten Sie bitte das Programm RAMDISK1.BAS! Nach kurzer Zeit meldet sich das Programm und teilt mit dem Hinweis

"Lade das Maschinenprogramm"

mit, daß nun der Maschinensprache-Teil geladen wird.

Kurz darauf erscheint die Meldung:

"Lese die Maskendaten".

Am Ende des Ladevorganges sehen Sie ein Bild ähnlich dem des Plazierens der Feldnamen beim Maskengenerator, denn so wollten Sie das Aussehen Ihrer elektronischen Karteikarten haben. Nun können Sie bereits mit dem Programm arbeiten.

Sinnvollerweise erfolgen nun erst einmal Eingaben.

Bevor Sie aber die erste Eingabe machen, ein ganz wichtiger Hinweis!

> Das Programm hat in der Zeile 190 die Variable "basis". Sie dient dazu, daß im Falle von vorliegenden Kundennummern, über die der Zugriff erfolgen soll, die Startadresse festgelegt werden kann.

Im Originalprogramm hat die Variable "basis" den Wert 1. Dies bedeutet, daß der erste Eintrag auch die Nummer 1 hat. Wenn nun aber ein Kundenstamm vorliegt, dessen erster Kunde die Nummer 134500 und der nächste die Nummer 134501 haben soll usw., dann kann der Variablen "basis" der Grundwert 134500 zugeordnet werden. Damit ist es auch möglich (mit gewissen Handicaps), Dateien zu verarbeiten, die nicht mehr auf einmal in den Speicher des CPC passen. Man teilt die Daten dann auf mehrere Files auf und baut ggf. eine kleine Eingabe-Routine für die Variable "basis" ein. Beim Programm RAMDISK1.BAS sollte diese Routine nicht zu umfangreich sein, denn es ist nur noch sehr wenig Speicherplatz frei!

Als Suchkriterium und auch als korrekter Offset für die "Kunden-Karten" dient die Eintragsnummer. Haben die Kundennummern mehr als sechs Stellen, dann muß im

jeweiligen Basic-Teil und natürlich auch in der richtigen Programmzeile die Zuweisung .. LEFT$(enr$,6) .. - und zwar der Wert 6 - geändert werden.

Falls nun die "basis" den Wert 134500 hat, dann hat bei den angenommenen Daten (78 Einträge mit je 363 Bytes) die höchste "Kunden-Nummer" den Wert 134577. Hat "basis" den Wert 82, dann ist die niedrigste Nummer eben 82 und die höchste 159. Ist doch ganz einfach, oder?

Nach dem Programmstart meldet sich das Programm auch mit der Eintragsnummer, die dem Wert in "basis" entspricht. Die weitere Bedienung des Programmes ist ein Kinderspiel.

Da nach dem Start alle Inhalte der Datenbytes auf den "fuellwert" gesetzt werden (fchar und Aufruf von filler), vorgesehen ist der Punkt, müssen durch den Befehl "L" (LOAD) gegebenenfalls erst die Daten geladen werden. Vorausgesetzt, es sind bereits welche vorhanden.

Wir müssen aber erst welche eingeben. Doch vorher sollen Sie erst einmal alle Befehle zur Programm-Bedienung kennenlernen.

6.1.1 Die Bedienungsbefehle für die Datei-Programme

L (LOAD) - Laden des Datenfiles
S (SAVE) - Speichern des Datenfiles
F (FIND) - Suchen nach einer Zeichenfolge
G (GOTO) - Gehe zu einer noch einzugebenden Nummer
E (ENDE) - Beenden des Programmes

Außerdem können auch die Cursor-Steuertasten zur Bedienung benutzt werden.

Cursor links - Eine Eintragnummer tiefer
Cursor rechts - Eine Eintragsnummer höher
Cursor hoch - Zehn Eintragsnummern höher
Cursor tief - Zehn Eintragsnummern tiefer

Alle Eingaben werden durch die COPY-Taste eingeleitet und können damit (vorzeitig) auch wieder beendet werden.

Ansonsten wird jede Eingabe mit der ENTER- oder RETURN-Taste abgeschlossen.

Mehr Funktionen gibt es zur Verwaltung der Daten nicht!

Einfacher in der Bedienung kann ein Dateiverwaltungsprogramm doch kaum sein! Die paar wenigen Befehle kann man sich sehr leicht merken, deshalb ist auch keinerlei Anzeige dieser Funktionen vorgesehen.

Falls Sie noch keine Daten eingegeben haben, drücken Sie nun bitte die COPY-Taste und füllen Sie das "Karten-Formular" aus. Dadurch ergibt sich dann ein Bild, das wie Abbildung 8 aussehen kann.

Bei den Eingaben ist zu berücksichtigen, daß die DEL-Taste nicht arbeitet. Falsche Eingaben werden deshalb einfach nur überschrieben. Um an die gewünschte Position zum Überschreiben zu kommen, braucht nur mit den Cursor-Steuertasten "hingefahren" zu werden.

```
Dateiverwaltung mit 64K-D1S8                    Dateieintraege:  78
Bitte Ihre Eingaben!
Kunden-Nummer :000001          Firma        :Bauer Heinrich.............
                               Strasse      :Wiesenstrasse 12...........
Postleitzahl  :9023            Ort          :Rummersdorf 23.............
                                            Telefon      :09831/1004.....
Einkauf       :Frau Schroeder.....    Technik      :Drommer Friedrich..
Artikelgruppe :0078
Besuche 1     :Jan 86/Mar 86/Sept 86/.................
Besuche 2     :.......................................
                                                 N. Besuch     :Nov 1986
Bemerkungen 1 :evtl. Interesse an Artikelgruppe B0087/5
Bemerkungen 2 :muss im November abgeklaert werden!.....
Bemerkungen 3 :Hierzu Katalog B008 mitnehmen!..........
Aendern                                                  1
```

Abbildung 8: Eingaben sind gleichbedeutend mit Änderungen.

Im Informationsfeld werden Sie zu Ihren Eingaben aufgefordert, im Eingabefeld - das nun nicht zur Eingabe dient - sehen Sie invertiert dargestellt, die Mitteilung "Aendern" und rechts daneben die Eintragsnummer. Die Eingaben selbst werden an den durch den Maskengeneratorlauf festgelegten Plätzen gemacht.

Wenn Sie sortieren oder ausdrucken wollen, so geht das auch, aber das machen dann andere Programme. Wie das geht erfahren Sie im Kapitel der Zusatzprogramme. Denn im Augenblick arbeiten wir mit einem "Datenverwalter", also mit einem Programm, mit dem nur "verwaltet" wird.

Nach der Eingabe können Sie mittels der Cursortaste "rechts" zum nächsten Eintrag gehen. Wollen Sie aber direkt eine Eintragsnummer anwählen, so geschieht dies durch Eingabe von "g" für GOTO und anschließender Eingabe der Nummer. Abbildung 9 zeigt die Bildschirmdarstellung.

```
Dateiverwaltung mit RAM-DISK                    Dateieintraege:  78
GOTO ...

Kunden-Nummer :000070          Firma          :Hintermaier Rudolf

                               Strasse        :Rosenstrasse 14

Postleitzahl  :8000            Ort            :Muenchen 93
                                                Telefon        :089/123456

Einkauf       :Friedr. Rupp                Technik        :Emil Kaiser

Artikelgruppe :????

Besuche 1     :Erstbesuch geplant
Besuche 2     :

                                                N. Besuch      :Jan 87
Bemerkungen 1 :Besuchsgrund: laufend techn. Anfragen
Bemerkungen 2 :wegen Produktlinie COSMOS
Bemerkungen 3 :Noch kein Kunde !!

Eintragsnummer? 12
```

Abbildung 9: Einfache Auswahl durch GOTO

Sie werden sehr schnell mit dieser Technik vertraut sein. Doch es gibt ja noch mehr Funktionen, die noch nicht besprochen sind. Sie können Einträge auch suchen lassen. Hierzu geben Sie den Befehl "f" (FIND) ein. Die Abbildung 1Ø zeigt, wie Sie nach bestimmten Zeichenfolgen suchen lassen können.

Zur Suche sollten Sie aber noch einiges wissen. Gesucht wird immer ab der nächsten "Karte", denn die augenblicklich "Aktuelle" haben Sie ja vor Augen. Das bedeutet, daß Sie dann, wenn Sie die gesamte Datei durchsuchen lassen wollen, zuerst auf die niedrigste Eintragsnummer zu positionieren haben.

Der Sinn dieser Suchtechnik ist, daß Sie alle Einträge bei denen Übereinstimmung vorliegt auch finden können. Denn würde der Suchvorgang immer beim ersten Eintrag beginnen, dann könnte auch immer nur die erste Übereinstimmung gefunden werden.

Weitere Übereinstimmungen, die in anderen Einträgen enthalten wären, kämen deshalb nicht auf den Bildschirm. Das Suchprogramm startet immer ab dem Ende des aktuellen - also auf dem Bildschirm sichtbaren - Eintrages.

Das Suchprogramm sucht aber nicht "feldmäßig", sondern "eintragsmäßig" und das bedeutet, daß dann, wenn der Suchbegriff beispielsweise "geweg" lauten würde, auch ein Eintrag ausfindig gemacht wird, der so aussieht wie es Abbildung 9 zeigt. Die Folge stimmt ja überein!

Wenn Sie sich diese Abbildung ansehen, wird auch klar warum: In der Spalte Bemerkungen 1 sind die letzten beiden Buchstaben "ge" und in Bemerkungen 2 die ersten drei Zeichen "weg" und deshalb muß dieser Eintrag bei der angenommenen Suchfolge gefunden werden.

Falls der Suchstring aber - z. Beispiel durch Groß-/Kleinschrift-Unterschiede - nicht exakt übereinstimmt, wird das Programm nicht fündig! In diesem Fall verhält sich das Programm so, als ob ein Fehler aufgetreten wäre.

Sind Sie sich über die genaue Schreibweise nicht im Klaren, dann geben Sie deshalb nur das ein, was Sie als definitiv richtig kennen, dann klappt die Sache.

Diese Fehlermeldung, deren Aussehen die Abbildung 11 zeigt, tritt beispielsweise auch dann auf, wenn Sie die maximal zulässige Eintragsnummer über- oder die minimale unterschritten haben. Allerdings brauchen Sie bei den Eingaben keine Befürchtungen zu haben, daß Ihnen durch falsche Angaben das Programm aussteigt oder ähnliches, es wird dann lediglich entweder auf die niedrigste oder auf die höchste Eintragsnummer positioniert!

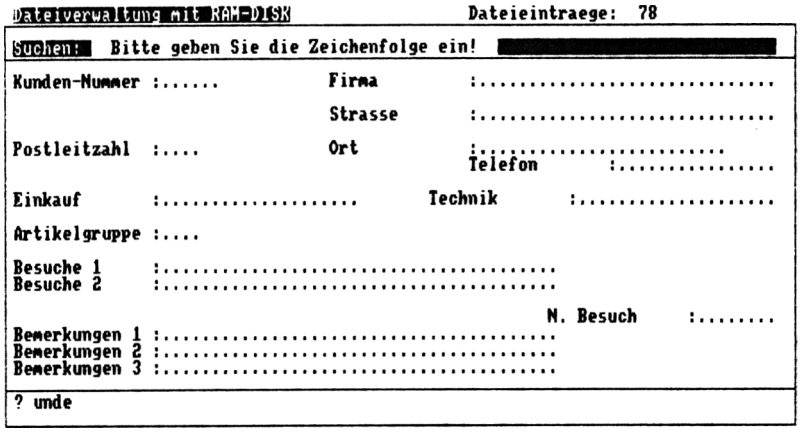


Abbildung 10: Komfortables Suchen nach Zeichenfolgen.

Sie werden bei beiden Programmen für den RAM-Disk-Betrieb erstaunt sein, mit welcher Geschwindigkeit die Daten geholt und auf den Bildschirm gebracht werden. So schnell geht es eben nur in Maschinensprache!

Diese hohe Geschwindigkeit gilt auch für die Suchvorgänge. Falls Sie aber den gesamten Datenbereich beispielsweise mit der Buchstabenkombination "abcdef" beschrieben haben und nur in den letzten Speicherstellen die Folge "abcdex" enthalten ist und Sie auch noch nach dieser suchen wollen, dann dauert es halt ein

paar Sekunden, bis das Suchprogramm bei dieser Art von "Veräppelung" alle "Fehleinträge" untersucht und wieder verworfen hat.

Ein paar Worte noch zu den Punkten, die Sie bei leeren Karten sehen. Um nicht auch noch den Aufwand für "Marker", die den Beginn und das Ende der Feldmaske anzeigen programmieren zu müssen, wurde diese Art von Darstellung gewählt.

Sie können selbstverständlich das Füllzeichen ändern und hätten beispielsweise durch den Füllzeichenwert 32 (&2Ø) lauter "Spaces" also Leerzeichen.

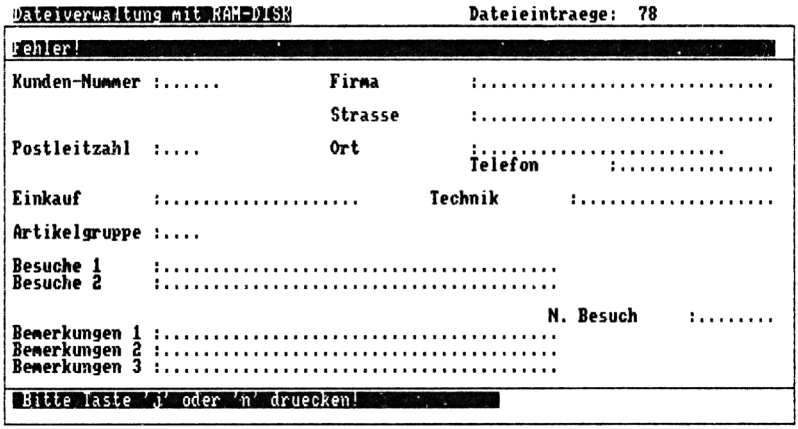


Abbildung 11: Anzeige eines aufgetretenen Fehlers oder auch, daß das Suchprogramm nicht "fündig" wurde.

Andererseits würden Sie dann aber keine Grenzen erkennen. Da aber bei Benutzung einer Datei ja auch Einträge gemacht werden, können Sie bei jeder Eingabe diese Punkte "wegradieren" wenn Sie diese nicht wünschen.

Die Punkte können aber auch noch einem anderen Zweck dienen.

Normalerweise beginnt kein gültiger Eintrag mit einem Punkt. Steht ein solcher jedoch am Anfang eines Feldes, dann sollte davon ausgegangen werden, daß dieses Feld leer ist. Diese Tatsache kann dann bei anderen Programmen ausgenutzt werden.

Ein Sortierprogramm beispielsweise kann dadurch noch schneller werden, indem es solche Einträge "verwirft", also gar nicht berücksichtigt. Sie sehen, auch Satzzeichen können mehrere Zwecke erfüllen.

6.2 Arbeiten mit RAMDISK2
(Nur für CPC 6128)

Um mit diesem Programm arbeiten zu können, müssen folgende Programmteile auf der Diskette vorhanden sein:

RAMDISK2.BAS
RAMDISK2.MPG
RAMDISK2.MSK

Der Teil RAMDISK2.MPG muß mit dem Hexlader-Programm RAMDISK2.HEX erzeugt worden sein. Die Dateimaske RAMDISK2.MSK wird beim Programmlauf des Maskengenerators geschrieben.

Bevor Sie aber starten können, sollten Sie die Programm-Bedienung von RAMDISK1 gelesen haben, da diese in etwa der Bedienung von RAMDISK2 entspricht. Das heißt, dort finden Sie die prinzipiellen Grundzüge der Vorgehensweise. Das Programm unterscheidet sich im Wesentlichen nur durch die Ausnutzung der zusätzlichen Speicherblöcke des CPC.

Die Abbildungen bei RAMDISK1 sind natürlich nicht ganz mit dem identisch, was Sie auf dem Bildschirm sehen, aber die Unterschiede sind so gering, daß es Ihnen bestimmt nichts ausmacht.

Die beiden Basiclistings unterscheiden sich aber stark, denn durch den begrenzten Speicherplatz beim CPC 464 und 664 konnte das Programm RAMDISK1.BAS nicht so übersichtlich programmiert und dargestellt werden.

Beim Programm RAMDISK2 steht Ihnen also auch für das Basic-Programm selbst mehr Platz zur Verfügung, da die HIMEM-Grenze mit &3FFF festgelegt wird und die Maschinenprogramme ab &8000 stehen werden. Da insgesamt fünf Blöcke a 16 KByte als Datenspeicher dienen, stehen Ihnen bei diesem Programm insgesamt 80 KByte Datenspeicher zur Verfügung! 16 KByte mehr als beim Einsatz von BANKMAN!

Auch für dieses Programm gilt, daß Sie eine Programm- und eine Datendiskette haben. Diese können ebenso wie bei RAMDISK1 die Vorder- und Rückseite einer Diskette sein.

Auf der Programmdiskette müssen die Programme (Basic-

und Maschinenprogramm) stehen und auf der Datendiskette die Daten (also auch die Daten der Maske!).

Starten Sie den Programmteil RAMDISK2.BAS! Bedingt durch die größere Datenmenge haben Sie längere Lade- und Speicherzeiten beim Einlesen der Datenfiles als beim Programm RAMDISK1, es hat aber auch noch andere Konsequenzen. Der Platz auf einer AMSDOS-Diskette ist sehr gering. Deshalb werden die Datenfiles mit der Extension .DAx gekennzeichnet.

Diese Art von Kennzeichnung bewirkt, daß beim Überschreiben dieser Files (z.B. durch mehrmaliges Abspeichern hintereinander) nur ein einziges File mit der Extension .BAK entsteht. Dadurch kann bei diesen Vorgängen keine "Diskette voll"-Meldung erscheinen.

Ein anderer Punkt ist auch noch wichtig. Sie können jederzeit die Diskette wechseln. Wenn Sie also Sicherheitskopien haben wollen, oder die Datenfiles für andere Programme duplizieren wollen, so ist dies jederzeit durch den Aufruf von "s" möglich. Auf der Datendiskette muß nur ausreichend Platz sein. So kann, wenn Sie nach dem "Großvater-Vater-Sohn"-Prinzip arbeiten, eigentlich überhaupt nichts mehr schief laufen. Datenverluste sind fast ausgeschlossen.

Files mit der Extension .BAK sollten immer so schnell wie möglich gelöscht werden, damit Sie beispielsweise für Sortierfiles Platz haben. Wie Löschungen erfolgen können, steht im Handbuch. Derartige Löschvorgänge können auch "programm-gesteuert" sein. Wer will kann deshalb nach den für das Abspeichern verantwortlichen Zeilen noch den Befehl zum Löschen nicht mehr benötigter Files unterbringen.

Das Programm arbeitet mit einem Laufwerk, kann aber ganz schnell so geändert werden, daß es mit zwei Laufwerken arbeitet. Dann kann beispielsweise im Laufwerk 1 immer die Programmdiskette und in Laufwerk 2 die Datendiskette stecken. Wie Sie es eben wünschen.

Ebenso kann als Zweitlaufwerk auch z.B. ein VORTEX-X-Laufwerk eingesetzt werden, da dann auch mehr Diskettenkapazität zur Verfügung steht, muß vor den Abspeichervorgängen dann nur selten etwas gelöscht werden. FÜr ausreichende Datensicherheit insofern, daß Sie Sicherheitsduplikate für die Daten anlegen, haben Sie

in diesem Falle dann auch selbst Sorge zu tragen. Ein mehrmaliges Speichern von Datenfiles, auch wenn dies unter anderen File-Namen erfolgen sollte, ist keine Sicherheitskopie!

Falls das Programm einmal - wegen zu voller Datendiskette - abbrechen sollte, sind Ihre zwischenzeitlich geänderten Daten nicht verloren. Es wird dann zwar die Bildmaske zerstört, aber sonst ist nichts passiert. Eine Diskette mit ausreichender Kapazität einlegen und weitermachen (CONT) bzw. die Zeile 2020 durch GOTO aufrufen.

Man kann eigentlich überhaupt nichts verkehrt machen, denn trotz der hohen Leistungsfähigkeit dieser Programme sind sie sehr einfach zu handhaben.

Wie Sie Daten ausdrucken und sortieren lassen können, ist im Kapitel der Zusatz- und Hilfsprogramme nachzulesen.

6.3 Arbeiten mit RANDOMD1

Auch für dieses Programm gilt, daß Sie sich die prinzipielle Arbeitsweise und vor allem die Bedienungsbefehle erst einmal durchlesen sollten. Zu finden ist es am Anfang dieses Kapitels. Lesen Sie deshalb bitte zuerst einmal den Teil unter 6.1 (Arbeiten mit RAMDISK1).

Folgende Programmteile müssen auf der Diskette vorhanden sein:

RANDOMD1.BAS
RANDOMD1.MPG
RANDOMD1.MSK.

Denken Sie auch hier daran: Daten- und Programmdiskette! Der Maschinensprache-Teil wird mit dem Programm RDKPL.HEX erzeugt.

Außerdem wird, um etwas mehr Kapazität zu gewinnen, bei den Disketten ein "Spezialformat" benötigt. Dieses Format kann mit dem Programm SPEZFORM.BAS erzeugt werden.

Sie arbeiten also mit zwei Disketten; der Programmdiskette und der Datendiskette. Bei den Schneider-Disketten kann es auch so gehandhabt werden, auf der Vorderseite die Programme und auf der Rückseite die Daten zu haben.

Als Programm sollten Sie RANDOMD1.BAS starten und eine speziell vorbereitete Datendiskette bereithalten. Auch dieses Programm kann so umgeschrieben werden, daß immer dann, wenn auf Daten zugegriffen wird, die Umschaltung auf ein zweites Laufwerk erfolgt.

Das Programm sagt Ihnen nach dem Start, was Sie zu tun haben! Alles andere ist wie gehabt.

Daß Ihnen die Befehle "l" für LOAD und "s" für SAVE nicht zur Verfügung stehen, werden Sie nicht nur bemerken, sondern Sie werden auch wissen warum: Geladen bzw. gespeichert wird immer nur dann, wenn ein Datenhandling zu erfolgen hat und dies mit direktem Zugriff auf einzelne Sektoren der Datendiskette.

Dies ist beispielsweise immer dann der Fall, wenn zu einer bestimmten Eintragsnummer gegangen werden soll. Dann werden die Daten in den Speicher geholt. Also auch wenn Sie mit Cursorsteuerung arbeiten wird für jeden Tastendruck ein Einlese-Vorgang ausgelöst.

Bei Änderungen ist es ebenso, nach der Änderung wird sofort auf die Diskette zurückgeschrieben.

Das Suchen dauert nun - bedingt dadurch, daß jeder Eintrag erst eingelesen werden muß - um einiges länger als bei reinem RAM-Disk-Betrieb. Aber ansonsten wird das Programm genauso "fündig" wie bei den Programmen RAMDISK.

Bei dieser Version habe ich die Offset-Möglichkeit durch "basis" nicht einprogrammiert, wer diese Funktion braucht, kann bei RANDOMD2.BAS nachsehen wie es geht und das Programm entsprechend abändern. Es ist hoffentlich nicht zuviel verlangt. Aber ich hatte es an anderer Stelle schon einmal erwähnt: Dieses Buch ist kein Dateiverwaltungsprogramm, sondern ein Anleitungsbuch, wie Sie eigene Dateiverwaltungsprogramme schreiben können.

Deshalb sind auch viele Dinge in den Musterprogrammen nicht programmiert worden. So kann zum Beispiel jederzeit mit der ESC-Taste abgebrochen werden, eine Sache die ich bei einem Arbeitsprogramm auf jeden Fall verhindern würde!

Die Ausgaben von Disketten-Fehlermeldungen im Dateifenster sollten auf alle Fälle verhindert werden. Überhaupt sollten alle auftretbaren Fehler über eine Fehler-Routine abgefangen werden. All das sind Punkte, die Sie dann durchaus noch selbst einprogrammieren können und dürfen. Das "Schönheits-Empfinden" verschiedener Menschen ist unterschiedlich. Betrachten Sie deshalb die ganzen Programme nur als Vorschläge. Ich gehe auch davon aus, denn sonst hätten Sie ja ein Datei-Verwaltungsprogramm erstanden und nicht dieses Buch.

Auch die zu diesem Buch erhältliche Diskette ist als reine Abtipphilfe gedacht, damit vor allem weniger programmiererfahrene CPC-Besitzer nicht für alle Zeiten nach Abtippfehlern suchen müssen.

6.4 Arbeiten mit RANDOMD2

Wenn Sie bisher noch mit keinem der anderen Dateiverwaltungsprogramme RAMDISK oder RANDOMD1 garbeitet haben, dann sollten Sie erst einmal die grundsätzliche Bedienung unter 6.1 nachlesen.

Haben Sie bereits mit RANDOMD1 gearbeitet, dann wissen Sie, daß bei diesem eine speziell formatierte Diskette benutzt wird.

Das soll nun auch geschehen. Allerdings noch weitergehend als bei RANDOMD1. Dort haben wir ja einen Sektor vor den Daten verschont. Bei diesem Programm wollen wir nun das maximale, was mit normalen Mitteln an Speicherkapazität "rauszuholen" ist, auch benutzen: Die gesamte Diskette dient als Datenspeicher. Es steht Ihnen also nicht einmal mehr ein Inhaltsverzeichnis zur Verfügung, sondern überall werden auf der Diskette nur Daten stehen.

Doch dazu brauchen wir eine speziell vorbereitete Diskette!

Wer sich mit dem Programm RANDOMD1 schon beschäftigt hat, der weiß wahrscheinlich auch, warum die Daten-Disketten speziell präpariert werden.

Wer es noch nicht weiß, erfährt es hier. Bei einem Formatiervorgang werden die Disketten in Spuren und Sektoren eingeteilt. Gleichzeitig werden alle Inhalte der Sektoren mit dem Fillerbyte beschrieben. Standardmäßig ist der Wert des Fillerbytes &E5. (Dieser Wert ist u.a. auch gleichzeitig die Kennzeichnung für gelöschte Files in den Directory-Einträgen.)

Da dann, wenn noch keinerlei Einträge in die Datei gemacht wurden, alle nutzbaren Bytes der Diskette den Wert &E5 haben, würde auf dem Bildschirm immer ein Wald von PIK-Bildchen (ein kleines ausfülltes Blatt) zu sehen sein. Da dies für Eingaben nicht gerade professionell aussieht, habe ich einfach bestimmt, daß die Disketten mit dem Fillerbyte für den Punkt formatiert werden. Das ist der Grund für den speziellen Formatiervorgang. Wen es nicht stört, daß in den Einträgen zu Beginn immer ein "Laubwald" steht, der kann den Spezial-Formatier-Vorgang auch unterbleiben lassen.

Wie bei RANDOMD1 fehlen auch bei diesem Programm die beiden Befehle für das Laden oder Speichern. Sie werden ja nicht benötigt.

Gleichzeitig gilt auch, daß die "F"-Funktion also das Finden von ASCII-Folgen eben länger dauert, da jeder Eintrag erst gelesen werden muß.

Doch nun zu einer weiteren Besonderheit. Um Ihnen möglichst viele Variationen bei den Programmen zur Datenverarbeitung zu zeigen, habe ich bei diesem Programm folgende Voraussetzungen angenommen:

Das Erstlaufwerk ist ein 3-Zoll-Laufwerk. Auf der dort eingelegten Diskette stehen die Programme und die Masken-Daten.

Die Files

RANDOMD2.BAS
RANDOMD2.MPG
RANDOMD2.MSK

müssen also auf der 3-Zoll-Diskette stehen. Als Laufwerk 2 steht ein VORTEX-X-Modul-Laufwerk zu Verfügung. Gearbeitet wird nicht unter AMSDOS, sondern unter VDOS!

Die Daten stehen auf der Diskette im VORTEX-Laufwerke, bzw. werden dorthin geschrieben. Das bedeutet für die Praxis aber, daß die Datendiskette besonders auffällig gekennzeichnet werden muß und daß die Datensicherung auf völlig andere Art zu geschehen hat als bisher.

Hierzu brauchen Sie ein Programm, das Ihre Daten-Diskette dupliziert! Und nun tritt ein kleines Problem auf: Ich weiß nicht welchen CPC Sie haben. Deshalb kann ich nur die Minimalkonfiguration annehmen. Sie haben einen CPC 464!!

Scherz beiseite, ich muß eben annehmen, daß Sie nur soviel Speicherplatz zur Verfügung haben, als hätten Sie einen CPC 464. Kopieren dauert dementsprechend lange und ist mit vielen Diskettenwechseln verbunden.

Der CPC 464 oder 664 bremst mit seinem geringem Speicherplatz das Anlegen von Sicherheitskopien und deshalb dauert der Kopiervorgang Ihrer Datendiskette et-

was. Sie werden auch immer wieder zum Diskettenwechsel aufgefordert.

Wer ein Programm hat, das in der Lage ist, alle Spuren und Sektoren einer Diskette zu lesen und dann auf einer zweiten zu schreiben kann dieses für die Duplizierung einsetzen. Wer keines hat, kann das Hilfsprogramm DUPLI1.BAS dieses Buches verwenden.

Doch nicht nur die 464-/664-Besitzer sollen Sicherheitsdisketten anlegen und deshalb ist noch ein weiteres Programm zum Anlegen von Kopien im Buch: DUPLI2.BAS. CPC 6128-Besitzer werden damit bestimmt große Freude haben.

Ansonsten ist das Datei-Programm im großen und ganzen dem Programm RANDOMD1 entsprechend. Eine große Ausnahme aber gibt es. Als Maschinenprogramm findet die Routine DISC2 Verwendung.

Die Zugriffe auf die Diskettenstation sind deshalb im Basiclisting um einiges anders.

7 Beschreibung der Programme

7.1 Vorbemerkungen zu den Programmbeschreibungen

Wegen des Umfanges der gesamten Programme kann für die Listings keine komplette Beschreibung jedes einzelnen in den Programmen verwendeten Befehles oder meiner Gedankengänge erfolgen, die mich zu diesm Lösungsweg veranlaßt haben. Dies würde den Rahmen dieses Buches sprengen. Im Vordergrund stand aber immer, alle Programme so zu schreiben, daß es für Sie nachvollziehbar ist, auch wenn dadurch manchmal alles etwas umständlicher (dafür aber übersichtlicher) programmiert werden mußte.

Trotzdem will ich versuchen, Ihnen für alle Programme wenigstens die wichtigsten Programmzeilen und Routinen zu erläutern, damit Sie - für evtl. eigene Anpassungen - das jeweilige Programm verstehen und den Teil, in dem die Änderungen zu erfolgen haben, schneller auffinden können.

7.2 Programm MASKGEN.BAS

Die Zeilen bis 230 sind reine Bemerkungszeilen und haben mit dem eigentlichen Programmlauf nichts zu tun.

Die Zeile 240 dient dazu, die Ein-Ausgabe-Buffer für das Lesen und Schreiben der Datenfiles bereits zu Beginn vom CPC einrichten zu lassen. Außerdem wird der 80-Zeichen-MODE festgelegt.

Die Zeilen 270 und 280 legen die verschiedenen Fenster fest.

Die Zeilen 310 und 320 zeichnen den Rahmen der "Karteikarte" und dienen in diesem Programm gleich als Arbeitsmaske.

Da maximal 18 Felder definiert werden sollten, wird in Zeile 350 dimensioniert. Die Dimensionierung erfolgt um einen Wert höher, als eigentlich erforderlich. Dies hat nur programmtechnische Gründe. Im Prinzip wäre die Dimensionierung mit 17 ausreichend, aber dann müßten Positionsangaben für den LOCATE-Befehl immer mit +1 versehen werden.

Die Zeilen 370 bis 480 legen bestimmte Variable vorab fest, damit Änderungen einfacher durchgeführt werden können. Gleichzeitig werden in diesen Zeilen die "Dummy"-Feldnamen (fenam$(i)) und "Dummy"-Feldlängen (feldlen(i)) festgelegt.

Die Zeilen 520 bis 640 dienen zur Auswahl, für welches Dateiprogramm die Maske erstellt werden soll. Außerdem werden in Zeile 640 die Arrays für die nicht dimensionierten Variablen x1$, x2$, usw. gelöscht.

Die Zeilen 660 und 670 sind dafür zuständig, ob eine bereits bestehende Maske eingelesen wird, oder nicht. Die Entscheidung hängt von Ihrer Eingabe ab. Falls eingelesen werden soll, dann werden durch die Zeilen 690 bis 830 die Maskendaten eingelesen und die Anzahl der Felder ausgegeben.

Danach wird durch die Zeilen 850 bis 880 abgefragt, ob nur die Plazierung oder ob andere Änderungen durchgeführt werden sollen. Wird nur die Plazierung ausgewählt, dann geht es bei 1580 weiter.

Bei Änderungen wird durch die Zeilen 900 und 910 abgefragt ob die Feldanzahl verändert werden soll. Falls ja, wird nach Zeile 1010 verzweigt. Falls nein, erfolgt durch die Routine ab 930 bis 980 die Eingabe der Feldnamen.

Die Zeilen 1010 bis 1090 sind für die Eingabe der Feldanzahl zuständig. Die erlaubten Grenzen werden überprüft und ungültige Eingaben unterdrückt.

Die Zeilen 1120 bis 1250 sind die Routine für die Eingabe der Feldnamen. Dabei wird mittels Schablonenausgabe durch PRINT USING gearbeitet. Zu lange Namen werden durch 1190 abgeschnitten. Zu kurze werden durch 1200 auf 14 Zeichen aufgefüllt. In 1210 wird dem Feldnamen ein Doppelpunkt angehängt.

Durch Zeile 122Ø erfolgt die formatierte Ausgabe. Die Feldnamen werden in das Array von fenam$(i) übernommen. Nach der Übernahme wird der Zähler hochgezählt (Zeile 123Ø), um dann durch Zeile 124Ø die maximale Eingabe-Nummer zu prüfen.

Nachdem die Feldnamen festliegen, werden von 128Ø bis 143Ø die Längen für jeden Feldnamen-Eintrag eingegeben. Diese Eingaben werden durch 134Ø bis 137Ø geprüft. Eine Laenge null wird ignoriert und entweder die vorbestimmte (Zeile 46Ø), oder - falls vorhanden die bereits eingelesene Länge übernommen. Die aktuelle Zeile wird durch einen inversen Balken dargestellt (Zeile 13ØØ).

Zur optisch guten Darstellung wird durch Zeile 139Ø dann eine Punktreihe ausgegeben, die der Länge des Feldnamen-Eintrags entspricht.

Die Zeile 138Ø springt das Unterprogramm an, das die Anzahl der Zeichenlängen "aufaddiert".

Die Variable "ben" in Zeile 146Ø enthält die bisher benutzten Längen der Einträge, um dann in Zeile 147Ø dem Offsetwert (off) zugewiesen zu werden.

Fehleingaben werden in den Programmen oft mehrfach verhindert, um kleine Fehler, die beim Abschreiben des Programmes geschehen sein könnten "auszubügeln". Dazu dient beispielsweise die Zeile 148Ø.

In Zeile 149Ø wird für den jeweils augenblicklichen Stand der Eingaben die Anzahl der möglichen Datei-Einträge berechnet. Der echte Wert steht erst am Schluß der Längen-Eingaben fest.

Die kontinuierliche Ausgabe soll laufend einen Überblick bei den Eingaben ermöglichen. Diese Information wird durch Zeile 15ØØ ausgegeben. Die Variable "baenke" ist erforderlich, weil beim CPC 6128 - beim RAM-Disk-Programm - insgesamt fünf Bänke (Blöcke) mit je 16 KByte zur Verfügung stehen.

Die Zeilen 153Ø bis 155Ø erweitern die Länge der Feldnamen um die Anzahl der zugehörigen Feldlängen. Die Feldnamen werden mit dem Punktzeichen "." erweitert.

Ab 157Ø beginnt die Routine zur Plazierung der Feldnamen mit den zugehörigen Längen.

Da die Positionen über die Variablen für die Horizontalposition xpos (i) und die Vertikalposition yposi (i) festliegen, erfolgt die Ausgabe sowohl über die Subroutine ab 236Ø als auch durch Zeile 164Ø. Dadurch erfolgt ein langsames, kurzes "Aufblinken" der Feldeinträge.

Die Zeilen 173Ø bis 183Ø dienen zur Positionierung, zur Invertierung bei der Ausgabe und (Zeile 174Ø) zur evtl. vorzeitigen Beendigung der Plazierung.

Die Zeilen 177Ø bis 181Ø verändern in Abhängigkeit von der Eingabe der Cursor-Steuertasten die Positionswerte, bzw. fixieren bei Eingabe der COPY-Taste den aktuellen Feld-Eintrag.

Die Zeilen 186Ø und 187Ø sind nur eine Abfrage, ob die Maske so wie sie bisher vorliegt übernommen werden soll.

Durch die Zeilen 19ØØ bis 193Ø erfolgt eine Ausgabe der Daten für die mögliche Eintragsanzahl und den Offset, also den Abstand (Anzahl der Zeichen) von Eintrag zu Eintrag.

Je nachdem, wohin die Ausgabe der Maskendaten erfolgen soll, wird dann - durch 195Ø bis 212Ø - diese Ausgabe durchgeführt und das Programm beendet.

Da im Programm mehrere Abfragen auf "Ja-/Nein"-Entscheidungen hinauslaufen, sind die Zeilen 214Ø bis 216Ø die hierfür zuständige Unterroutine. Gleichzeitig werden alle Eingaben in Kleinbuchstaben gewandelt. Andere Zeichen als "j" und "n" werden nicht zugelassen.

Die Zeilen 218Ø bis 221Ø sind eine Unterroutine für evtl. auftretende Eingabefehler.

Die Zeilen 227Ø bis 232Ø verhindern eine Positionierung außerhalb der zulässigen Grenzen.

Die Zeilen 236Ø sind mit für die Ausgabe bei der Positionierung zuständig.

Die Zeilen 2430 bis 2460 enthalten die für die vier Dateiverwaltungsprogramme erforderlichen Daten. In Zeile 2410 steht die Reihenfolge der Datastatements.

Und nun das Programm selbst.

7.2.1 Basic-Listing: MASKGEN.BAS

```
100 '*******************************************************
110 '*                    MASKGEN.BAS                      *
120 '*_____________________________________________________*
130 '*                                                     *
140 '* Maskengenerator fuer die Dateiverwaltungsprogramme: *
150 '*                                                     *
160 '*                   RAMDISK1.BAS                      *
170 '*                   RAMDISK2.BAS                      *
180 '*                   RANDOMD1.BAS                      *
190 '*                   RANDOMD2.BAS                      *
200 '*-----------------------------------------------------*
210 '*       Copyright 1986 by Lothar Miedel 1986          *
220 '*******************************************************
230 '
240 OPENOUT"dummy":MEMORY HIMEM-1:CLOSEOUT:MODE 2
250 '
260 'Die Ausgabe-Fenster definieren
270 WINDOW #0,2,79,5,22:WINDOW #1,2,60,1,1:WINDOW #2,35,78,1,1
280 WINDOW #3,2,78,3,3:WINDOW #4,2,78,24,24:WINDOW #5,27,50,8,18
290 '
300 'Die Umrandungen (=Rahmen) zeichnen
310 DRAW 0,380:DRAW 639,380:DRAW 639,349:DRAW 0,349:DRAW 0,35
320 DRAW 639,35:DRAW 639,379:DRAW 639,0:DRAW 0,0
330 '
340 'Dimensionierungen
350 DIM fenam$(18),feldlen(18),xposi(18),yposi(18),blank$(18)
360 '-------------------------------------------------------
370 'Vordefinitionen
380 inv$=CHR$(24):m1$="Maskengenerator (c) by L. Miedel"
390 f1$="Fehler, bitte Taste (j oder n) druecken !"
400 blank$=SPACE$(56):invblank$=STRING$(56,CHR$(143))
410 '
420 FOR i = 1 TO 18
430   fenam$(i)="Feld-Nr. "+STR$(i)
440   IF LEN(fenam$(i))<14 THEN fenam$(i)=fenam$(i)+" ":GOTO 440
450   fenam$(i)=fenam$(i)+":"
460   feldlen(i)=20
470   xposi(i)=1:yposi(i)=i
480 NEXT i
```

```
490 '============================================================
500 PRINT #1,inv$ m1$ inv$
510 '
520 'Abfrage fuer welches Programm
530 FOR I = 1 TO 4
540   READ x1$(i),x2$(i),x3$(i),x4$(i),x5(i),x6(i)
550   PRINT #5,"    "i;"  "x1$(i):PRINT #5
560 NEXT i
570 '
580 PRINT #3,"Maskenerstellung -  Fuer welches Dateiverwaltungsprogramm?
590 PRINT #4,"Bitte Ihre Eingabe (1 bis 4 !)
600 x$=INKEY$:IF x$< "1" OR x$ >"4" THEN 600 ELSE x=VAL(x$)
610 masknam$=x2$(x):buffanf$=x3$(x):buffend$=x4$(x)
620 platz=x5(x):baenke=x6(x)
630 PRINT #3,"Gewaehltes Dateiprogramm: "x1$(x)
640 ERASE x1$,x2$,x3$,x4$,x5,x6:CLS #5
650 '
660 PRINT#4,"Soll eine bestehende Maske geladen werden (j/n)":GOSUB 2140
670 IF x$="j" THEN 690 ELSE GOTO 1000
680 '
690 PRINT#3,inv$"Ich lese die vorhandene Maske!" inv$
700 OPENIN masknam$
710  INPUT #9,buffanf$
720  INPUT #9,buffend$
730  INPUT #9,dateieintraege
740  INPUT #9,off
750  INPUT #9,felder
760   FOR i = 1 TO felder
770     INPUT #9,fenam$(i)
780     INPUT #9,feldlen(i)
790     INPUT #9,xposi(i)
800     INPUT #9,yposi(i)
810   NEXT i
820 CLOSEIN:PRINT #3,inv$"Maske gelesen!"inv$
830 PRINT #2,felder" Feld(er) pro Eintrag"
840 '
850 CLS #4:PRINT #4,inv$ "P" inv$ "lazierung oder "inv$"A"inv$"enderungen"
860 x$=LOWER$(INKEY$):IF x$="" THEN 860
870 IF x$="p" OR x$="a" THEN 880 ELSE GOTO 860
880 CLS #3:IF x$="p" THEN 1580
890 '
900 PRINT #4,"Aenderung der Feldanzahl (j/n)?"
910 GOSUB 2140:IF x$="j" THEN 1010
920 '
930 FOR i = 1 TO felder
940   fenam$(i)=LEFT$(fenam$(i),15)
950   LOCATE #0,1,i
960   PRINT USING "##\ \ \             \ ";i;".) ";fenam$(i)
```

```
970 NEXT i
980 PRINT#2,"Feldnameneingabe":GOTO 1240
990 '-------------------------------------------------------------------
1000 'Eingabe der Feldanzahl
1010 PRINT #3
1020 LOCATE #1,1,1
1030 PRINT #4,"Wieviele Felder soll ein Eintrag haben (max 18!) ";
1040 INPUT #4,felder$:felder =VAL(felder$)
1050 IF felder <1 OR felder >18 THEN GOSUB 2190:GOTO 1020
1060 PRINT #2,felder" Feld(er)"
1070 PRINT #0,felder" Feld(er)"
1080 GOSUB 2230:IF x$="n" THEN CLS #2:CLS #0:GOTO 1010
1090 CLS #3
1100 '-------------------------------------------------------------------
1110 'Eingabe der Feldnamen
1120 PRINT#2,"Feldnameneingabe
1130 CLS #0:PRINT#3,"Maximale Laenge der Feldnamen: 14 Zeichen
1140 i=1:LOCATE #0,1,1
1150 PRINT USING "## \ \";i;CHR$(242);
1160 PRINT #4,"Feld-Name "i":";
1170 INPUT #4,feldnam$:CLS #4
1180 IF feldnam$="" THEN 1220
1190 IF LEN(feldnam$)>14 THEN feldnam$=LEFT$(feldnam$,14)
1200 IF LEN(feldnam$)<14 THEN feldnam$=feldnam$+" ":GOTO 1200
1210 fenam$(i)=feldnam$+":"
1220 LOCATE #0,1,i:PRINT USING "##\ \ \              \ ";i;".) ";fenam$(i)
1230 i=i+1
1240 IF i>felder THEN GOSUB 2230:IF x$="n" THEN 1140 ELSE 1280
1250 GOTO 1150
1260 '-------------------------------------------------------------------
1270 'Eingabe der Anzahl von Feldzeichen
1280 PRINT#3,"Maximale Zeichenanzahl pro Feld: 56 Zeichen!
1290 i=1:ben=0:fehler=0:dateieintraege=platz*baenke:laenge=0:GOSUB 1460
1300 LOCATE #0,22,i:PRINT invblank$:CLS #4
1310 PRINT #4,LEFT$(fenam$(i),15);"Laenge (=";feldlen(i);
1320 PRINT #4,"/ ENTER = uebernehmen)";
1330 INPUT #4,laenge$:laenge=VAL(laenge$)
1340 IF laenge=0 THEN laenge$=""
1350 IF laenge$="" THEN IF feldlen(i)>0 THEN laenge=feldlen(i):GOTO 1370
1360 feldlen(i)=ABS(laenge)
1370 IF feldlen(i) <1 OR feldlen(i) >56 THEN 1300
1380 GOSUB 1460
1390 LOCATE #0,22,i:FOR j = 1 TO laenge:PRINT #0,".";:NEXT j
1400 FOR j =laenge TO 55:PRINT #0," ";:NEXT j
1410 i=i+1
1420 IF i>felder THEN GOSUB 2230:IF x$="n" THEN 1290 ELSE 1530
1430 GOTO 1300
1440 '-------------------------------------------------------------
```

```
1450 'addieren
1460 ben=INT(ben+laenge)
1470 off=ben
1480 IF off<1 THEN off=1
1490 dateieintraege=INT(platz/off)
1500 PRINT#2,"Anzahl der moeglichen Eintraege: "dateieintraege*baenke
1510 RETURN
1520 '-------------------------------------------------------------
1530 FOR i = 1 TO felder
1540   fenam$(i)=LEFT$(fenam$(i),15)+STRING$(feldlen(i),".")
1550 NEXT i
1560 '-------------------------------------------------------------
1570 'Plazierung der Felder
1580 CLS #0:PRINT #2,"Plazierung":PRINT #4,"Ihre Eingaben bitte!
1590 PRINT #3,inv$"Steuerung mit Cursortasten  -  Fixieren =COPY  "inv$;
1600 PRINT #3,inv$"   e= Ende der Plazierung"inv$
1610 i=1:x=1:y=1
1620 GOSUB 2360
1630 '-------------------------------------------------------
1640 LOCATE xposi(i),yposi(i):PRINT inv$ fenam$(i) inv$
1650 blank$(i)=STRING$(LEN(fenam$(i))," ")
1660 x=xposi(i):y=yposi(i)
1670 '
1680 x$=LOWER$(INKEY$):IF x$="" THEN 1680
1690 IF x$=CHR$(13) THEN 1680
1700 '
1710 LOCATE xposi(i),yposi(i):PRINT blank$(i)
1720 '
1730 IF x$="" THEN 1680
1740 IF x$="e" THEN LOCATE xposi(i),yposi(i):PRINT fenam$(i):GOTO 1860
1750 '
1760 aw=ASC(x$)'                      aw=ASCII-Wert
1770 IF aw=224 THEN i=i+1:GOTO 2340
1780 IF aw=240 THEN y=y-1
1790 IF aw=241 THEN y=y+1
1800 IF aw=242 THEN x=x-1
1810 IF aw=243 THEN x=x+1
1820 '
1830 GOSUB 2270:GOTO 1620
1840 '-------------------------------------------------------------
1850 'Maske in Ordnung ?
1860 GOSUB 2230
1870 IF x$="n" THEN 1580
1880 '-------------------------------------------------------------
1890 'Ausgabe der wichtigsten Daten
1900 CLS #2:PRINT #3,"Plazierung beendet
1910 CLS #0:PRINT:PRINT
1920 PRINT"Anzahl der moeglichen Eintraege : "dateieintraege*baenke
```

```
1930 PRINT"Offset pro Eintrag: "off
1940 '-------------------------------------------------------
1950 PRINT #4,"Soll die Feldmaske abgespeichert werden (j/n) ?"
1960 x$=LOWER$(INKEY$):IF x$="" THEN 1960
1970 IF x$="n" THEN output =0:GOTO 1990
1980 IF x$="j" THEN output=9:OPENOUT masknam$
1990 PRINT #output,buffanf$
2000 PRINT #output,buffend$
2010 PRINT #output,dateieintraege*baenke
2020 PRINT #output,off
2030 PRINT #output,felder
2040 FOR i = 1 TO felder
2050 PRINT #output,fenam$(i)
2060 PRINT #output,feldlen(i)
2070 PRINT #output,xposi(i)
2080 PRINT #output,yposi(i)
2090 NEXT i
2100 PRINT #output,"ende"
2110 CLOSEOUT:MODE 2:PRINT"Programmlauf beendet."
2120 END
2130 '-------------------------------------------------------
2140 x$=LOWER$(INKEY$):IF x$="" THEN 2140
2150 IF x$="j" OR x$="n" THEN RETURN
2160 GOTO 2140
2170 '-------------------------------------------------------
2180 'Fehlermeldung
2190 PRINT#1,inv$ f1$ inv$:fehler=1
2200 GOSUB 2140:IF x$="" THEN 2190 ELSE PRINT #1,m1$
2210 RETURN
2220 '-------------------------------------------------------
2230 PRINT#4,"Eingaben korrekt (j/n) ?"
2240 CLS #3
2250 GOSUB 2140:RETURN
2260 '-------------------------------------------------------
2270 IF x<=0 THEN x=x+1
2280 IF y<=0 THEN y=y+1
2290 IF x+LEN(fenam$(i))>=79 THEN x=x-1
2300 IF y>=19 THEN y=y-1
2310 xposi(i)=x:yposi(i)=y
2320 RETURN
2330 '-------------------------------------------------
2340 GOSUB 2360:IF i>felder THEN 1860 ELSE GOTO 1620
2350 '-------------------------------------------------
2360 FOR j = 1 TO felder
2370     LOCATE xposi(j),yposi(j)
2380     PRINT fenam$(j)
2390 NEXT j:RETURN
2400 '=======================================================
```

```
241Ø 'DATA progname,maskenname,bufferanfang,bufferende,platz,baenke
242Ø '
243Ø DATA RAMDISK1.BAS,RAMDISK1.MSK,3ØØØ,9FFF,28672,1
244Ø DATA RAMDISK2.BAS,RAMDISK2.MSK,4ØØØ,7FFF,16384,5
245Ø DATA RANDOMD1.BAS,RANDOMD1.MSK,9ØØØ,97ff,183Ø8,1
246Ø DATA RANDOMD2.BAS,RANDOMD2.MSK,9ØØØ,97ff,737284,1
```

Ein kleines Problem kann mit dem Maskengenerator auftreten, wenn bereits erstellte Masken eingelesen und verändert werden. Wurde die eingelesene Maske bereits plaziert und reicht nach der Änderung einer Feldlänge der Platz bis zum rechten Rand nicht mehr, dann wird dieses Feld in der nächsten Zeile ausgegeben.

Das wissen Sie aber! Dieser Fehler wird vom Programm nicht korrigiert. In einem derartigen Falle sollten Sie die Zeile 229Ø so abändern, daß am Ende THEN x=1 steht. Wenn Sie dann neu plazieren und auch das entsprechende Feld bewegen, dann wird alles wieder richtiggestellt!

7.3 Programm RAMDISK1

Das Programm besteht aus mehreren Teilen. Der Teil RAMDISK1.BAS ist der erste Teil zur Dateiverwaltung mit RAM-Disk-Betrieb. Es ist sozusagen des Steuerteil. Für den Programmlauf werden auch noch der Maschinenspracheteil, sowie die Maskendaten benötigt.

Das Programm läuft auf allen CPCs, es ist aber ganz speziell an die vorhandenen Voraussetzungen des CPC 464 bzw. 664 angepaßt.

Aus verschiedenen Gründen, die schon erwähnt wurden, ist die oberste Grenze, die vom Programm benutzt werden darf &9FFF. Da aber ein möglichst großer Bereich für Daten zuständig sein soll, wurde das Programm sehr stark komprimiert und der RAM-Bereich für die Maschinenroutinen zwischen &2E3B und &2FFF reserviert.

Damit das Laden und Speichern der Binärfiles klappt, muß der Buffer hierfür vorhanden sein. Dies bewerkstelligt die Zeile 11Ø. Gleichzeitig wird in dieser Zeile der Basiswert für die erste Eintragsnummer bestimmt und die Variable inv$, die zur Umkehrdarstellung von Zeichen dient, definiert.

Die Fensterdefinition und die Festlegung des Darstellungs-Rahmens geschieht durch die Zeilen 13Ø bis 17Ø.

In Zeile 19Ø geschieht das Laden des Maschinenprogramms und durch die von 21Ø angesprungene Unterroutine werden die Maskendaten eingelesen.

In den Zeilen 23Ø bis 28Ø erfolgen die Vordefinitionen der wichtigsten Variablen. So werden dort alle - für das Maschinenprogramm wichtigen - Adressen und Werte festgelegt. Zusätzliche Werte werden in 3ØØ berechnet. Ramlow und ramhigh sind die Low- bzw. Highbyte-Werte für den Anfang des Datenbereiches eines Eintrages, der beim Programmstart natürlich auf den Bufferanfang gestellt wird (Zeile 32Ø und 33Ø).

In den Zeilen 32Ø bis 34Ø werden alle Werte des Buffers in den Speicher gebracht.

Die Zeile 35Ø übergibt an die Maschinenroute "FILLER" das Zeichen, mit dem der definierte Bereich, der dem Datenbereich entspricht, "gefüllt" wird.

Unmittelbar danach erfolgt die Ausführung dieses MPG-Teiles.

Da die Maskendaten schon gelesen wurden, erfolgt dann ein Sprung zu Zeile 5ØØ. In dieser Zeile wird die mit dem Maskengenerator erstellte Maske ausgegeben. Dann beginnt das eigentliche Verwaltungsprogramm.

In 53Ø wird der Wert von "basis" als niedrigste Eintragsnummer (enr) übernommen und durch den Sprung nach 78Ø sofort geprüft, ob dieser Wert gültig ist. Dies geschieht deshalb, falls durch irgendeinen Grund (z.B. Fehler im Maskengenerator) falsche Daten eingelesen wurden.

Von Zeile 55Ø bis 6ØØ ist der Programmteil, der beim Ausführen des GOTO-Befehles die Eingabe und die Prüfung der Eingaben übernimmt.

Falls die Eingaben lt. der Prüfroutinen korrekt waren, stehen bei Ausführung der Zeilen ab 62Ø gültige Daten auf dem Schirm und dieses wird nun mitgeteilt. Sind die Datei-Daten noch nicht eingelesen, dann sind die Feldinhalte lauter "Punkte".

Von 65Ø bis 79Ø ist die Routine, die Ihre Eingabe erwartet und gegebenenfalls gleich fehlerhafte Eintragsnummern verhindert. Das bedeutet, zu niedrige Eingaben werden auf den niedrigsten (Zeile 78Ø) und zu hohe auf den höchsten (Zeile 79Ø) Wert gesetzt.

Nun müssen die Eintragsnummern innerhalb der erlaubten Grenzen liegen und deshalb können die Werte für den Bufferbeginn und für den Offset (Zeilen 82Ø und 83Ø) berechnet werden.

Ähnliches gilt dann für die Zeile und Anzahl der Zeichen des Feldeintrages (Zeile 87Ø), sowie für die höchste Adresse (ramhigh und ramlow in Zeile 88Ø).

Der weitere Vorgang läuft nun in einer kleinen Schleife ab, die von der Feldanzahl abhängig ist (Zeile 85Ø), wobei das Lesen aus dem Daten-Speicher und das Schreiben auf den Bildschirm durch die Routine ramwr (Schreiben aus dem RAM) bewerkstelligt wird.

Nun wird das Editier-Fenster - in diesen Programmen ist es immer der Stream #7 - definiert und die Grenzen

(1240) festgelegt. Durch die Zeile 1250 werden diese Werte in die hierfür vorgesehenen Speicherstellen des Maschinenprogrammes geladen.

In 900 erfolgt ein GOSUB-Sprung nach 1180, wo verschiedene Parameter an die Maschinenroutine übergeben werden.

Sind alle Felder des Eintrages ausgegeben, dann wird wieder nach Zeile 620 gesprungen (Zeile 900).

Je nach Eingabe - auf die in Zeile 650 gewartet wird - fährt das Programm fort.

Wurde "e" eingegeben, dann wird nochmals abgefragt, ob Sie den Programmlauf auch wirklich beenden wollen (Zeile 1310-1320).

Bei "f" wird die Suchroutine aufgerufen, deren Basic-Teil die Zeilen 1060 bis 1160 umfaßt.

Bei "g" erfolgt die Abarbeitung der Teilroutine, die für GOTO zuständig ist: Zeilen 550 bis 600.

Bei "l" für laden, erfolgt das Einlesen des Binärfiles mit den Daten (Zeile 1280 und 1290), bei "s" das Abspeichern des Datenfiles (Zeile 1290 und 1300).

Wurde die COPY-Taste gedrückt, dann haben Sie die Eingabe von Daten gewählt und die Routine ab Zeile 920 wird abgearbeitet. Hier läuft der umgekehrte Vorgang des Auslesens aus dem Speicher ab. Da im Maschinenprogramm die COPY-Taste abgefragt wird und bei gedrückter COPY-Taste über eine "Merker"-Speicherstelle dies signalisiert wird, kann der Basic-Teil es erkennen und darauf reagieren.

Nach jeder Feldeingabe (=Änderung) wird - durch Zeile 990 mit dem Befehl GOSUB 1180 - die auch bei der Ausgaberoutine verwendete Unterroutine angesprungen. Die Eingaben bei Änderungen werden "feldmäßig" behandelt.

Die Maschinenroutine für die Eingaben wird nach jeder Feldeingabe verlassen und in Zeile 1210 kann erkannt werden, ob während der Eingaben die COPY-Taste gedrückt wurde.

In 990 kommt die Gegenroutine des Lesens vom Bildschirm zum Arbeiten: RAMRD - Lesen vom Bildschirm und Schreiben in die RAM-Speicherstellen.

Die Zeilen 1010 bis 1040 dienen der Fehlerbehandlung.

Die Zeilen 1060 bis 1160 sind für die Suche nach, also für das Auffinden von, einzugebenden Zeichenfolgen zuständig.

Diese Zeilen will ich noch etwas genauer erläutern. Die Zeile 1060 dürfte klar sein, da geschehen die Informations-Ausgaben.

Die Zeile 1070 übernimmt die Eingabe in die Variable such$ und prüft, ob die Länge der Eingabe null ist. Trifft dies zu, dann wird als wirklicher Suchstring die alte Zeichenfolge (also eine vorher schon einmal eingegebene) benutzt.

Die Zeile 1080 stellt die Länge der Zeichenfolge fest und reduziert - falls es erforderlich sein sollte - auf 56 Zeichen.

Die Zeile 1090 schreibt die ASCII-Zeichen des Suchwortes für die Maschinenroutine in die extra dafür reservierten Speicherstellen.

Ab Zeile 1100 wird auf den nächsten Dateieintrag positioniert und in Zeile 1150 dann die Maschinenroutine für den Suchvorgang aufgerufen.

Da die Maschinenroutine bei erfolgreicher Suche einen Wert größer null in den Speicherstellen, die fundpos und fundpos +1 zugeordnet sind, übergibt, kann der Beginn der Übereinstimmung - nur bei einem komplett gefundenen "Wort" in die Variable fs übernommen werden (Zeile 1150).

In 1160 wird dann daraus der auszugebende Eintrag berechnet und durch den Sprung nach 780 ausgegeben.

Nun folgt das Basiclisting dieses Programmes. Die Hexlader für die Maschinenroutinen finden Sie nach dem Basic-Programm.

7.3.1 Basic-Listing: RAMDISK1.BAS

```
100 'RAMDISK1.BAS
110 MODE 2:MEMORY &2E3A:OPENOUT"dummy":CLOSEOUT:basis=1:inv$=CHR$(24)
120 '
130 WINDOW #0,2,79,5,22:WINDOW #1,2,79,1,1:WINDOW #2,61,78,1,1
140 WINDOW #3,2,78,3,3:WINDOW #4,60,78,24,24: WINDOW #5,2,50,24,24
150 DRAW 0,380:DRAW 639,380:DRAW 639,349:DRAW 0,349:DRAW 0,35
160 DRAW 639,35:DRAW 639,379:DRAW 639,0:DRAW 0,0
170 '
180 PRINT#1,inv$ "Dateiverwaltung mit RAM-DISK"inv$
190 PRINT#3,"Lade das Maschinenprogramm":LOAD"ramdisk1.mpg",&2E3B
200 PRINT#3,"Bitte Datendiskette einlegen und eine Taste druecken":CALL &BB06
210 GOSUB 370:'Maskendaten holen
220 '
230 zeilem=&2FC9:mspalte=&2FCA:manzahl=&2FCB:stream=&2FCC
240 linksm=&2FCD:rechtsm=&2FCE:memhig=&2FF4:fchar=&2FF6
250 filler=&2FD2:zeichen=&2E:ramdi=&2FD0:ctaste=&2FCF
260 ramrd=&2EF6:wed=&2F4B:ramwr=&2F2C:fundpos=&2EB2:suchwort=&2EBA
270 suche=&2E3B:ramlen=&2EAC:laenge=&2EAE:aktstart=&2EB0
280 zeile=1:spalte=1:anzahl=56:str=7
290 '
300 ramlow=VAL("&"+MID$(buffanf$,3,2)):ramhigh=VAL("&"+MID$(buffanf$,1,2))
310 '
320 POKE memhig+1,VAL("&"+MID$(buffend$,1,2))
330 POKE memhig,VAL("&"+MID$(buffend$,3,2))
340 POKE ramdi,ramlow:POKE ramdi+1,ramhigh
350 POKE fchar,zeichen:CALL filler:GOTO 500
360 '
370 PRINT#3,"Lese die Maskendaten":OPENIN"ramdisk1.msk"
380  INPUT #9,buffanf$,buffend$,eintraege,off,felder
390 buffanf$="3123"
400 buffanf=VAL("&"+buffanf$):buffend=VAL("&"+buffend$):diff=buffend-buffanf
410 IF diff <0 THEN diff=diff+2^16
420 DIM fenam$(felder),feldlen(felder),xposi(felder),yposi(felder)
430 DIM feldinh$(felder)
440   FOR i = 1 TO felder
450     INPUT #9,fenam$(i),feldlen(i),xposi(i),yposi(i)
460     benutzt=benutzt+feldlen(i)
470   NEXT i
480 CLOSEIN:PRINT#3,inv$ "Maske gelesen - Ausgabe der Maske"inv$:RETURN
490 '
500 FOR j = 1 TO felder:LOCATE xposi(j),yposi(j):PRINT fenam$(j):NEXT j
510 '
520 PRINT #3,"Auf geht's !":LOCATE #1,47,1:PRINT#1,"Dateieintraege: "eintraeg
530 enr=basis:GOTO 780
540 '
550 PRINT #3,inv$"GOTO ... "inv$:INPUT #5,"Eintragsnummer";enr$
```

```
560 IF LEN(enr$)<1 THEN 550
570 IF ASC(LEFT$(enr$,1))<48 THEN 550
580 IF ASC(LEFT$(enr$,1))>57 THEN 550
590 enr$=LEFT$(enr$,6)
600 enr=VAL(enr$):GOTO 780
610 '
620 CLS #3:PRINT #3,inv$ "Daten gueltig, bitte Ihre Eingabe!"inv$
630 CLS #5:PRINT #5,"Eintragsnummer: "enr:CLS #4
640 '
650 x$=LOWER$(INKEY$):IF x$="" THEN 650
660 IF x$="e" THEN 1310
670 IF x$="f" THEN 1060
680 IF x$="g" THEN 550
690 IF x$="l" THEN 1270
700 IF x$="s" THEN 1290
710 IF ASC(x$)=224 THEN 920
720 IF ASC(x$)<240 THEN 650
730 IF ASC(x$)>243 THEN 650
740 IF ASC(x$)=243 THEN enr=enr+1
750 IF ASC(x$)=242 THEN enr=enr-1
760 IF ASC(x$)=240 THEN enr=enr+10
770 IF ASC(x$)=241 THEN enr=enr-10
780 IF enr<basis THEN GOSUB 1010:enr=basis
790 IF enr>=eintraege+basis THEN GOSUB 1010:enr=eintraege+basis-1
800 '
810 CLS #3:PRINT #3,inv$"Moment bitte!"inv$
820 buffoff=(enr-basis)*off
830 begad=VAL("&"+buffanf$)+buffoff:'  Datenbuffer festlegen
840 '
850 POKE stream,str:zz=0:FOR i = 1 TO felder
860 beginn=begad+zz:zz=zz+feldlen(i)
870 zeile=yposi(i):anzahl=feldlen(i)-1
880 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
890 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+16,zeile+4,zeile+4:CLS #7
900 GOSUB 1180:CALL ramwr:NEXT i:GOTO 620
910 '
920 PRINT #3,"Bitte Ihre Eingaben!":PRINT #5,inv$"Aendern"inv$:PRINT #4,enr
930 POKE stream,str:zz=0
940 FOR i = 1 TO felder
950 beginn=begad+zz:zz=zz+feldlen(i):zeile=yposi(i):anzahl=feldlen(i)-1
960 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
970 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+17,zeile+4,zeile+4
980 GOSUB 1240:CALL WED
990 anzahl=feldlen(i):GOSUB 1180:CALL ramrd:NEXT i:PRINT#5:GOTO 620
1000 '
1010 PRINT #3,inv$ "Fehler! "inv$;
1020 PRINT #5,inv$" Bitte Taste 'j' oder 'n' druecken!"inv$:PRINT CHR$(7);
1030 x$=LOWER$(INKEY$):IF x$="" THEN 1030
```

```
1040 IF x$<> "j" AND x$<> "n" THEN 1030 ELSE CLS #5:RETURN
1050 '
1060 PRINT #3,inv$"Suchen: "inv$"  Bitte geben Sie die Zeichenfolge ein!  "
1070 INPUT #5,such$:IF LEN(such$)=0 THEN sw$=sw$ ELSE sw$=such$
1080 wlaenge=LEN(sw$):IF wlaenge>56 THEN sw$=LEFT$(sw$,56):GOTO 1080
1090 FOR i =1 TO wlaenge:POKE suchwort+i-1,ASC(MID$(sw$,i,1)):NEXT i
1100 stand=buffanf+enr*off:sthi=INT(stand/256):stlo=stand-sthi*256
1110 rest=buffend-stand:IF rest <0 THEN rest=rest+2^16
1120 ramlhi=INT(rest/256):ramllo=rest-ramlhi*256
1130 POKE laenge,wlaenge:POKE aktstart,stlo:POKE aktstart+1,sthi
1140 POKE ramlen,ramllo:POKE ramlen+1,ramlhi
1150 CALL suche:fs=PEEK(fundpos)+PEEK(fundpos+1)*256
1160 enr=INT((fs-buffanf)/off)+basis:GOTO 780
1170 '
1180 zeile=1:spalte=1:POKE zeilem,zeile
1190 POKE manzahl,anzahl:POKE mspalte,spalte
1200 POKE ramdi,ramlow:POKE ramdi+1,ramhigh
1210 IF PEEK(ctaste)=224 THEN i=felder
1220 POKE ctaste,0:RETURN
1230 '
1240 links=16:rechts=feldlen(i)
1250 POKE stream,str:POKE linksm,1:POKE rechtsm,rechts:RETURN
1260 '
1270 PRINT#3,inv$"Datenfile laden"inv$:GOSUB 1020
1280 IF x$="j" THEN LOAD"ramdisk1.dat",buffanf:GOTO 780 ELSE GOTO 780
1290 PRINT#3,inv$"Datenfile speichern"inv$:GOSUB 1020
1300 IF x$="j" THEN SAVE"ramdisk1.dat",b,buffanf,diff:GOTO 780 ELSE GOTO 780
1310 PRINT#3,inv$"Programm beenden ?"inv$:GOSUB 1020
1320 IF x$="n" THEN GOTO 780  ELSE MODE 2:END
```

Damit Sie die einzelnen Maschinen-Routinen nicht nur für die Programme dieses Buches einsetzen können, wurden sie getrennt assembliert und deshalb stehen Ihnen die Lade-Programme auch als Module zur Verfügung.

Zunächst nun die Hexlader der einzelnen Module und dann anschließend die kommentierten Assembler-Listings. Für das Programm sollten Sie den Hexlader KPLRD1.HEX abschreiben und laufen lassen. Im Anschluß an den Programmlauf steht dann der komplette Maschinensprache-Teil auf der Diskette. Die einzelnen kommentierten Assembler-Listings brauchen Sie gedanklich nur zusammenzufügen, dann haben Sie auch das kommentierte Assemblerlisting für die komplette Maschinensprache-Routine.

7.3.2 Hexlader: SUCHER.HEX

```
100 'SUCHER.HEX
110 MEMORY &2E3A
120 '
130 a= 11835:e= 11949:zb=1000
140 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 170
150 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
160 ps=0:d$="":IF i=e THEN 220 ELSE i=i-1:zb=zb+1:GOTO 200
170 d$="&"+d$
180 POKE i,VAL(d$):ps=ps+VAL(d$)
190 PRINT VAL(d$)
200 IF i < e THEN NEXT i
210 '
220 SAVE"sucher.bin",b,&2E3B,&71
230 '
1000 DATA ED,4B,AC,2E,11,BA,2E,2A,B0,2E,22,B4,2E,3A,AE,2E,&062D
1001 DATA 32,AF,2E,1A,ED,B1,C2,A5,2E,2B,22,B0,2E,22,B8,2E,&068F
1002 DATA 22,B4,2E,3A,AE,2E,32,AF,2E,1A,ED,B1,C2,A5,2E,D5,&074B
1003 DATA E5,2A,B4,2E,23,E5,D1,E1,E5,ED,52,E1,D1,22,B4,2E,&0985
1004 DATA CA,88,2E,2A,B0,2E,23,22,B0,2E,C3,3B,2E,22,B6,2E,&05DD
1005 DATA 3A,AF,2E,3D,32,AF,2E,FE,00,CA,9B,2E,13,C3,64,2E,&065C
1006 DATA ED,5B,AE,2E,ED,52,22,B2,2E,C9,21,00,00,22,B2,2E,&0651
1007 DATA C9,00,&00C9
```

Falls Sie das Suchprogramm also getrennt benötigen, brauchen Sie nur das vorstehende Ladeprogramm abzutippen. Die erforderlichen Adressen für den Aufruf bzw. für von Basic aus zu übergebenden Werte stehen im Basic-Programm RAMDISK1.BAS und sind auch bei den Assembler-Listings zu finden.

Falls Sie den "Sucher" aber in einem anderen Speicherbereich benötigen, bei der Beschreibung RAMDISK2 finden Sie ihn noch für die Adresse ab &8000.

7.3.3 Hexlader: RD.HEX

```
100 REM RD.HEX
110 MEMORY &2EF5
120 '
130 a= 12022:e= 12233:zb=1000
140 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 170
150 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
160 ps=0:d$="":IF i=e THEN 220 ELSE i=i-1:zb=zb+1:GOTO 200
170 d$="&"+d$
180 POKE i,VAL(d$):ps=ps+VAL(d$)
```

```
190 PRINT VAL(d$)
200 IF i < e THEN NEXT i
210 '
220 SAVE"rd.bin",b,&2EF6,&D2
230 '
1001 DATA 3A,CC,2F,CD,B4,BB,3A,C9,2F,F5,CD,72,BB,F1,32,C9,&097E
1002 DATA 2F,3A,CA,2F,F5,CD,6F,BB,F1,3C,32,CA,2F,2A,CB,2F,&07CA
1003 DATA 7D,FE,00,C8,2B,22,CB,2F,CD,60,BB,2A,D0,2F,77,23,&0735
1004 DATA 22,D0,2F,C3,07,2F,3A,CC,2F,CD,B4,BB,2A,D0,2F,7E,&0732
1005 DATA 23,22,D0,2F,CD,5D,BB,2A,CB,2F,7D,FE,00,C8,2B,22,&06DD
1006 DATA CB,2F,C3,32,2F,3A,CC,2F,CD,B4,BB,CD,78,BB,CD,75,&08D1
1007 DATA BB,CD,8A,BB,CD,06,BB,4F,CD,8D,BB,79,FE,0D,C8,FE,&0A09
1008 DATA E0,CA,C5,2F,FE,F2,CA,91,2F,FE,F3,CA,A7,2F,FE,7F,&0B26
1009 DATA F2,51,2F,FE,20,FA,51,2F,CD,5A,BB,CD,78,BB,3A,CE,&08F4
1010 DATA 2F,3C,57,7C,BA,CA,BD,2F,C3,51,2F,3E,08,4F,CD,78,&06CB
1011 DATA BB,3A,CD,2F,57,7C,BA,CA,51,2F,79,CD,5A,BB,C3,51,&0837
1012 DATA 2F,3E,09,4F,CD,78,BB,3A,CE,2F,57,7C,BA,CA,51,2F,&06D3
1013 DATA 79,CD,5A,BB,C3,51,2F,3E,08,CD,5A,BB,C3,51,2F,32,&073B
1014 DATA CF,2F,C9,&01C7
```

Auch der Hexlader für die RAMDISK kann eigenständig benutzt werden. Sinngemäß gilt das beim SUCHER.HEX Geschriebene.

7.3.4 Hexlader: FILLER.HEX

```
100 REM FILLER.HEX
110 MEMORY &2FCF
120 '
130 a= 12240:e= 12280:zb=1000
140 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 170
150 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
160 ps=0:d$="":IF i=e THEN 210 ELSE i=i-1:zb=zb+1:GOTO 200
170 d$="&"+d$
180 POKE i,VAL(d$):ps=ps+VAL(d$)
190 PRINT VAL(d$)
200 IF i < e THEN NEXT i
210 SAVE"FILLER.BIN",B,&2FD0,&27
220 '
1001 DATA 00,00,2A,D0,2F,2B,E5,C1,2A,F4,2F,E5,ED,42,E5,C1,&0801
1002 DATA E1,3A,F6,2F,77,2B,0B,78,FE,00,C2,E1,2F,79,FE,00,&07AC
1003 DATA C8,C3,E1,2F,00,00,2E,00,&02C9
```

Wie bei den beiden vorangegangenen Ladern kann auch mit diesem ein eigenständiger Maschinenspracheteil erzeugt werden.

Nun zu dem Lade-Programm, das den kompletten Maschinenspracheteil erzeugt, den Sie für das Programm RAMDISK1 brauchen:

7.3.5 Hexlader: RAMDISK1.HEX

```
100 REM RAMDISK1.HEX
110 MEMORY &2E3A
120 '
130 a= 11835:e= 12280:zb=1000
140 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 170
150 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
160 ps=0:d$="":IF i=e THEN 220 ELSE i=i-1:zb=zb+1:GOTO 200
170 d$="&"+d$
180 POKE i,VAL(d$):ps=ps+VAL(d$)
190 PRINT VAL(d$)
200 IF i < e THEN NEXT i
210 '
220 SAVE"ramdisk1.mpg",b,&2E3B,&1BC
230 '
1000 DATA ED,4B,AC,2E,11,BA,2E,2A,B0,2E,22,B4,2E,3A,AE,2E,&062D
1001 DATA 32,AF,2E,1A,ED,B1,C2,A5,2E,2B,22,B0,2E,22,B8,2E,&068F
1002 DATA 22,B4,2E,3A,AE,2E,32,AF,2E,1A,ED,B1,C2,A5,2E,D5,&074B
1003 DATA E5,2A,B4,2E,23,E5,D1,E1,E5,ED,52,E1,D1,22,B4,2E,&0985
1004 DATA CA,88,2E,2A,B0,2E,23,22,B0,2E,C3,3B,2E,22,B6,2E,&05DD
1005 DATA 3A,AF,2E,3D,32,AF,2E,FE,00,CA,9B,2E,13,C3,64,2E,&065C
1006 DATA ED,5B,AE,2E,ED,52,22,B2,2E,C9,21,00,00,22,B2,2E,&0651
1007 DATA C9,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&00C9
1008 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
1009 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
1010 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
1011 DATA 00,00,00,00,00,00,00,00,00,00,00,3A,CC,2F,CD,B4,&02B6
1012 DATA BB,3A,C9,2F,F5,CD,72,BB,F1,32,C9,2F,3A,CA,2F,F5,&091F
1013 DATA CD,6F,BB,F1,3C,32,CA,2F,2A,CB,2F,7D,FE,00,C8,2B,&07E1
1014 DATA 22,CB,2F,CD,60,BB,2A,D0,2F,77,23,22,D0,2F,C3,07,&06B2
1015 DATA 2F,3A,CC,2F,CD,B4,BB,2A,D0,2F,7E,23,22,D0,2F,CD,&0758
1016 DATA 5D,BB,2A,CB,2F,7D,FE,00,C8,2B,22,CB,2F,C3,32,2F,&06EA
1017 DATA 3A,CC,2F,CD,B4,BB,CD,78,BB,CD,75,BB,CD,8A,BB,CD,&0A4D
1018 DATA 06,BB,4F,CD,8D,BB,79,FE,0D,C8,FE,E0,CA,C5,2F,FE,&0A0B
1019 DATA F2,CA,91,2F,FE,F3,CA,A7,2F,FE,7F,F2,51,2F,FE,20,&0A1A
1020 DATA FA,51,2F,CD,5A,BB,CD,78,BB,3A,CE,2F,3C,57,7C,BA,&085C
1021 DATA CA,BD,2F,C3,51,2F,3E,08,4F,CD,78,BB,3A,CD,2F,57,&071B
1022 DATA 7C,BA,CA,51,2F,79,CD,5A,BB,C3,51,2F,3E,09,4F,CD,&0781
```

```
1Ø23 DATA 78,BB,3A,CE,2F,57,7C,BA,CA,51,2F,79,CD,5A,BB,C3,&Ø85F
1Ø24 DATA 51,2F,3E,Ø8,CD,5A,BB,C3,51,2F,32,CF,2F,C9,Ø1,Ø1,&Ø5E6
1Ø25 DATA Ø1,Ø7,Ø5,Ø6,ØØ,ØØ,ØØ,2A,DØ,2F,2B,E5,C1,2A,F4,2F,&Ø45A
1Ø26 DATA E5,ED,42,E5,C1,E1,3A,F6,2F,77,2B,ØB,78,FE,ØØ,C2,&Ø8DF
1Ø27 DATA E1,2F,79,FE,ØØ,C8,C3,E1,2F,ØØ,ØØ,2E,ØØ,&Ø55Ø
```

Da alle Maschinenroutinen so liegen, daß ab &3ØØØ die Daten stehen können, sind die mit den Hexladern zu erzeugenden Maschinenspracheteile vor allem für den CPC 464 und 664 gedacht. Für den CPC 6128 finden Sie in diesem Buch auch die Hexlader die entsprechende Binärfiles ab der Adresse &8ØØØ erzeugen.

Nun folgen noch die kommentierten Assemblerlistings zu diesen Ladeprogrammen.

7.3.6 Assembler-Listing: SUCHER.EDI

```
                    ;*****************************************
                    ;*              SUCHER.EDI               *
                    ;*       (c) 1986 by Lothar Miedel       *
                    ;*---------------------------------------*
                    ;* Dieses Maschinen-Programm sucht Woerter *
                    ;* (=ASCII-Folgen) im Speicher des CPC     *
                    ;*****************************************
;
;Erklaerungen
;============
;AKSTART= aktuelle Startadresse fuer die Suche
;ALTMERK= Merker fuer alten Suchstart
;FUNDPOS= Adresse des ersten Buchstabens bei
;         erfolgreicher Suche (Begriff gefunden)
;EFIND  = Adresse der ersten Uebereinstimmung
;         bei jedem Suchbeginn
;ERFOLG = Wort komplett gefunden
;ERSTBYT= Adresse der ersten Uebereinstimmung
;         bei erfolgreicher Suche
;ERSTPR = Suchbeginn fuer ersten Buchstaben
;KOMPL? = Pruefung ob Wort komplett gefunden
;LAENGE = Zeichenlaenge des Suchbegriffes
;NOERFOL= Start der Subroutine wenn Suche erfolglos
;PRUEF  = Pruefroutine ob Wort komplett
;RAMLEN = Laenge des zu durchsuchenden RAM-Bereiches
;RESTLEN= Anzahl der noch fehlenden Uebereinstimmungen
;SUCHER = Einsprungadresse von Basic aus
;SUCHW  = Adresse an der das Suchwort beginnt
```

```
SUCH2  = Start der Suche nach erster Uebereinstimmung
;START  = wirklicher Beginn des Programmes
-----------------------------------------------------

                        org   &2e3b

2E3B  ED 4B AC 2E   START   ld    bc,(RAMLEN)   ;Laenge Suchbereich
2E3F  11 BA 2E              ld    de,SUCHW      ;Adresse des 1. Buchstaben
2E42  2A BØ 2E              ld    hl,(AKSTART)  ;Startadresse der Suche
2E45  22 B4 2E              ld    (ALTMERK),hl  ;Merker alter Suchbeginn

2E48  3A AE 2E      ERSTPR  ld    a,(LAENGE)    ;Laenge des Wortes
2E4B  32 AF 2E              ld    (RESTLEN),a   ;in RESTLEN schreiben
2E4E  1A                    ld    a,(de)        ;ersten Buchstaben laden
2E4F  ED B1                 cpir                ;und suchen
2E51  C2 A5 2E              jp    nz,NOERFOL    ;nichts gefunden
2E54  2B                    dec   hl            ;Startadresse erniedrigen
2E55  22 BØ 2E              ld    (AKSTART),hl  ;aktueller Start
2E58  22 B8 2E              ld    (ERSTBYT),hl  ;Adresse erstes Bytes
2E5B  22 B4 2E              ld    (ALTMERK),hl  ;auch merken
2E5E  3A AE 2E              ld    a,(LAENGE)    ;Laenge des Wortes
2E61  32 AF 2E              ld    (RESTLEN),a   ;in RESTLEN merken

2E64  1A            SUCH2   ld    a,(de)        ;Suchzeichen laden
2E65  ED B1                 cpir                ;suchen
2E67  C2 A5 2E              jp    nz,NOERFOL    ;nicht gefunden

2E6A  D5            PRUEF   push  de            ;Position im Wort retten
2E6B  E5                    push  hl            ;gefundene Position + 1
2E6C  2A B4 2E              ld    hl,(ALTMERK)  ;Merker laden
2E6F  23                    inc   hl            ;und fuer SBC erhoehen
2E7Ø  E5                    push  hl            ;de mit
2E71  D1                    pop   de            ;(ALTMERK)+1 laden
2E72  E1                    pop   hl            ;gefundene Position +1
2E73  E5                    push  hl            ;merken
2E74  ED 52                 sbc   hl,de         ;subtrahieren
2E76  E1                    pop   hl            ;Position + 1 wieder holen
2E77  D1                    pop   de            ;Position im Wort
2E78  22 B4 2E              ld    (ALTMERK),hl  ;als alten Wert merken
2E7B  CA 88 2E              jp    z,KOMPL?      ;ist Wort komplett?
2E7E  2A BØ 2E              ld    hl,(AKSTART)  ;Zeichenfolge falsch
2E81  23                    inc   hl            ;ein Byte weiter
2E82  22 BØ 2E              ld    (AKSTART),hl  ;merken und dort
2E85  C3 3B 2E              jp    START         ;Suche fortsetzen

2E88  22 B6 2E      KOMPL?  ld    (EFIND),hl    ;erste Fundadresse
2E8B  3A AF 2E              ld    a,(RESTLEN)   ;Restlaenge holen
2E8E  3D                    dec   a             ;Zeichen stimmt in der
```

```
2E8F  32 AF 2E            ld    (RESTLEN),a  ;Folge - merken!
2E92  FE 00               cp    a,0          ;komplettes Wort gefunden?
2E94  CA 9B 2E            jp    z,ERFOLG     ;ja - Suche erfolgreich
2E97  13                  inc   de           ;noch nicht alle Zeichen
2E98  C3 64 2E            jp    SUCH2        ;deshalb Suche fortsetzen

2E9B  ED 5B AE 2E  ERFOLG ld    de,(LAENGE)  ;Laenge des Wortes
2E9F  ED 52               sbc   hl,de        ;subtrahieren
2EA1  22 B2 2E            ld    (FUNDPOS),hl ;Beginn des Wortes
2EA4  C9                  ret                ;zurueck nach Basic

2EA5  21 00 00     NOERFOL ld    hl,&0000     ;nichts gefunden - keine
2EA8  22 B2 2E             ld    (FUNDPOS),hl ;Fundposition fuer Basic
2EAB  C9                   ret                ;zurueck nach Basic

2EAC  (0002)       RAMLEN  defs  2
2EAE  (0001)       LAENGE  defs  1
2EAF  (0001)       RESTLEN defs  1
2EB0  (0002)       AKSTART defs  2
2EB2  (0002)       FUNDPOS defs  2
2EB4  (0002)       ALTMERK defs  2
2EB6  (0002)       EFIND   defs  2
2EB8  (0002)       ERSTBYT defs  2
2EBA  (003C)       SUCHW   defs  60
```

```
Fehler: 00000  Warnungen: 00000

Symboltabelle:

2EB0 AKSTART   2EB4 ALTMERK   2E48 ERSTPR   2E9B ERFOLG   2EB6 EFIND
2EB8 ERSTBYT   2EB2 FUNDPOS   2E88 KOMPL?   2EAE LAENGE   2EA5 NOERFOL
2E6A PRUEF     2EAC RAMLEN    2EAF RESTLEN  2E3B START    2E64 SUCH2
2EBA SUCHW
```

7.3.7 Assembler-Listing: RD.EDI

```
                ;****************************************
                ;*                RD.EDI                *
                ;* Maschinenprogramme fuer RAMDISK.BAS *
                ;*                                      *
                ;****************************************

                org   &2ef6
;
;Benutzte Firmware-Routinen mit Funktion (F),
;Einsprungbedingungen (E) und Aussprungbedingungen (A)
```

```
;txt cur enable          (CUREN)  F=Lasse Darstellung Anwender-Cursor zu
;       E= keine
;       A= af zerstoert
;
;txt cur disable         (CURDI)  F=Verhindere Darstellung Anwender-Cursor
;       E= keine
;       A= af zerstoert
;
;txt read char           (RDCHAR) F= Lese Zeichen vom Bildschirm
;       E= keine
;       A=   wenn Zeichen erkannt wurde
;             CARRY an - a enthaelt Zeichen
;            wenn kein Zeichen erkannt wurde
;             CARRY aus - a enthaelt Null
;       immer alle anderen Flags zerstoert
;
;txt set column          (SETCOL) F= Setze horizontale Position des Cursors
;       E= a enthaelt die logische Spalte
;       A= af und hl zerstoert
;
;txt set row             (SETROW) F= Setze die vertikale Position des Cursors
;       E= a enthaelt die logische Zeile
;       A= af und hl zerstoert
;
;txt stream select       (STRSEL) F= Selektiere das Ein-/Ausgabegeraet (Stream)
;       E= a enthaelt Stream-Nummer
;       A= a enthaelt vorherigen Stream
;          hl und Flags zerstoert
;
;txt wr char             (WRCHAR) F= Schreibe Zeichen auf den Schirm
;       E= a enthaelt das Zeichen
;       A= af,bc,de,hl zerstoert
;
;km wait char            (WAITCH) F= Wartet auf naechstes Tastaturzeichen
;       E= keine
;       A= Carry ein
;          a enthaelt das Zeichen
;          Alle anderen Flags zerstoert
;
;txt output              (TXTOUT) F= Zeichen/Steuercode an Text-VDU uebergeben
;       E= Akku enthaelt das Zeichen
;       A= alles bleibt erhalten
;
;txt set cursor          (SETCUR) F= Cursor positionieren
;       E= h logische Spalte
;          l logische Zeile
;       A= af, hl zerstoert
;
```

```
;txt place cursor          (PLACUR) F= Stelle Cursor auf dem Bildschirm dar
;       E= keine
;       A= af zerstoert
;
;txt remove cursor         (REMCUR) F= Loesche Cursor auf dem Bildschirm
;       E= keine
;       A= af zerstoert
;
;txt get cursor            (GETCUR) F= Frage nach aktueller Cursorposition
;       E= keine
;       A= h enthaelt logische Spalte
;          l enthaelt logische Zeile
;          alle Flags zerstoert
;----------------------------------------
;
;* Merker und Label *
;
;ANZAHL  =Merker fuer zugelassene Zeichenanzahl im Stream
;ARBEIT  =Einsprungpunkt bei RAMRD fuer komplettes Lesen
;COPY    =WED-Routine wenn Copy-Taste gedrueckt wurde
;CTASTE  =Merker ob COPY-Taste gedrueckt wurde
;LINKS   =WED-Routine wenn Cursor links eingegeben wurde
;LINKSM  =Merker linke Grenze des Streams
;RECHTS  =WED-Routine wenn Cursor rechts eingegeben
;RECHTSM =Merker rechte Grenze des Streams
;RAMDI   =Zeiger auf aktuelle RAM-Disk Adresse (wird auch von FILLER benutzt!)
;RAMWR   =Einsprung fuer Lesen aus RAM-Disk und Schreiben auf Schirm
;RAMLES  =Einsprungpunkt bei RAMWR fuer komplettes Schreiben
;RAMRD   =Einsprung fuer Lesen vom Schirm und Schreiben in RAM-Disk
;SPALTE  =Merker Spaltenposition des aktuellen Streams
;STREAM  =Merker Streamnummer
;WED     =Einsprungadresse des WINDOW-Editors
;WLOOP   =Schleifeneinsprung fuer Window-Editor
;WRR     =WED-Routine wenn rechter Window-Rand erreicht
;ZEILE   =Merker Zeilenposition des aktuellen Streams
;
;* Definitionen *
;
CURDI   equ   &bb7e:CUREN   equ   &bb7b:RDCHAR  equ   &bb6Ø
SETCOL  equ   &bb6f:SETROW  equ   &bb72:STRSEL  equ   &bbb4
WRCHAR  equ   &bb5d:WAITCH  equ   &bbØ6:TXTOUT  equ   &bb5a
SETCUR  equ   &bb75:PLACUR  equ   &bb8a:REMCUR  equ   &bb8d
GETCUR  equ   &bb78
;
;
```

```
                    ;******************** RAMRD ********************
                    ;* Lesen von Bildschirmzeichen und schreiben *
                    ;*                 in RAM-Disk               *
                    ;***********************************************

2EF6  3A CC 2F      RAMRD   ld    a,(STREAM)   ;Streamauswahl
2EF9  CD B4 BB              call  STRSEL       ;aufrufen
2EFC  3A C9 2F              ld    a,(ZEILE)    ;Zeilenposition
2EFF  F5                    push  af           ;retten
2F00  CD 72 BB              call  SETROW       ;positionieren
2F03  F1                    pop   af           ;Position wieder
2F04  32 C9 2F              ld    (ZEILE),a    ;rueckspeichern

2F07  3A CA 2F      ARBEIT  ld    a,(SPALTE)   ;Spalte holen
2F0A  F5                    push  af           ;und merken
2F0B  CD 6F BB              call  SETCOL       ;Spalte positionieren
2F0E  F1                    pop   af           ;wieder holen
2F0F  3C                    inc   a            ;erhoehen
2F10  32 CA 2F              ld    (SPALTE),a   ;neue Spalte merken

2F13  2A CB 2F              ld    hl,(ANZAHL)  ;verbliebene Zeichen
2F16  7D                    ld    a,l          ;Anzahl zum Vergleich in Akku
2F17  FE 00                 cp    0            ;Ende ?
2F19  C8                    ret   z            ;ja -> Basic

2F1A  2B                    dec   hl           ;Zeichenanzahl erniedrigen
2F1B  22 CB 2F              ld    (ANZAHL),hl  ;neue Anzahl merken

2F1E  CD 60 BB              call  RDCHAR       ;Lesen und Zeichen in den Akku
2F21  2A D0 2F              ld    hl,(RAMDI)   ;RAM-Disk-Adresse holen und
2F24  77                    ld    (hl),a       ;Akkuinhalt in RAM-Disk
2F25  23                    inc   hl           ;RAM-Disk-Adresse erhoehen
2F26  22 D0 2F              ld    (RAMDI),hl   ;und merken
2F29  C3 07 2F              jp    ARBEIT       ;weitermachen

                    ;***********************************************
                    ;************          RAMWR        ************
                    ;*  Lesen von Zeichen aus der RAM-Disk und   *
                    ;*         schreiben auf den Bildschirm      *
                    ;***********************************************

2F2C  3A CC 2F      RAMWR   ld    a,(STREAM)   ;Stream auswaehlen
2F2F  CD B4 BB              call  STRSEL
2F32  2A D0 2F      RAMLES  ld    hl,(RAMDI)   ;Adresse der RAM-Disk holen
2F35  7E                    ld    a,(hl)       ;Zeichen aus RAM-Disk lesen
2F36  23                    inc   hl           ;RAM-Diskadresse erhoehen
2F37  22 D0 2F              ld    (RAMDI),hl   ;und merken
2F3A  CD 5D BB              call  WRCHAR       ;Zeichen aus Akku schreiben
```

```
2F3D  2A CB 2F            ld    hl,(ANZAHL)  ;Zeichenanzahl holen
2F40  7D                  ld    a,l          ;Anzahl in den Akku
2F41  FE 00               cp    a,0          ;vergleichen auf Null
2F43  C8                  ret   z            ;zurueck ins Basic
2F44  2B                  dec   hl           ;Zeichenanzahl erniedrigen
2F45  22 CB 2F            ld    (ANZAHL),hl  ;und merken
2F48  C3 32 2F            jp    RAMLES

                ;****************************************
                ;** WED  - Window-Editor  ***************
                ;****************************************

2F4B  3A CC 2F    WED     ld    a,(STREAM)   ;STREAM holen
2F4E  CD B4 BB            call  STRSEL       ;und auswaehlen

2F51  CD 78 BB    WLOOP   call  GETCUR       ;Cursorposition holen
2F54  CD 75 BB            call  SETCUR       ;Cursor setzen
2F57  CD 8A BB            call  PLACUR       ;Cursor auf erste Position
2F5A  CD 06 BB            call  WAITCH       ;Auf Zeichen warten
2F5D  4F                  ld    c,a          ;Zeichen retten
2F5E  CD 8D BB            call  REMCUR       ;Cursorsymbol entfernen
2F61  79                  ld    a,c          ;Zeichen wieder holen
2F62  FE 0D               cp    a,13         ;ist es ENTER (RETURN)?
2F64  C8                  ret   z            ;ja -> Basic
2F65  FE E0               cp    a,224        ;Copy-Taste ?
2F67  CA C5 2F            jp    z,COPY       ;Sprung zur Routine Copy-Taste
2F6A  FE F2               cp    a,242        ;ist es Cursor links ?
2F6C  CA 91 2F            jp    z,LINKS      ;ja -> Routine links
2F6F  FE F3               cp    a,243        ;ist es Cursor rechts ?
2F71  CA A7 2F            jp    z,RECHTS     ;ja -> Routine rechts
2F74  FE 7F               cp    a,&7f        ;Zeichen groesser &7f
2F76  F2 51 2F            jp    p,WLOOP      ;ja weitermachen
2F79  FE 20               cp    a,32         ;Zeichen kleiner 32
2F7B  FA 51 2F            jp    m,WLOOP      ;ja weitermachen

2F7E  CD 5A BB            call  TXTOUT       ;und ausgeben
2F81  CD 78 BB            call  GETCUR       ;und Cursorposition holen
2F84  3A CE 2F            ld    a,(RECHTSM)  ;rechten Maximalwert
2F87  3C                  inc   a            ;erhoehen
2F88  57                  ld    d,a          ;und merken
2F89  7C                  ld    a,h          ;aktuelle Position holen
2F8A  BA                  cp    a,d          ;und vergleichen
2F8B  CA BD 2F            jp    z,WRR        ;Windowposition rechts erreicht
2F8E  C3 51 2F            jp    WLOOP        ;weitermachen

2F91  3E 08       LINKS   ld    a,8          ;Akku mit Backspace laden
2F93  4F                  ld    c,a          ;und merken
2F94  CD 78 BB            call  GETCUR       ;Cursor holen
```

```
2F97  3A CD 2F            ld    a,(LINKSM)    ;linken Grenzwert
2F9A  57                  ld    d,a           ;merken und mit
2F9B  7C                  ld    a,h           ;aktuellem Wert
2F9C  BA                  cp    a,d           ;vergleichen
2F9D  CA 51 2F            jp    z,WLOOP       ;Rand erreicht
2FA0  79                  ld    a,c           ;Backspace
2FA1  CD 5A BB            call  TXTOUT        ;ausfuehren
2FA4  C3 51 2F            jp    WLOOP         ;weitermachen

2FA7  3E 09         RECHTS ld    a,9           ;Akku mit Cursor vorw. laden
2FA9  4F                  ld    c,a           ;und merken
2FAA  CD 78 BB            call  GETCUR        ;aktuellen Wert holen
2FAD  3A CE 2F            ld    a,(RECHTSM)   ;rechten Grenzwert
2FB0  57                  ld    d,a           ;merken
2FB1  7C                  ld    a,h           ;aktuellen Wert
2FB2  BA                  cp    a,d           ;vergleichen
2FB3  CA 51 2F            jp    z,WLOOP       ;Rand erreicht
2FB6  79                  ld    a,c           ;Cursor vorwaerts
2FB7  CD 5A BB            call  TXTOUT        ;ausfuehren
2FBA  C3 51 2F            jp    WLOOP         ;und weitermachen

2FBD  3E 08         WRR    ld    a,08          ;Backspace
2FBF  CD 5A BB             call  TXTOUT       ;ausgeben
2FC2  C3 51 2F             jp    WLOOP        ;und weiter

2FC5  32 CF 2F      COPY   ld    (CTASTE),a    ;Flag setzen
2FC8  C9                   ret                ;und zurueck ins Basic

                  ;Speicherstellen fuer POKEn von Basic aus

2FC9  01            ZEILE  defb  &01
2FCA  01            SPALTE defb  &01
2FCB  01            ANZAHL defb  &01
2FCC  07            STREAM defb  &07
2FCD  05            LINKSM defb  &05
2FCE  06            RECHTSM defb &06
2FCF  (0001)        CTASTE defs  1
2FD0  (0002)        RAMDI  defs  2
```

Fehler: 00000 Warnungen: 00000

Symboltabelle:

```
2F07 ARBEIT   2FCB ANZAHL   BB7E CURDI    BB7B CUREN    2FC5 COPY
2FCF CTASTE   BB78 GETCUR   2F91 LINKS    2FCD LINKSM   BB8A PLACUR
BB60 RDCHAR   BB8D REMCUR   2EF6 RAMRD    2F2C RAMWR    2F32 RAMLES
2FA7 RECHTS   2FCE RECHTSM  2FD0 RAMDI    BB6F SETCOL   BB72 SETROW
```

```
BBB4 STRSEL   BB75 SETCUR   2FCA SPALTE   2FCC STREAM   BB5A TXTOUT
BB5D WRCHAR   BBØ6 WAITCH   2F4B WED       2F51 WLOOP    2FBD WRR
2FC9 ZEILE
```

7.3.8 Assembler-Listing: FILLER.EDI

```
                    ;*********************************************
                    ;*                FILLER.EDI                 *
                    ;*        (c) 1986  by Lothar Miedel         *
                    ;*-------------------------------------------*
                    ;* Routine um den durch RAMDISK (Anfang) und *
                    ;* MEMHIGH (Ende) festgelegten Speicher-     *
                    ;* bereich mit einem von Basic aus zu ueber- *
                    ;* gebenden Zeichen (FCHAR) aufzufuellen.    *
                    ;*********************************************

;FCHAR  =Merker fuer das von Basic uebergebene Fuellzeichen
;FILLER =Einsprungadresse des Fuellprogrammes
;FUELL  =Einsprung fuer Fuellschleife
;RAMDI  =Merker fuer Zeiger auf RAM-Disk-Adresse diese Speicherstelle wird
;        auch beim Programm WED verwendet und stellt dort die letzten
;        beiden Bytes dar!

                    org   &2fdØ
RAMDI   defs  2


                    org   &2fd2

 2FD2  2A DØ 2F     FILLER  ld    hl,(RAMDI)    ;Startadresse der RAM-Disk
 2FD5  2B                   dec   hl            ;erniedrigen und zum
 2FD6  E5                   push  hl            ;Austausch retten
 2FD7  C1                   pop   bc            ;Austausch = BC uebernehmen
 2FD8  2A F4 2F             ld    hl,(memhig)   ;Endadresse der RAM-Disk
 2FDB  E5                   push  hl            ;retten
 2FDC  ED 42                sbc   hl,bc         ;Subtraktion
 2FDE  E5                   push  hl            ;fuer Austausch retten
 2FDF  C1                   pop   bc            ;austauschen (Laenge in BC)
 2FEØ  E1                   pop   hl            ;Endadresse

 2FE1  3A F6 2F     FUELL   ld    a,(FCHAR)     ;Fuellerzeichen
 2FE4  77                   ld    (hl),a        ;in RAM-Diskadresse schreiben
 2FE5  2B                   dec   hl            ;Adresse erniedrigen
 2FE6  ØB                   dec   bc            ;Laenge erniedrigen
 2FE7  78                   ld    a,b           ;pruefen ob
 2FE8  FE ØØ                cp    Ø             ;das Ende erreicht
```

```
 2FEA  C2 E1 2F              jp    nz,FUELL     ;nein - weitermachen
 2FED  79                    ld    a,c          ;weitere Pruefung
 2FEE  FE ØØ                 cp    Ø            ;ob Ende erreicht
 2FFØ  C8                    ret   z            ;Ja -> Basic
 2FF1  C3 E1 2F              jp    FUELL        ;nein - weitermachen

 2FF4  (ØØØ2)          MEMHIG  defs  2
 2FF6  2E ØØ           FCHAR   defw  "."

Fehler: ØØØØØ  Warnungen: ØØØØØ

Symboltabelle:

2FD2 FILLER    2FE1 FUELL   2FF6 FCHAR   2FF4 MEMHIG   2FDØ RAMDI
```

7.4 Programmbeschreibung RAMDISK2
(Für den CPC 6128)

Dieses zweite Programm für den RAM-Disk-Betrieb läuft nur auf dem CPC 6128, da es die zusätzlichen RAM-Bänke dieses CPC-Typs benutzt.

Der Programmaufbau ist dem von RAMDISK1.BAS sehr ähnlich, das heißt aus diesem wurde es auch entwickelt. Warum sollte man das Rad zweimal erfinden? Da jedoch zuviele Details unterschiedlich sind, finden Sie es hier komplett abgedruckt.

Für das Verstehen dieses Programmes sollten Sie deshalb die Beschreibung von RAMDISK1.BAS lesen, denn hier finden Sie nur einige Unterschiede aufgeführt.

Im Gegensatz zum anderen Programm für einen RAM-Disk-Betrieb haben wir bei diesem nun mehr Speicherplatz zur Verfügung, denn es ist ja speziell für den CPC 6128 und der hat ja zusätzliche 64 KByte RAM-Speicher.

Da dieses Programm aber ohne den Bankmanager arbeiten soll, muß eine eigene Routine in Maschinensprache weiterhelfen, denn unsinnigerweise wurde der Bankmanager von Schneider oder Amstrad geschützt und ist für reine Bankumschaltungen auch viel zu lang. Wer Datei-Verwaltungen mittels des Bankmanagers aufbauen will, findet außerdem im Handbuch ein entsprechendes Programmbeispiel (ANAGRAMS, Kapitel 8 des 6128 Handbuches).

Derjenige hat dann im Regelfall auch nur 64 KByte-Datenspeicher, mit diesem Programm aber 8Ø!

Wenn im Laufe dieser Beschreibung von "bankswitching" gesprochen wird, dann ist eigentlich "blockswitching" gemeint, aber in Anlehnung an das CPC-Handbuch, in dem aufgeführt ist, daß der Vorgang zur Blockauswahl "bankswitching" heißt, habe ich mich an diese Definition gehalten.

Doch nun zu den wichtigen Punkten des Programmes.

Die Zeilen bis einschließlich 43Ø sind im Prinzip mit denen des ersten RAM-Disk-Programmes identisch, nur mußte beim ersten Programm komprimiert werden, da sonst der Speicherplatz nicht gereicht hätte. In Zeile

430 ist jedoch eine Maschinensprache-Einsprungadresse hinzugekommen, nämlich die Aufrufadresse des "Bankschalters".

Diese liegt bei &81BD. Und damit fällt auch gleich auf, daß im Gegensatz zu den Maschinenprogrammadressen des vorigen Programmes nun ein anderer Bereich gewählt wurde.

Das hat Gründe. Die Bankumschaltung beim CPC 6128 funktioniert im Normalfalle so, daß die gewünschte Bank in den Bereich von &4000 bis &7FFF gebracht wird. Da dieses Programm nun alle vier Blöcke (so nennt man einen 16-KByte-Bereich) der zweiten Bank, plus den Block der "Normalbank" verwenden soll, wurde in Zeile 190 der Speicher auf &3FFF begrenzt. Der Bereich oberhalb der Datenbänke steht für Maschinenprogramme zur Verfügung.

Die Maschinenroutinen sind zwar die gleichen, wie beim Programm RAMDISK1, aber der Bereich ist ein anderer. Das hat, weil die Routinen auch für diesen Bereich assembliert abgedruckt sind, für Sie den Vorteil, daß Sie fertige Routinen in zwei Adreßbereichen haben. Sie könnene diese nun wahlweise (je nach Problem) in anderen Programmen einsetzen.

Deshalb haben die Adressen in den Zeilen 370, 390, 400, 410 und 430 als erste Ziffer eine Acht.

Eine weitere Abweichung vom Schema stellen die Zeilen 500 bis 540 dar. In jeder wird eine bestimmte Bank ausgewählt und diese Bank mit dem Füllzeichen beschrieben. Damit wird sichergestellt, daß beim erstmaligen Arbeiten mit diesem Programm - also wenn noch keine Daten vorhanden sind - nicht irgendwelcher "Blödsinn" in den Blöcken enthalten ist. Wer will kann diese Zeilen später entfernen, denn wenn Daten vorliegen, können diese ja auch geladen werden.

In den Zeile 1160 und 1170 wird die aus der Eintragsnummer sich ergebende Bank berechnet. Ebenso die Block-Eintragsnummer "benr".

In Zeile 1580 wird die "alte Bank" in der Variablen "altbank" aufgehoben, um bei einem negativen Suchvorgang (F = FIND) auf dem aktuellen Eintrag stehen zu bleiben.

Die Variable "stand" in 159Ø beinhaltet die "Beginnadresse" für die Suche, da der Offset addiert wird.

Die Variable "rest" in 162Ø beinhaltet die Anzahl der zu durchsuchenden Bytes bis zum Ende des Blockes (&7FFF).

Beim eigentlichen Suchvorgang (Zeile 167Ø) wird immer nur eine Bank durchsucht, deshalb erfolgt durch die Zeilen 169Ø bis 177Ø die definierte Umschaltung auf die anderen Blöcke und die Suche in den anderen Bänken.

Ist eine Suche erfolglos, dann bleibt das Programm auf der Eintragsnummer stehen, die bei Beginn des Suchvorganges aktuell war. Hierzu dient Zeile 17ØØ.

Die Variable "altbank" aus Zeile 158Ø wird eigentlich nicht benötigt, kann aber bei einem noch nicht fehlerfreien Programm als Hilfsmittel dienen. Deshalb ist sie noch im Programm!

Das Laden und Speichern der 16 KByte-Blöcke geschieht in den Zeilen 191Ø bis 2Ø8Ø. Die einzelnen Blöcke werden als getrennte Files behandelt.

Das ist eigentlich alles, was zusätzlich zu diesem Basic-Teil erwähnt werden musste.

7.4.1 Basic-Listing: RAMDISK2.BAS für den CPC 6128

```
1ØØ '****************************************
11Ø '*                                      *
12Ø '*            RAMDISK2.BAS              *
13Ø '*______________________________________*
14Ø '*                                      *
15Ø '*   Copyright by Lothar Miedel 1986    *
16Ø '*                                      *
17Ø '****************************************
18Ø '
19Ø MODE 2:MEMORY &3FFF:basis=1
2ØØ '
21Ø 'Fenster definieren
22Ø WINDOW #Ø,2,79,5,22:WINDOW #1,2,79,1,1
23Ø WINDOW #2,61,78,1,1:WINDOW #3,2,78,3,3
24Ø WINDOW #4,6Ø,78,24,24: WINDOW #5,2,5Ø,24,24
25Ø WINDOW #6,6Ø,78,15,2Ø
26Ø '
```

```
270 'Rahmen zeichnen
280 DRAW 0,380:DRAW 639,380:DRAW 639,349:DRAW 0,349:DRAW 0,35
290 DRAW 639,35:DRAW 639,379:DRAW 639,0:DRAW 0,0
300 '------------------------------------------------------------
310 PRINT#1,CHR$(24) "Dateiverwaltung mit 6128-RAM-DISK"CHR$(24)
320 PRINT#3,"Lade das Maschinenprogramm":LOAD"ramdisk2.mpg",&8000
325 PRINT #3,"Bitte Datendisk einlegen und Taste druecken!":CALL &BB06
330 GOSUB 580:'Maskendaten holen
340 '
350 'Definitionen
360 inv$=CHR$(24):norm$=inv$
370 zeilem=&818E:mspalte=zeilem+1:manzahl=zeilem+2:stream=zeilem+3
380 linksm=stream+1:rechtsm=stream+2:memhig=&81B9:fchar=&81BB
390 filler=&8197:zeichen=&2E:ramdi=zeilem+7:ctaste=&8194
400 ramrd=&80BB:wed=&8110:ramwr=&80F1:fundpos=&8077:suchwort =&807F
410 suche =&8000:ramlen=&8071:laenge=&8073:aktstart=&8075
420 ramlow=VAL("&"+MID$(buffanf$,3,2)):ramhigh=VAL("&"+MID$(buffanf$,1,2))
430 zeile=1:spalte=1:anzahl=56:str=7:bswitch=&81BD
440 '************************************************************
450 ' Die Bloecke mit dem Zeichen fchar belegen
460 POKE memhig,VAL("&"+MID$(buffend$,3,2))
470 POKE memhig+1,VAL("&"+MID$(buffend$,1,2))
480 POKE ramdi,ramlow:POKE ramdi+1,ramhigh
490 POKE fchar,zeichen
500 CALL bswitch,4:CALL filler:' Zusatzbank 1
510 CALL bswitch,5:CALL filler:' Zusatzbank 2
520 CALL bswitch,6:CALL filler:' Zusatzbank 3
530 CALL bswitch,7:CALL filler:' Zusatzbank 4
540 CALL bswitch,0:CALL filler
550 GOTO 760
560 '------------------------------------------------------------
570 'Maskendaten lesen
580 PRINT#3,"Lese die Maskendaten
590 OPENIN"ramdisk2.msk"
600  INPUT #9,buffanf$,buffend$,eintraege,off,felder
610 buffanf=VAL("&"+buffanf$):buffend=VAL("&"+buffend$):diff=buffend-buffanf+1
620 IF diff <0 THEN diff=diff+2^16
630 DIM fenam$(felder),feldlen(felder),xposi(felder),yposi(felder)
640 DIM feldinh$(felder)
650   FOR i = 1 TO felder
660     INPUT #9,fenam$(i)
670     INPUT #9,feldlen(i)
680     benutzt=benutzt+feldlen(i)
690     INPUT #9,xposi(i)
700     INPUT #9,yposi(i)
710   NEXT i
720 anzprobank=INT(eintraege/5)
730 CLOSEIN:PRINT#3,inv$ "Maske gelesen - Ausgabe der Maske" norm$
```

```
740 RETURN
750 '-------------------------------------------------------------
760  FOR j = 1 TO felder
770       LOCATE xposi(j),yposi(j)
780       PRINT fenam$(j)
790 NEXT j
800 '-------------------------------------------------------------
810 PRINT #3,"Auf geht's !
820 LOCATE #1,47,1:PRINT#1,"Dateieintraege: "eintraege"*"off
830 nr=basis:GOTO 1090
840 '
850 PRINT #3,inv$"GOTO ...."inv$
860 INPUT #5,"Eintragsnummer";nr$:enr$=LEFT$(nr$,6)
870 IF LEN(nr$)<1 THEN 860
880 IF ASC(LEFT$(nr$,1))>57 THEN 860
890 IF ASC(LEFT$(nr$,1))<48 THEN 860
900 nr=VAL(nr$):GOTO 1090
910 '
920 CLS #3:PRINT #3,inv$ "Daten gueltig! Bitte Ihre Eingabe!" norm$
930 CLS #5:PRINT #5,"Eintragsnummer: "nr
940 CLS #4
950 '
960 x$=LOWER$(INKEY$):IF x$="" THEN 960
970 IF x$="e" THEN 2090
980 IF x$="f" THEN 1530
990 IF x$="g" THEN 850
1000 IF x$="l" THEN 1910
1010 IF x$="s" THEN 2000
1020 IF ASC(x$)=224 THEN 1350
1030 IF ASC(x$)<240 THEN 960
1040 IF ASC(x$)>243 THEN 960
1050 IF ASC(x$)=243 THEN nr=nr+1
1060 IF ASC(x$)=242 THEN nr=nr-1
1070 IF ASC(x$)=240 THEN nr=nr+10
1080 IF ASC(x$)=241 THEN nr=nr-10
1090 '
1100 IF nr<basis THEN GOSUB 1460:nr=basis
1110 IF nr-basis+1>=5*anzprobank THEN nr=5*anzprobank+basis-1
1120 enr=nr-basis
1130 '
1140 CLS #3:PRINT #3,inv$"Moment bitte!"inv$
1150 IF enr<anzprobank THEN bank=0:GOTO 1170
1160 bank=INT(enr/anzprobank) + 3 MOD anzprobank
1170 benr=(enr MOD anzprobank):enr=(enr MOD anzprobank)
1180 '
1190 'Berechne Bufferoffset und Beginnadresse
1200 buffoff=(benr)*off
1210 IF buffoff <0 THEN buffoff=0
```

```
1220 begad=VAL("&"+buffanf$)+buffoff:'  Datenbuffer festlegen
1230 '
1240 'Leseschleife
1250 IF bank > 7 THEN bank = 0
1260 CALL bswitch,bank
1270 POKE stream,str:zz=0:FOR i = 1 TO felder
1280 beginn=begad+zz:zz=zz+feldlen(i)
1290 zeile=yposi(i):anzahl=feldlen(i)-1
1300 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
1310 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+16,zeile+4,zeile+4:CLS #7
1320 GOSUB 1830:CALL ramwr:NEXT i:GOTO 920
1330 '-----------------------------------------
1340 'Aendern
1350 PRINT #3,inv$"Bitte Ihre Eingaben!"inv$:PRINT #5,"Aendern":PRINT #4,nr
1360 POKE stream,str:zz=0
1370 FOR i = 1 TO felder
1380 beginn=begad+zz:zz=zz+feldlen(i):zeile=yposi(i):anzahl=feldlen(i)-1
1390 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
1400 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+17,zeile+4,zeile+4
1410 GOSUB 1890:CALL WED
1420 '
1430 anzahl=feldlen(i):GOSUB 1830:CALL ramrd:NEXT i:PRINT #5:GOTO 920
1440 '
1450 'Fehlerbehandlung
1460 PRINT #3,inv$ "Fehler! "norm$;
1470 PRINT #5,inv$" Bitte Taste 'j' oder 'n' druecken!"inv$:PRINT CHR$(7);
1480 x$=LOWER$(INKEY$):IF x$="" THEN 1480
1490 IF x$<> "j" AND x$<> "n" THEN 1480 ELSE CLS #5:RETURN
1500 CLS #3
1510 RETURN
1520 '-------------------------------------------------------------
1530 PRINT #3,inv$"Suchen: "inv$"  Bitte geben Sie die Zeichenfolge ein!  "
1540 INPUT #5,such$:IF LEN (such$)=0 THEN sw$=sw$ ELSE sw$=such$
1550 wlaenge=LEN(sw$):IF wlaenge>56 THEN sw$=LEFT$(sw$,56):GOTO 1550
1560 FOR i=1 TO wlaenge:POKE suchwort+i-1,ASC(MID$(sw$,i,1)):NEXT i
1570 POKE laenge,wlaenge
1580 altbank=bank:altnr=nr
1590 stand=begad+off
1600 '
1610 sthi=INT(stand/256):stlo=stand-sthi*256
1620 rest=buffend-stand
1630 ramlhi=INT(rest/256):ramllo=rest-ramlhi*256
1640 '
1650 POKE aktstart,stlo:POKE aktstart+1,sthi
1660 POKE ramlen,ramllo:POKE ramlen+1,ramlhi
1670 CALL suche:fs=PEEK(fundpos)+PEEK(fundpos+1)*256
1680 '
1690 IF fs>&7FFF THEN fs=0
```

```
1700 IF bank=7 THEN IF fs=0 THEN nr=altnr:GOTO 1800
1710 IF fs=0 THEN 1760
1720 enr=INT((fs-buffanf)/off)+basis
1730 IF bank =0 THEN nr=enr
1740 IF bank<>0 THEN nr=enr+anzprobank*(bank-3)
1750 IF enr>0 THEN 1780
1760 IF bank <7 THEN bank = bank+1:IF bank <4 THEN 1760
1770 CALL bswitch,bank:IF bank <> altbank THEN stand=&4000:GOTO 1610
1780 '
1790 GOTO 1090
1800 PRINT #3,inv$"Nicht gefunden !"inv$:GOSUB 1470:GOTO 1090
1810 '-------------------------------------------------------------
1820 'Lesen aus Window 7 und schreiben in RAM-Disk
1830 zeile=1:spalte=1:POKE zeilem,zeile
1840 POKE manzahl,anzahl:POKE mspalte,spalte
1850 POKE ramdi,ramlow:POKE ramdi+1,ramhigh
1860 IF PEEK(ctaste)=224 THEN i = felder
1870 POKE ctaste,0:RETURN
1880 '
1890 links=16:rechts=feldlen(i)
1900 POKE stream,str:POKE linksm,1:POKE rechtsm,rechts:RETURN
1910 PRINT#3,inv$"Datenfile laden"inv$:GOSUB 1470
1920 IF x$<>"j" THEN 1090
1930 CALL bswitch,0:LOAD"ramdisk2.da1",buffanf
1940 CALL bswitch,4:LOAD"ramdisk2.da2",buffanf
1950 CALL bswitch,5:LOAD"ramdisk2.da3",buffanf
1960 CALL bswitch,6:LOAD"ramdisk2.da4",buffanf
1970 CALL bswitch,7:LOAD"ramdisk2.da5",buffanf
1980 CALL bswitch,0
1990 GOTO 1090
2000 PRINT#3,inv$"Datenfiles speichern"inv$:GOSUB 1470
2010 IF x$<>"j" THEN 1090
2020 CALL bswitch,0:SAVE"ramdisk2.da1",b,&4000,&3FFF
2030 CALL bswitch,4:SAVE"ramdisk2.da2",b,&4000,&3FFF
2040 CALL bswitch,5:SAVE"ramdisk2.da3",b,&4000,&3FFF
2050 CALL bswitch,6:SAVE"ramdisk2.da4",b,&4000,&3FFF
2060 CALL bswitch,7:SAVE"ramdisk2.da5",b,&4000,&3FFF
2070 CALL bswitch,0
2080 GOTO 1090
2090 PRINT #3,"Programm beenden ???"
2100 GOSUB 1470
2110 IF x$="n" THEN GOTO 1090
2120 IF x$="j" THEN MODE 2:END
```

Um die Maschinenprogramme zu erzeugen, gibt es die nachfolgenden Hexlader. Für das Programm selbst ist nur der mit der Bezeichnung RAMDISK2.HEX (7.4.6) erforderlich, die anderen können als Hilfe für Ihre eigenen Programme dienen.

7.4.2 Hexlader: SUCHER2.HEX für den CPC 6128

```
100 REM SUCHER2.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 32768:e= 32881:zb=1001
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"sucher2.bin",b,&8000,&70
240 '
1001 DATA ED,4B,71,80,11,7F,80,2A,75,80,22,79,80,3A,73,80,&06A0
1002 DATA 32,74,80,1A,ED,B1,C2,6A,80,2B,22,75,80,22,7D,80,&06EB
1003 DATA 22,79,80,3A,73,80,32,74,80,1A,ED,B1,C2,6A,80,D5,&07A7
1004 DATA E5,2A,79,80,23,E5,D1,E1,E5,ED,52,E1,D1,22,79,80,&09B3
1005 DATA CA,4D,80,2A,75,80,23,22,75,80,C3,00,80,22,7B,80,&0650
1006 DATA 3A,74,80,3D,32,74,80,FE,00,CA,60,80,13,C3,29,80,&06B8
1007 DATA ED,5B,73,80,ED,52,22,77,80,C9,21,00,00,22,77,80,&0696
1008 DATA C9,&00C9
```

7.4.3 Hexlader: RD2.HEX für den CPC 6128

```
100 REM RD2.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 32955:e= 33175:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
```

```
220 '
230 SAVE"rd2.bin",b,&80BB,&DB
240 '
2000 DATA 3A,91,81,CD,B4,BB,3A,8E,81,F5,CD,72,BB,F1,32,8E,&0971
2001 DATA 81,3A,8F,81,F5,CD,6F,BB,F1,3C,32,8F,81,2A,90,81,&0861
2002 DATA 7D,FE,00,C8,2B,22,90,81,CD,60,BB,2A,95,81,77,23,&0763
2003 DATA 22,95,81,C3,CC,80,3A,91,81,CD,B4,BB,2A,95,81,7E,&088D
2004 DATA 23,22,95,81,CD,5D,BB,2A,90,81,7D,FE,00,C8,2B,22,&070B
2005 DATA 90,81,C3,F7,80,3A,91,81,CD,B4,BB,CD,78,BB,CD,75,&0A15
2006 DATA BB,CD,8A,BB,CD,06,BB,4F,CD,8D,BB,79,FE,0D,C8,FE,&0A09
2007 DATA E0,CA,8A,81,FE,F2,CA,56,81,FE,F3,CA,6C,81,FE,7F,&0B6B
2008 DATA F2,16,81,FE,20,FA,16,81,CD,5A,BB,CD,78,BB,3A,93,&08E7
2009 DATA 81,3C,57,7C,BA,CA,82,81,C3,16,81,3E,08.4F,CD,78,&074B
2010 DATA BB,3A,92,81,57,7C,BA,CA,16,81,79,CD,5A,BB,C3,16,&082A
2011 DATA 81,3E,09,4F,CD,78,BB,3A,93,81,57,7C,BA,CA,16,81,&0753
2012 DATA 79,CD,5A,BB,C3,16,81,3E,08,CD,5A,BB,C3,16,81,32,&0769
2013 DATA 94,81,C9,01,01,01,07,05,06,00,00,00,&01F3
```

7.4.4 Hexlader: FILLER2.HEX für den CPC 6128

```
100 REM FILLER2.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 33175:e= 33213:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"filler2.bin",b,&8197,&25
240 '
2000 DATA 2A,95,81,2B,E5,C1,2A,B9,81,E5,ED,42,E5,C1,E1,3A,&094A
2001 DATA BB,81,77,2B,0B,78,FE,00,C2,A6,81,79,FE,00,C8,C3,&084A
2002 DATA A6,81,00,00,2E,00,&0155
```

7.4.5 Hexlader: BSWITCH.HEX für den CPC 6128

```
100 REM BSWITCH.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 33213:e= 33220:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
```

```
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"bswitch.bin",b,&81BD,6
240 '
2000 DATA dd,7e,00,cd,5b,bd,c9,&0409
```

Der Hexlader für das komplette Maschinenprogramm der Dateiverwaltung mit RAM-Disk für den CPC 6128.

7.4.6 Hexlader: RAMDISK2.HEX

```
100 REM RAMDISK2.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 32768:e= 33220:zb=1001
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"ramdisk2.mpg",b,&8000,&1C4
240 '
1001 DATA ED,4B,71,80,11,7F,80,2A,75,80,22,79,80,3A,73,80,&06A0
1002 DATA 32,74,80,1A,ED,B1,C2,6A,80,2B,22,75,80,22,7D,80,&06EB
1003 DATA 22,79,80,3A,73,80,32,74,80,1A,ED,B1,C2,6A,80,D5,&07A7
1004 DATA E5,2A,79,80,23,E5,D1,E1,E5,ED,52,E1,D1,22,79,80,&09B3
1005 DATA CA,4D,80,2A,75,80,23,22,75,80,C3,00,80,22,7B,80,&0650
1006 DATA 3A,74,80,3D,32,74,80,FE,00,CA,60,80,13,C3,29,80,&06B8
1007 DATA ED,5B,73,80,ED,52,22,77,80,C9,21,00,00,22,77,80,&0696
1008 DATA C9,00,00,00,00,00,00,00,00,00,00,00,00,C0,00,00,&00C9
1009 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
1010 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
1011 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
1012 DATA 00,00,00,00,00,00,00,00,00,00,00,3A,91,81,CD,B4,&02CD
1013 DATA BB,3A,8E,81,F5,CD,72,BB,F1,32,8E,81,3A,8F,81,F5,&0964
1014 DATA CD,6F,BB,F1,3C,32,8F,81,2A,90,81,7D,FE,00,C8,2B,&080F
1015 DATA 22,90,81,CD,60,BB,2A,95,81,77,23,22,95,81,C3,CC,&07BC
```

```
1Ø16 DATA 8Ø,3A,91,81,CD,B4,BB,2A,95,81,7E,23,22,95,81,CD,&Ø7EE
1Ø17 DATA 5D,BB,2A,9Ø,81,7D,FE,ØØ,C8,2B,22,9Ø,81,C3,F7,8Ø,&Ø82E
1Ø18 DATA 3A,91,81,CD,B4,BB,CD,78,BB,CD,75,BB,CD,8A,BB,CD,&ØA64
1Ø19 DATA Ø6,BB,4F,CD,8D,BB,79,FE,ØD,C8,FE,EØ,CA,8A,81,FE,&ØA22
1Ø2Ø DATA F2,CA,56,81,FE,F3,CA,6C,81,FE,7F,F2,16,81,FE,2Ø,&ØA5F
1Ø21 DATA FA,16,81,CD,5A,BB,CD,78,BB,3A,93,81,3C,57,7C,BA,&Ø88A
1Ø22 DATA CA,82,81,C3,16,81,3E,Ø8,4F,CD,78,BB,3A,92,81,57,&Ø76Ø
1Ø23 DATA 7C,BA,CA,16,81,79,CD,5A,BB,C3,16,81,3E,Ø9,4F,CD,&Ø7AF
1Ø24 DATA 78,BB,3A,93,81,57,7C,BA,CA,16,81,79,CD,5A,BB,C3,&Ø88D
1Ø25 DATA 16,81,3E,Ø8,CD,5A,BB,C3,16,81,32,94,81,C9,Ø1,Ø1,&Ø62B
1Ø26 DATA Ø1,Ø7,Ø5,Ø6,ØØ,ØØ,ØØ,2A,95,81,2B,E5,C1,2A,B9,81,&Ø488
1Ø27 DATA E5,ED,42,E5,C1,E1,3A,BB,81,77,2B,ØB,78,FE,ØØ,C2,&Ø8F6
1Ø28 DATA A6,81,79,FE,ØØ,C8,C3,A6,81,ØØ,ØØ,2E,ØØ,DD,7E,ØØ,&Ø6D9
1Ø29 DATA CD,5B,BD,C9,&Ø2AE
```

Nun folgen noch die Assembler-Listings, auch diesmal komplett, obwohl außer der Adressen-Änderung nur der Bankschalter hinzukam. Aber durch den kompletten Abdruck haben Sie bessere Vergleichsmöglichkeiten bei Fehlern und vor allem helfen diese auch besser zum Verstehen der Programme. Sie können also die Adressen des Basiclistings direkt umsetzen.

Entfallen sind aber die Erklärungen der Labels, diese können bei den Assembler-Listings von RAMDISK1 nachgelesen werden.

7.4.7 Assembler-Listing: SUCHER.EDI (6128)

```
                        ;*****************************************
                        ;*                SUCHER.EDI             *
                        ;*       (c) 1986 by Lothar Miedel       *
                        ;*---------------------------------------*
                        ;* Dieses Maschinen-Programm sucht Woerter *
                        ;* (=ASCII-Folgen) im Speicher des CPC     *
                        ;*****************************************

                         org   &8ØØØ

 8ØØØ  ED 4B 71 8Ø   START   ld    bc,(RAMLEN)  ;Laenge Suchbereich
 8ØØ4  11 7F 8Ø              ld    de,SUCHW     ;Adresse des 1. Buchstaben
 8ØØ7  2A 75 8Ø              ld    hl,(AKSTART) ;Startadresse der Suche
 8ØØA  22 79 8Ø              ld    (ALTMERK),hl ;Merker alter Suchbeginn

 8ØØD  3A 73 8Ø      ERSTPR  ld    a,(LAENGE)   ;Laenge des Wortes
 8Ø1Ø  32 74 8Ø              ld    (RESTLEN),a  ;in RESTLEN schreiben
 8Ø13  1A                    ld    a,(de)       ;ersten Buchstaben laden
```

```
8014  ED B1              cpir                ;und suchen
8016  C2 6A 80           jp    nz,NOERFOL    ;nichts gefunden
8019  2B                 dec   hl           ;Startadresse erniedrigen
801A  22 75 80           ld    (AKSTART),hl  ;aktueller Start
801D  22 7D 80           ld    (ERSTBYT),hl  ;Adresse erstes Bytes
8020  22 79 80           ld    (ALTMERK),hl  ;auch merken
8023  3A 73 80           ld    a,(LAENGE)    ;Laenge des Wortes
8026  32 74 80           ld    (RESTLEN),a   ;in RESTLEN merken

8029  1A           SUCH2  ld    a,(de)        ;Suchzeichen laden
802A  ED B1              cpir                ;suchen
802C  C2 6A 80           jp    nz,NOERFOL    ;nicht gefunden

802F  D5           PRUEF  push  de            ;Position im Wort retten
8030  E5                 push  hl            ;gefundene Position + 1
8031  2A 79 80           ld    hl,(ALTMERK)  ;Merker laden
8034  23                 inc   hl            ;und fuer SBC erhoehen
8035  E5                 push  hl            ;de mit
8036  D1                 pop   de            ;(ALTMERK)+1 laden
8037  E1                 pop   hl            ;gefundene Position +1
8038  E5                 push  hl            ;merken
8039  ED 52              sbc   hl,de         ;subtrahieren
803B  E1                 pop   hl            ;Position + 1 wieder holen
803C  D1                 pop   de            ;Position im Wort
803D  22 79 80           ld    (ALTMERK),hl  ;als alten Wert merken
8040  CA 4D 80           jp    z,KOMPL?      ;ist Wort komplett?
8043  2A 75 80           ld    hl,(AKSTART)  ;Zeichenfolge falsch
8046  23                 inc   hl            ;ein Byte weiter
8047  22 75 80           ld    (AKSTART),hl  ;merken und dort
804A  C3 00 80           jp    START         ;Suche fortsetzen

804D  22 7B 80     KOMPL? ld    (EFIND),hl    ;erste Fundadresse
8050  3A 74 80           ld    a,(RESTLEN)   ;Restlaenge holen
8053  3D                 dec   a             ;Zeichen stimmt in der
8054  32 74 80           ld    (RESTLEN),a   ;Folge - merken!
8057  FE 00              cp    a,0           ;komplettes Wort gefunden?
8059  CA 60 80           jp    z,ERFOLG      ;ja - Suche erfolgreich
805C  13                 inc   de            ;noch nicht alle Zeichen
805D  C3 29 80           jp    SUCH2         ;deshalb Suche fortsetzen

8060  ED 5B 73 80  ERFOLG ld    de,(LAENGE)   ;Laenge des Wortes
8064  ED 52              sbc   hl,de         ;subtrahieren
8066  22 77 80           ld    (FUNDPOS),hl  ;Beginn des Wortes
8069  C9                 ret                 ;zurueck nach Basic

806A  21 00 00     NOERFOL ld   hl,&0000     ;nichts gefunden - keine
806D  22 77 80           ld    (FUNDPOS),hl  ;Fundposition fuer Basic
8070  C9                 ret                 ;zurueck nach Basic
```

```
 8071  (0002)          RAMLEN  defs  2
 8073  (0001)          LAENGE  defs  1
 8074  (0001)          RESTLEN defs  1
 8075  (0002)          AKSTART defs  2
 8077  (0002)          FUNDPOS defs  2
 8079  (0002)          ALTMERK defs  2
 807B  (0002)          EFIND   defs  2
 807D  (0002)          ERSTBYT defs  2
 807F  (003C)          SUCHW   defs  60

 80BB  (80BB)                  end

Fehler: 00000  Warnungen: 00000

Symboltabelle:

8075 AKSTART  8079 ALTMERK  800D ERSTPR  8060 ERFOLG  807B EFIND   807D ERSTBYT
8077 FUNDPOS  804D KOMPL?   8073 LAENGE  806A NOERFOL 802F PRUEF   8071 RAMLEN
8074 RESTLEN  8000 START    8029 SUCH2   807F SUCHW
```

7.4.8 Assembler-Listing: RD.EDI (6128)

```
                    ;****************************************
                    ;*                RD.EDI                *
                    ;* Maschinenprogramme fuer RAMDISK.BAS *
                    ;*                                      *
                    ;****************************************

                    org   &80bb
;Definitionen

CURDI   equ   &bb7e:CUREN   equ   &bb7b:RDCHAR  equ   &bbb0
SETCOL  equ   &bb6f:SETROW  equ   &bb72:STRSEL  equ   &bbb4
WRCHAR  equ   &bb5d:WAITCH  equ   &bb06:TXTOUT  equ   &bb5a
SETCUR  equ   &bb75:PLACUR  equ   &bb8a:REMCUR  equ   &bb8d
GETCUR  equ   &bb78
;
                    ;******************* RAMRD ******************
                    ;* Lesen von Bildschirmzeichen und schreiben *
                    ;*                in RAM-Disk                *
                    ;********************************************

 80BB  3A 91 81      RAMRD   ld    a,(STREAM)   ;Streamauswahl
 80BE  CD B4 BB              call  STRSEL       ;aufrufen
 80C1  3A 8E 81              ld    a,(ZEILE)    ;Zeilenposition
 80C4  F5                    push  af           ;retten
```

```
80C5  CD 72 BB            call  SETROW       ;positionieren
80C8  F1                  pop   af           ;Position wieder
80C9  32 8E 81            ld    (ZEILE),a    ;rueckspeichern

80CC  3A 8F 81    ARBEIT  ld    a,(SPALTE)   ;Spalte holen
80CF  F5                  push  af           ;und merken
80D0  CD 6F BB            call  SETCOL       ;Spalte positionieren
80D3  F1                  pop   af           ;wieder holen
80D4  3C                  inc   a            ;erhoehen
80D5  32 8F 81            ld    (SPALTE),a   ;neue Spalte merken

80D8  2A 90 81            ld    hl,(ANZAHL)  ;verbliebene Zeichen
80DB  7D                  ld    a,l          ;Anzahl zum Vergleich in Akku
80DC  FE 00               cp    0            ;Ende ?
80DE  C8                  ret   z            ;ja -> Basic

80DF  2B                  dec   hl           ;Zeichenanzahl erniedrigen
80E0  22 90 81            ld    (ANZAHL),hl  ;neue Anzahl merken

80E3  CD 60 BB            call  RDCHAR       ;Lesen und Zeichen in den Akku
80E6  2A 95 81            ld    hl,(RAMDI)   ;RAM-Disk-Adresse holen und
80E9  77                  ld    (hl),a       ;Akkuinhalt in RAM-Disk
80EA  23                  inc   hl           ;RAM-Disk-Adresse erhoehen
80EB  22 95 81            ld    (RAMDI),hl   ;und merken
80EE  C3 CC 80            jp    ARBEIT       ;weitermachen

                  ;*******************************************
                  ;************        RAMWR        ************
                  ;*  Lesen von Zeichen aus der RAM-Disk und  *
                  ;*         schreiben auf den Bildschirm     *
                  ;*******************************************

80F1  3A 91 81    RAMWR   ld    a,(STREAM)   ;Stream auswaehlen
80F4  CD B4 BB            call  STRSEL

80F7  2A 95 81    RAMLES  ld    hl,(RAMDI)   ;Adresse der RAM-Disk holen
80FA  7E                  ld    a,(hl)       ;Zeichen aus RAM-Disk lesen

80FB  23                  inc   hl           ;RAM-Diskadresse erhoehen
80FC  22 95 81            ld    (RAMDI),hl   ;und merken
80FF  CD 5D BB            call  WRCHAR       ;Zeichen aus Akku schreiben

8102  2A 90 81            ld    hl,(ANZAHL)  ;Zeichenanzahl holen
8105  7D                  ld    a,l          ;Anzahl in den Akku
8106  FE 00               cp    a,0          ;vergleichen auf Null
8108  C8                  ret   z            ;zurueck ins Basic

8109  2B                  dec   hl           ;Zeichenanzahl erniedrigen
```

```
810A  22 90 81            ld    (ANZAHL),hl  ;und merken
810D  C3 F7 80            jp    RAMLES

                  ;****************************************
                  ;** WED  - Window-Editor  ***************
                  ;****************************************

8110  3A 91 81     WED    ld    a,(STREAM)   ;STREAM holen
8113  CD B4 BB            call  STRSEL       ;und auswaehlen

8116  CD 78 BB     WLOOP  call  GETCUR       ;Cursorposition holen
8119  CD 75 BB            call  SETCUR       ;Cursor setzen
811C  CD 8A BB            call  PLACUR       ;Cursor auf erste Position
811F  CD 06 BB            call  WAITCH       ;Auf Zeichen warten
8122  4F                  ld    c,a          ;Zeichen retten
8123  CD 8D BB            call  REMCUR       ;Cursorsymbol entfernen
8126  79                  ld    a,c          ;Zeichen wieder holen
8127  FE 0D               cp    a,13         ;ist es ENTER (RETURN)?
8129  C8                  ret   z            ;ja -> Basic
812A  FE E0               cp    a,224        ;Copy-Taste ?
812C  CA 8A 81            jp    z,COPY       ;Sprung zur Routine Copy-Taste
812F  FE F2               cp    a,242        ;ist es Cursor links ?
8131  CA 56 81            jp    z,LINKS      ;ja -> Routine links
8134  FE F3               cp    a,243        ;ist es Cursor rechts ?
8136  CA 6C 81            jp    z,RECHTS     ;ja -> Routine rechts
8139  FE 7F               cp    a,&7f        ;Zeichen groesser &7f
813B  F2 16 81            jp    p,WLOOP      ;ja weitermachen
813E  FE 20               cp    a,32         ;Zeichen kleiner 32
8140  FA 16 81            jp    m,WLOOP      ;ja weitermachen

8143  CD 5A BB            call  TXTOUT       ;und ausgeben
8146  CD 78 BB            call  GETCUR       ;und Cursorposition holen
8149  3A 93 81            ld    a,(RECHTSM)  ;rechten Maximalwert
814C  3C                  inc   a            ;erhoehen
814D  57                  ld    d,a          ;und merken
814E  7C                  ld    a,h          ;aktuelle Position holen
814F  BA                  cp    a,d          ;und vergleichen
8150  CA 82 81            jp    z,WRR        ;Windowposition rechts erreicht
8153  C3 16 81            jp    WLOOP        ;weitermachen

8156  3E 08        LINKS  ld    a,8          ;Akku mit Backspace laden
8158  4F                  ld    c,a          ;und merken
8159  CD 78 BB            call  GETCUR       ;Cursor holen
815C  3A 92 81            ld    a,(LINKSM)   ;linken Grenzwert
815F  57                  ld    d,a          ;merken und mit
8160  7C                  ld    a,h          ;aktuellem Wert
8161  BA                  cp    a,d          ;vergleichen
8162  CA 16 81            jp    z,WLOOP      ;Rand erreicht
```

```
8165  79              ld    a,c         ;Backspace
8166  CD 5A BB        call  TXTOUT      ;ausfuehren
8169  C3 16 81        jp    WLOOP       ;weitermachen

816C  3E Ø9     RECHTS  ld    a,9         ;Akku mit Cursor vorw. laden
816E  4F              ld    c,a         ;und merken
816F  CD 78 BB        call  GETCUR      ;aktuellen Wert holen
8172  3A 93 81        ld    a,(RECHTSM) ;rechten Grenzwert
8175  57              ld    d,a         ;merken
8176  7C              ld    a,h         ;aktuellen Wert
8177  BA              cp    a,d         ;vergleichen
8178  CA 16 81        jp    z,WLOOP     ;Rand erreicht
817B  79              ld    a,c         ;Cursor vorwaerts
817C  CD 5A BB        call  TXTOUT      ;ausfuehren
817F  C3 16 81        jp    WLOOP       ;und weitermachen

8182  3E Ø8     WRR    ld    a,Ø8        ;Backspace
8184  CD 5A BB        call  TXTOUT      ;ausgeben
8187  C3 16 81        jp    WLOOP       ;und weiter

818A  32 94 81   COPY   ld    (CTASTE),a  ;Flag setzen
818D  C9               ret               ;und zurueck ins Basic

                ;Speicherstellen fuer POKEn von Basic aus

818E  Ø1         ZEILE  defb  &Ø1
818F  Ø1         SPALTE defb  &Ø1
8190  Ø1         ANZAHL defb  &Ø1
8191  Ø7         STREAM defb  &Ø7
8192  Ø5         LINKSM defb  &Ø5
8193  Ø6         RECHTSM defb &Ø6
8194  (ØØØ1)     CTASTE defs  1
8195  (ØØØ2)     RAMDI  defs  2

Fehler: ØØØØØ  Warnungen: ØØØØØ

Symboltabelle:

8ØCC ARBEIT  8190 ANZAHL  BB7E CURDI   BB7B CUREN   818A COPY    8194 CTASTE
BB78 GETCUR  8156 LINKS   8192 LINKSM  BB8A PLACUR  BB6Ø RDCHAR  BB8D REMCUR
8ØBB RAMRD   8ØF1 RAMWR   8ØF7 RAMLES  816C RECHTS  8193 RECHTSM 8195 RAMDI
BB6F SETCOL  BB72 SETROW  BBB4 STRSEL  BB75 SETCUR  818F SPALTE  8191 STREAM
BB5A TXTOUT  BB5D WRCHAR  BBØ6 WAITCH  8110 WED     8116 WLOOP   8182 WRR
818E ZEILE
```

7.4.9 Assembler-Listing: FILLER.EDI (6128)

```
                    ;*******************************************
                    ;*                FILLER.EDI               *
                    ;*        (c) 1986  by Lothar Miedel       *
                    ;*-----------------------------------------*
                    ;* Routine um den durch RAMDISK (Anfang) und *
                    ;* MEMHIGH (Ende) festgelegten Speicher-     *
                    ;* bereich mit einem von Basic aus zu ueber- *
                    ;* gebenden Zeichen (FCHAR) aufzufuellen.    *
                    ;*******************************************


                    org   &8195
RAMDI   defs  2
                    org   &8197

 8197  2A 95 81     FILLER  ld    hl,(RAMDI)    ;Startadresse der RAM-Disk
 819A  2B                   dec   hl            ;erniedrigen und zum
 819B  E5                   push  hl            ;Austausch retten
 819C  C1                   pop   bc            ;Austausch = BC uebernehmen
 819D  2A B9 81             ld    hl,(memhig)   ;Endadresse der RAM-Disk
 81AØ  E5                   push  hl            ;retten
 81A1  ED 42                sbc   hl,bc         ;Subtraktion
 81A3  E5                   push  hl            ;fuer Austausch retten
 81A4  C1                   pop   bc            ;austauschen (Laenge in BC)
 81A5  E1                   pop   hl            ;Endadresse

 81A6  3A BB 81     FUELL   ld    a,(FCHAR)     ;Fuellerzeichen
 81A9  77                   ld    (hl),a        ;in RAM-Diskadresse schreiben
 81AA  2B                   dec   hl            ;Adresse erniedrigen
 81AB  ØB                   dec   bc            ;Laenge erniedrigen
 81AC  78                   ld    a,b           ;pruefen ob
 81AD  FE ØØ                cp    Ø             ;das Ende erreicht
 81AF  C2 A6 81             jp    nz,FUELL      ;nein - weitermachen
 81B2  79                   ld    a,c           ;weitere Pruefung
 81B3  FE ØØ                cp    Ø             ;ob Ende erreicht
 81B5  C8                   ret   z             ;Ja -> Basic
 81B6  C3 A6 81             jp    FUELL         ;nein - weitermachen

 81B9  (ØØØ2)       MEMHIG  defs  2
 81BB  2E ØØ        FCHAR   defw  "."

Fehler: ØØØØØ  Warnungen: ØØØØØ

Symboltabelle:

8197 FILLER   81A6 FUELL   81BB FCHAR   81B9 MEMHIG  8195 RAMDI
```

Und nun zum Schluß noch der Bank-Umschalter, dessen Werte Sie direkt übernehmen und an den von Ihnen gewünschten Platz für einen einfachen Bankschalter bringen können.

Da keine absoluten Adressen (außer der Umschaltroutine selbst) enthalten sind, können die Werte in jedem von Ihnen frei wählbaren Bereich stehen.

Sie brauchen also lediglich die sieben Hexadezimalwerte DD, 7E, ØØ, CD, 5B, BD, C9 an den von Ihnen gewünschten Platz zu bringen und können dann von Basic aus die Bänke durch den Aufruf:

```
CALL Adresse,bank
```

umschalten wie Sie wollen. Adresse ist die Speicherstelle, an der Sie den ersten Wert abgelegt haben. Als Bank nur den Wert Ø oder die Werte 4 bis 7 benutzen!

7.4.10 Assembler-Listing: BSWITCH.EDI

```
                    ;****************************
                    ;*       BSWITCH.EDI        *
                    ;*                          *
                    ;(c) 1986 by Lothar Miedel  *
                    ;*--------------------------*
                    ;* Ein aeusserst einfacher  *
                    ;* Bankumschalter fuer den  *
                    ;*       CPC 6128           *
                    ;****************************
                    ;
                    ;Die Banknummern sind Ø (=normal) bzw. 4 bis 7
;
RAMSW   equ   &bd5b
                     org   &81bd

 81BD  DD 7E ØØ      BSWITCH ld    a,(ix+Ø)
 81CØ  CD 5B BD               call  RAMSW
 81C3  C9                     ret

Fehler: ØØØØØ  Warnungen: ØØØØØ

Symboltabelle:

81BD BSWITCH        BD5B RAMSW
```

7.5 Programmbeschreibung RANDOMD1
(für alle CPCs mit AMSDOS)

Dieses Programm ist das erste von zweien, die immer nur einen Teil der Daten in den CPC-Speicher laden. Prinzipiell könnte immer ein einziger Sektor eingelesen werden. Aber auch auf der Diskette soll der Speicherplatz optimal genutzt werden. Deshalb sind einige Überlegungen vorher sinnvoll.

Die Sektorlänge bei den normalen Sektorgrößen auf einer Diskette beträgt 512 Byte. Nun könnte pro Eintrag ein Sektor reserviert werden, dann wären immer soviele Einträge verfügbar, wie die Anzahl der nutzbaren Sektoren.

Nehmen wir eine ganz normale 3-Zoll-Diskette im Data-Only-Format. Auf dieser befinden sich neben den frei verfügbaren Sektoren auch welche für das Inhaltsverzeichnis (Directory). Wenn aber große Datenmengen verarbeitet werden sollen, dann ist es eigentlich nicht nötig, durch das Inhaltsverzeichnis vier Sektoren zu belegen, sondern es reicht einer! Dadurch gewinnen wir 3*512=1536 Byte. Aus diesem Grunde werden wir bei den Datendisketten für dieses Programm ein "Spezialformat" verwenden.

Die andere Überlegung ist aber, wenn wir schon so mit Byte knausern, dann wäre es totaler Unsinn, jedem Eintrag einen Sektor zu spendieren, wenn dieser ihn gar nicht benötigt. Andererseits aber soll auch die Möglichkeit geschaffen werden, mehr als 512 Byte pro Eintrag zuzulassen.

Deshalb wird bei diesem Programm die Diskette optimal genutzt, mit nur einer Ausnahme, daß wir noch einen Directoryblock zulassen.

Für einfache Überlegungen kann man sich die Diskette nun als einen großen Massenspeicher mit 40 Spuren (je 9 Sektoren) vorstellen, bei dem der erste Sektor fehlt und, das ist nun das Wichtige, bei dem immer nur ein Ausschnitt zu sehen ist. Dieser Ausschnitt ist immer ein Sektor.

Bevor wir nun weitermachen, wird erst einmal die Frage geklärt, wieviel Speicherplatz denn in diesem Falle zur Verfügung steht. Die Rechnung ist ganz einfach,

nämlich 39 (Spuren) * 9 (Sektoren) *512 Byte plus eine Spur * 8 Sektoren * 512 Byte.

Das ergibt eine für Daten zur Verfügung stehende Kapazität von 179712 plus 4096 = 183808 Byte und das ist auch die Zahl, die Sie beim Maskengenerator in Zeile 2450 als eines der Datastatements finden!

Die Anzahl der möglichen Einträge in eine derartige Datei ist deshalb die Integerzahl der Division

Zur Verfügung stehende Bytes/Zeichenzahl pro Eintrag.

Der Rest geht in diesem Falle verloren. Im ungünstigsten Fall hat der Rest den Wert:

Anzahl der Zeichen pro Eintrag minus 1.

Um das Programm zu verstehen, sind aber noch weitere Überlegungen wichtig. Nehmen wir an, daß ein Eintrag 500 Zeichen Länge haben soll, dann beginnt der erste Eintrag in Spur 0, Sektor 1, Byte 0 (oder 1 - das ist nur eine Betrachtungsweise), der zweite beginnt ebenfalls in Spur 0, Sektor 1, aber eben bei Byte 500 und erstreckt sich bis in den zweiten Sektor hinein. Bei anderen Eintragslängen, die nicht genau einer Sektorlänge entsprechen, gilt vergleichbares.

Bei Eintragslängen größer als eine Sektorlänge, z.B. 1008, kann sich deshalb ein Eintrag nicht nur über zwei, sondern insgesamt über drei Sektoren verteilen! Um in einem derartigen Falle aber alle Daten des Eintrages im CPC zu haben, müssen alle drei Sektoren eingelesen werden!

Das ist der Grund, weshalb Sie im Programm drei Lese- oder Schreibzugriffe pro Datenzugriff finden! Je nach Eintragslänge können diese Basic-Zeilen modifiziert werden.

Bis zu einer Eintragslänge von 1023 Byte kann das Programm aber unverändert belassen werden. Da ich davon ausgehe, daß dies für übliche Anwendungsfälle reicht, werden die wenigsten von Ihnen das Basic-Programm verändern müssen und können deshalb direkt mit dem abgedruckten Programm arbeiten.

Zwei wichtige Punkte sollen nicht vergessen werden.

1. Die Maschinenroutine für den Diskettenzugriff ist insgesamt zweimal vorhanden und dies auch in zwei verschiedenen Versionen. Dies soll helfen, wenn mit einem anderen Disketten-Betriebssystem als AMSDOS gearbeitet werden soll. Bei diesem Programm kommt die erste Version, bei RANDOMD2 die zweite Version zum Einsatz, um zu zeigen wie es geht!

2. Beim Einsatz dieses Programmes benötigt die Diskette ein ganz spezielles Format, diese Formatierroutine finden Sie im Abschnitt der Hilfs- und Zusatzprogramme!
   Es wäre möglich, auch noch mehr Spuren als die normalerweise zugelassenen zu benutzen, damit mir aber niemand - weil seine Diskettenstation defekt wurde - einen Vorwurf machen kann, unterbleibt selbst das Aufzeigen wie es geht. Der Hinweis, daß es geht, muß reichen!

Nun zur Beschreibung des Programmes.

Die Zeilen bis 450 sind mit denen der Programme RAMDISK1.BAS und RAMDISK2.BAS vergleichbar. Nur sind Werte anders bzw. es sind noch welche hinzugekommen. Von besonderem Interesse sind die Werte &84 und &85 in Zeile 450, dies sind die Befehle für Lesen und Schreiben von einzelnen Sektoren.

Ebenso sind die Zuordnungen für drive1 und drive2 für Spezialisten interessant, die damit Daten von einem Laufwerk auf ein anderes übernehmen können.

Auch ist es möglich die Daten nur auf dem Zweitlaufwerk zu verarbeiten, dann braucht man die Programmdiskette nicht gegen die Datendiskette auszutauschen. Da derartiges aber nicht der Regelfall sein dürfte, soll es bei diesem Hinweis bleiben.

Es ist wichtig zu bemerken, daß in diesem Programm - im Gegensatz zu den anderen - keine Offsetmöglichkeit mittels der Variablen "basis" vorgesehen ist. Wer dies haben will, kann bei den anderen Programmen "spicken".

Die Zeilen 490 bis 1140 entsprechen auch in etwa den in RAMDISK1 und RAMDISK2 verwendeten, nur die Zeilen-

nummern haben sich teilweise geändert, die Grundstruktur ist aber gleich.

Daß deshalb, weil bei dieser Art von Dateiverwaltung keine Binärfiles gelesen oder geschrieben werden müssen, die in anderen Programmen dafür vorhandenen Basic-Zeilen entfallen sind, dürfte jedem verständlich sein. Dafür sind aber andere hinzugekommen und diese sollen nun besprochen werden, denn Erklärungen zur Grundstruktur des Programmes finden Sie in den Beschreibungen der anderen und im Prinzip müßte ich hier nur alles mit anderen Zeilennummern wiederholen.

In Zeile 127Ø wird der Befehl für den Diskettenzugriff - in diesem Falle "schreiben" - definiert, da durch den Maschinensprache-Aufruf CALL befsuch,befehl die Adresse für den ROM-Einsprung und dessen ROM-Selekt (ROM-Nummer) gefunden werden soll. Durch das Maschinenprogramm werden diese beiden gefundenen Werte gleich in entsprechende Speicherstellen geschrieben, die dann durch die Routine die durch 128Ø angesprungen wird, als Adresse für den Einsprung und als ROM-Auswahl-Nummer übernommen werden.

Was geschieht aber nun in Zeile 128Ø? An die von dort angesprungene Routine werden die Parameter für den Bufferanfang (wohin der Sektor eingelesen werden soll), das Sektorformat (sf1) des ersten einzulesenden Sektors, die Spurnummer (track1) und die Laufwerksnummer übergeben und der Sektor wird eingelesen. Das Gleiche geschieht auch in den Zeilen 131Ø und 132Ø aber eben für den zweiten und dritten Sektor. Nur die Werte sind andere.

Neu sind auch die Zeilen 182Ø bis 198Ø. In diesen erfolgt die Berechnung der Spuren, der Sektorenformate, des Offsets usw. Sie dienen also für die Lese- und Schreibzugriffe als Unterroutinen. Die Zeile 198Ø ist mit 128Ø zu vergleichen, nur hat sich der Befehl geändert.

Durch die Zeilen 2ØØØ und 2Ø1Ø werden während des Zugriffs die erste Spur und der erste Sektor für den Zugriff auf dem Bildschirm ausgegeben.
Die Zeilen 2Ø5Ø bis 2Ø6Ø führen wieder den Befehl aus.

Warum ggf. drei Zugriffe erfolgen, wurde eingangs schon erklärt.

7.5.1 Programm RANDOMD1.BAS

```
100 '*************************************
110 '*                                   *
120 '*            RANDOMD1.BAS           *
130 '*___________________________________*
140 '*                                   *
150 '*   Copyright by Lothar Miedel 1986 *
160 '*                                   *
170 '*************************************
180 '
190 MODE 2:MEMORY &7FFF
200 '
210 'Fenster definieren
220 WINDOW #0,2,79,5,22:WINDOW #1,2,79,1,1
230 WINDOW #2,61,78,1,1:WINDOW #3,2,78,3,3
240 WINDOW #4,60,78,24,24: WINDOW #5,2,50,24,24
250 WINDOW #6,60,78,15,20
260 '
270 'Rahmen zeichnen
280 DRAW 0,380:DRAW 639,380:DRAW 639,349:DRAW 0,349:DRAW 0,35
290 DRAW 639,35:DRAW 639,379:DRAW 639,0:DRAW 0,0
300 '-------------------------------------------------------------
310 PRINT#1,CHR$(24) "Random-Dateiverwaltung mit DISK"CHR$(24)
320 PRINT#3,"Lade das Maschinenprogramm":LOAD"randomd1.mpg",&8000
330 GOSUB 530:'Maskendaten holen
340 '
350 'Definitionen
360 inv$=CHR$(24):norm$=inv$
370 zeilem=&818E:mspalte=zeilem+1:manzahl=zeilem+2:stream=zeilem+3
380 linksm=stream+1:rechtsm=stream+2:memhig=&81B9:fchar=&81BB
390 filler=&8197:zeichen=&2E:ramdi=zeilem+7:ctaste=&8194
400 ramrd=&80BB:wed=&8110:ramwr=&80F1:fundpos=&8077:suchwort =&807F
410 suche =&8000:ramlen=&8071:laenge=&8073:aktstart=&8075
420 beginn=VAL(buffanf$)+2^16
430 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
440 zeile=1:spalte=1:str=7:befsuch=&81BD:exe=&81D4
450 lesen=&84:schreiben=&85:seclen=512:secoff=1:drive1=0:drive2=0
460 '
470 '*****************************************************************
480 '
490 POKE memhig,&FF:POKE memhig+1,&9F:POKE ramdi,&0:POKE ramdi+1,&90
500 POKE fchar,zeichen:CALL filler:GOTO 710
510 '-----------------------------------------------------
520 'Maskendaten lesen
530 PRINT#3,"Lese die Maskendaten
540 OPENIN"randomd1.msk"
550  INPUT #9,buffanf$,buffend$,eintraege,off,felder
```

```
560 buffanf=VAL(buffanf$):buffend=VAL(buffend$)
570 IF buffend<0 THEN buffend=buffend+2^16
580 diff=buffend-buffanf+1
590 DIM fenam$(felder),feldlen(felder),xposi(felder),yposi(felder)
600 DIM feldinh$(felder)
610   FOR i = 1 TO felder
620     INPUT #9,fenam$(i)
630     INPUT #9,feldlen(i)
640     benutzt=benutzt+feldlen(i)
650     INPUT #9,xposi(i)
660     INPUT #9,yposi(i)
670   NEXT i
680 CLOSEIN:PRINT#3,inv$ "Maske gelesen - Ausgabe der Maske" norm$
690 RETURN
700 '-------------------------------------------------------------
710  FOR j = 1 TO felder
720      LOCATE xposi(j),yposi(j)
730      PRINT fenam$(j)
740 NEXT j
750 '-------------------------------------------------------------
760 PRINT #3,"Auf geht's !
770 LOCATE #1,47,1:PRINT#1,"Dateieintraege: "eintraege"*"off
780 PRINT#5,"Datendiskette einlegen und g fuer goto eingeben!
790 GOTO 910
800 '
810 PRINT #3,inv$"GOTO ...."inv$
820 INPUT #5,"Eintragsnummer";nr$:enr$=LEFT$(nr$,6)
830 IF LEN(nr$)<1 THEN 820
840 IF ASC(LEFT$(nr$,1))>57 THEN 820
850 IF ASC(LEFT$(nr$,1))<48 THEN 820
860 nr=VAL(nr$):GOTO 1030
870 '
880 CLS #3:PRINT #3,inv$ "Daten gueltig! Bitte Ihre Eingabe!" norm$
890 CLS #5:PRINT #5,"Eintragsnummer: "enr"  "HEX$(begad)
900 '
910 x$=LOWER$(INKEY$):IF x$="" THEN 910
920 IF x$="e" THEN 1760
930 IF x$="f" THEN 1430
940 IF x$="g" THEN 810
950 IF ASC(x$)=224 THEN 1180
960 IF ASC(x$)<240 THEN 910
970 IF ASC(x$)>243 THEN 910
980 IF ASC(x$)=243 THEN enr=enr+1
990 IF ASC(x$)=242 THEN enr=enr-1
1000 IF ASC(x$)=240 THEN enr=enr+10
1010 IF ASC(x$)=241 THEN enr=enr-10
1020 nr=enr
1030 IF nr<1 THEN nr=1
```

```
1040 '
1050 IF nr>eintraege THEN nr=eintraege
1060 enr=nr
1070 '
1080 GOSUB 1810
1090 'Leseschleife
1100 POKE stream,str:zz=0:FOR i = 1 TO felder
1110 beginn=begad+zz:zz=zz+feldlen(i)
1120 zeile=yposi(i):anzahl=feldlen(i)-1
1130 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
1140 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+16,zeile+4,zeile+4:CLS #7
1150 GOSUB 1680:CALL ramwr:NEXT i:GOTO 880
1160 '-----------------------------------------
1170 'Aendern
1180 PRINT #3,inv$"Bitte Ihre Eingaben!"inv$:PRINT #5,"Aendern":PRINT #4,nr
1190 POKE stream,str:zz=0
1200 FOR i = 1 TO felder
1210 beginn=begad+zz:zz=zz+feldlen(i):zeile=yposi(i):anzahl=feldlen(i)-1
1220 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
1230 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+17,zeile+4,zeile+4
1240 GOSUB 1740:CALL WED
1250 '
1260 anzahl=feldlen(i):GOSUB 1660:CALL ramrd:NEXT i:PRINT #5
1270 befehl=schreiben:CALL befsuch,befehl
1280 CALL exe,buffanf,sf1,track1,drive2
1290 IF track2>39 THEN 1330
1300 IF track3>39 THEN 1330
1310 CALL exe,buffanf+&200,sf2,track2,drive2
1320 CALL exe,buffanf+&400,sf3,track3,drive2
1330 CLS #4:GOTO 880
1340 '
1350 'Fehlerbehandlung
1360 PRINT #3,inv$ "Fehler! "norm$;
1370 PRINT #5,inv$" Bitte Taste 'j' oder 'n' druecken!"inv$:PRINT CHR$(7);
1380 x$=LOWER$(INKEY$):IF x$="" THEN 1380
1390 IF x$<> "j" AND x$<> "n" THEN 1380 ELSE CLS #5:RETURN
1400 CLS #3
1410 RETURN
1420 '-------------------------------------------------------------
1430 PRINT #3,inv$"Suchen: "inv$"  Bitte geben Sie die Zeichenfolge ein!  "
1440 INPUT #5,such$:IF LEN (such$)=0 THEN sw$=sw$ ELSE sw$=such$
1450 wlaenge=LEN(sw$):IF wlaenge>56 THEN sw$=LEFT$(sw$,56):GOTO 1450
1460 FOR i=1 TO wlaenge:POKE suchwort+i-1,ASC(MID$(sw$,i,1)):NEXT i
1470 POKE laenge,wlaenge
1480 stand=begad
1490 '
1500 sthi=INT(stand/256):stlo=stand-sthi*256
1510 rest=off+1
```

```
1520 ramlhi=INT(rest/256):ramllo=rest-ramlhi*256
1530 '
1540 POKE aktstart,stlo:POKE aktstart+1,sthi
1550 POKE ramlen,ramllo:POKE ramlen+1,ramlhi
1560 '
1570 CALL suche:fs=PEEK(fundpos)+PEEK(fundpos+1)*256
1580 IF fs<>0 THEN IF fs>=begad+off THEN enr=enr+1:GOTO 1020
1590 IF fs<>0 THEN 1020
1600 '
1610 PRINT #5,"Eintragsnummer: "enr
1620 IF enr =eintraege THEN IF fs>&97FF+2^16 THEN 1650
1630 enr=enr+1:IF enr > eintraege THEN 1650 ELSE GOSUB 1810
1640 GOTO 1480
1650 enr=eintraege:PRINT #3,inv$"Nicht gefunden !"inv$:GOSUB 1370:GOTO 1070
1660 '------------------------------------------------------------
1670 'Lesen aus Window 7 und schreiben in RAM-Disk
1680 zeile=1:spalte=1:POKE zeilem,zeile
1690 POKE manzahl,anzahl:POKE mspalte,spalte
1700 POKE ramdi,ramlow:POKE ramdi+1,ramhigh
1710 IF PEEK(ctaste)=224 THEN i = felder
1720 POKE ctaste,0:RETURN
1730 '
1740 links=16:rechts=feldlen(i)
1750 POKE stream,str:POKE linksm,1:POKE rechtsm,rechts:RETURN
1760 PRINT #3,"Programm beenden ???"
1770 GOSUB 1370
1780 IF x$="n" THEN GOTO 1070
1790 IF x$="j" THEN MODE 2:END
1800 '
1810 CLS #3:PRINT #3,inv$"Moment bitte!"inv$
1820 bytstand1=enr*off-off:bytstand2=bytstand1+seclen
1830 bytstand3=bytstand2+seclen
1840 hilfsnr1=INT(bytstand1/seclen):secnr1=hilfsnr1+1
1850 secnr2=secnr1+1
1860 secnr3=secnr1+2
1870 sf1=&C1+secnr1 MOD 9:sf2=&C1+secnr2 MOD 9:sf3=&C1+secnr3 MOD 9
1880 track1=INT((bytstand1+secoff*seclen)/(seclen*9))
1890 track2=INT((bytstand2+secoff*seclen)/(seclen*9))
1900 track3=INT((bytstand3+secoff*seclen)/(seclen*9))
1910 '
1920 secbytes=(secnr1-1)*seclen
1930 bytes=enr*off-off
1940 off1=bytes-secbytes
1950 IF enr >1 THEN off1=off1
1960 IF enr =1 THEN off1=0
1970 '
1980 befehl=lesen:CALL befsuch,befehl
1990 '
```

```
2000 PRINT #5,"Track1: "track1"  ";
2010 PRINT #5,"Sektor: "HEX$(sf1)"  ";
2020 CALL exe,buffanf,sf1,track1,drive1
2030 IF track2>39 THEN RETURN
2040 IF track3>39 THEN RETURN
2050 CALL exe,buffanf+&200,sf2,track2,drive1
2060 CALL exe,buffanf+&400,sf3,track3,drive1
2070 buffoff=off1:begad=buffanf+buffoff+2^16
2080 RETURN
```

Nun folgen die Ladeprogramme für die verschiedenen Maschinenroutinen.

Für das Dateiverwaltungsprogramm RANDOMD1 wird neben den Maskendaten nur der Binärteil, der durch RDKPL.HEX erzeugt wird, benötigt. Die anderen Lader sind wieder nur für Ihre eigenen Dateiverwaltungsprogramme gedacht, falls Sie mit diesen Routinen arbeiten wollen!

7.5.2 Das Ladeprogramm DISC1.HEX

```
100 REM DISC1.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 33213:e= 33305:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"disc1.bin",b,&81BD,&5A
240 '
2000 DATA DD,6E,00,DD,66,01,22,F2,81,21,F2,81,CD,D4,BC,22,&0837
2001 DATA EE,81,79,32,F0,81,C9,DD,7E,00,5F,DD,7E,02,57,DD,&089F
2002 DATA 7E,04,4F,DD,6E,06,DD,66,07,DF,EE,81,D8,32,F1,81,&0836
2003 DATA C9,00,00,00,00,00,00,00,00,C5,02,00,00,C1,02,00,&0253
2004 DATA 00,C6,02,00,00,C2,02,00,00,C7,02,00,00,C3,02,00,&031A
2005 DATA 00,C8,02,00,00,C4,02,00,00,C9,02,00,&025B
```

Der vorstehende Lader erzeugt das Diskettenzugriffsprogramm der ersten Version und der nun folgende die andere.

7.5.3 Das Ladeprogramm DISC2.HEX

```
100 REM DISC2.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 33304:e= 33343:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"disc2.bin",b,&8218,&2F
240 '
2000 DATA 3A,42,82,32,42,82,21,3E,82,CD,D4,BC,22,3F,82,79,&068E
2001 DATA 32,41,82,21,42,82,5E,21,43,82,56,21,44,82,4E,2A,&04D3
2002 DATA 46,82,DF,3F,82,C9,84,&03B5
```

7.5.4 Das Ladeprogramm DISCKPL.HEX

```
100 REM DISCKPL.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 33213:e= 33343:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"disckpl.bin",b,&81BD,&82
240 '
2000 DATA DD,6E,00,DD,66,01,22,F2,81,21,F2,81,CD,D4,BC,22,&0837
2001 DATA EE,81,79,32,F0,81,C9,DD,7E,00,5F,DD,7E,02,57,DD,&089F
2002 DATA 7E,04,4F,DD,6E,06,DD,66,07,DF,EE,81,D8,32,F1,81,&0836
2003 DATA C9,00,00,00,00,00,00,00,00,C5,02,00,00,C1,02,00,&0253
2004 DATA 00,C6,02,00,00,C2,02,00,00,C7,02,00,00,C3,02,00,&031A
2005 DATA 00,C8,02,00,00,C4,02,00,00,C9,02,3A,42,82,32,42,&03CD
2006 DATA 82,21,3E,82,CD,D4,BC,22,3F,82,79,32,41,82,21,42,&0674
2007 DATA 82,5E,21,43,82,56,21,44,82,4E,2A,46,82,DF,3F,82,&05E3
2008 DATA C9,84,&014D
```

7.5.5 Das Ladeprogramm RDKPL.HEX

```
100 REM RDKPL.HEX
110 '
120 MEMORY &7FFF
130 '
140 a= 32768:e= 33342:zb=2000
150 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 180
160 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
170 ps=0:d$="":IF i=e THEN 230 ELSE i=i-1:zb=zb+1:GOTO 210
180 d$="&"+d$
190 POKE i,VAL(d$):ps=ps+VAL(d$)
200 PRINT VAL(d$)
210 IF i < e THEN NEXT i
220 '
230 SAVE"randomd1.mpg",b,&8000,&23D
240 '
2000 DATA ED,4B,71,80,11,7F,80,2A,75,80,22,79,80,3A,73,80,&06A0
2001 DATA 32,74,80,1A,ED,B1,C2,6A,80,2B,22,75,80,22,7D,80,&06EB
2002 DATA 22,79,80,3A,73,80,32,74,80,1A,ED,B1,C2,6A,80,D5,&07A7
2003 DATA E5,2A,79,80,23,E5,D1,E1,E5,ED,52,E1,D1,22,79,80,&09B3
2004 DATA CA,4D,80,2A,75,80,23,22,75,80,C3,00,80,22,7B,80,&0650
2005 DATA 3A,74,80,3D,32,74,80,FE,00,CA,60,80,13,C3,29,80,&06B8
2006 DATA ED,5B,73,80,ED,52,22,77,80,C9,21,00,00,22,77,80,&0696
2007 DATA C9,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&00C9
2008 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
2009 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
2010 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,&0000
2011 DATA 00,00,00,00,00,00,00,00,00,00,00,3A,91,81,CD,B4,&02CD
2012 DATA BB,3A,8E,81,F5,CD,72,BB,F1,32,8E,81,3A,8F,81,F5,&0964
2013 DATA CD,6F,BB,F1,3C,32,8F,81,2A,90,81,7D,FE,00,C8,2B,&080F
2014 DATA 22,90,81,CD,60,BB,2A,95,81,77,23,22,95,81,C3,CC,&07BC
2015 DATA 80,3A,91,81,CD,B4,BB,2A,95,81,7E,23,22,95,81,CD,&07EE
2016 DATA 5D,BB,2A,90,81,7D,FE,00,C8,2B,22,90,81,C3,F7,80,&082E
2017 DATA 3A,91,81,CD,B4,BB,CD,78,BB,CD,75,BB,CD,8A,BB,CD,&0A64
2018 DATA 06,BB,4F,CD,8D,BB,79,FE,0D,C8,FE,E0,CA,8A,81,FE,&0A22
2019 DATA F2,CA,56,81,FE,F3,CA,6C,81,FE,7F,F2,16,81,FE,20,&0A5F
2020 DATA FA,16,81,CD,5A,BB,CD,78,BB,3A,93,81,3C,57,7C,BA,&088A
2021 DATA CA,82,81,C3,16,81,3E,08,4F,CD,78,BB,3A,92,81,57,&0760
2022 DATA 7C,BA,CA,16,81,79,CD,5A,BB,C3,16,81,3E,09,4F,CD,&07AF
2023 DATA 78,BB,3A,93,81,57,7C,BA,CA,16,81,79,CD,5A,BB,C3,&088D
2024 DATA 16,81,3E,08,CD,5A,BB,C3,16,81,32,94,81,C9,01,01,&062B
2025 DATA 01,07,05,06,00,00,00,2A,95,81,2B,E5,C1,2A,B9,81,&0488
2026 DATA E5,ED,42,E5,C1,E1,3A,BB,81,77,2B,0B,78,FE,00,C2,&08F6
2027 DATA A6,81,79,FE,00,C8,C3,A6,81,00,00,2E,00,DD,6E,00,&06C9
2028 DATA DD,66,01,22,F2,81,21,F2,81,CD,D4,BC,22,EE,81,79,&08D4
2029 DATA 32,F0,81,C9,DD,7E,00,5F,DD,7E,02,57,DD,7E,04,4F,&0788
2030 DATA DD,6E,06,DD,66,07,DF,EE,81,D8,32,F1,81,C9,00,00,&082E
```

```
2031 DATA 00,00,00,00,00,00,C5,02,00,00,C1,02,00,00,C6,02,&0252
2032 DATA 00,00,C2,02,00,00,C7,02,00,00,C3,02,00,00,C8,02,&031C
2033 DATA 00,00,C4,02,00,00,C9,02,3A,42,82,32,42,82,21,3E,&03E4
2034 DATA 82,CD,D4,BC,22,3F,82,79,32,41,82,21,42,82,5E,21,&0694
2035 DATA 43,82,56,21,44,82,4E,2A,46,82,DF,3F,82,C9,&05AB
```

Nun folgen die Assemblerlistings. Da die Programmteile für den Filler-, RD- und Sucherteil mit denen des Programms RAMDISK2 identisch sind und lediglich der Teil des Bankschalters entfällt, finden Sie hier nur die Assemblerroutinen für den Diskettenzugriff.

7.5.6 Assembler-Listing: DISC1.EDI

```
                        ;****************************
                        ;*       disc1.edi          *
                        ;--------------------------*
                        ;* gedacht fuer Disketten- *
                        ;* Zugriffe unter AMSDOS   *
                        ;****************************

bsuch   equ   &bcd4

                         org   &81bd

                        ;Routine zur Befehlssuche

81BD  DD 6E 00      Einsp1  LD    L,(IX+00)     ;Befehlsadresse
81C0  DD 66 01              LD    H,(IX+01)     ;an Speicherstelle
81C3  22 F2 81              LD    (CMD),hl      ;Kommando uebergeben
81C6  21 F2 81      Einsp2  LD    HL,CMD        ;HL nun als Befehlszeiger
81C9  CD D4 BC              CALL  bsuch         ;Befehl suchen
81CC  22 EE 81              LD    (FAR),hl      ;Adresse merken
81CF  79                    LD    A,C           ;ROM-Auswahl-Nummer
81D0  32 F0 81              LD    (ROM),A       ;in ROM merken
81D3  C9                    RET


                        ;Routine zur Befehlsausfuehrung

81D4  DD 7E 00              LD    A,(IX+00)
81D7  5F                    LD    E,A
81D8  DD 7E 02              LD    A,(IX+02)
81DB  57                    LD    D,A
81DC  DD 7E 04              LD    A,(IX+04)
```

```
81DF  4F                  LD    C,A
81E0  DD 6E 06            LD    L,(IX+06)
81E3  DD 66 07            LD    H,(IX+07)
81E6  DF EE 81            RST   &18,FAR       ;Befehl ausfuehren
81E9  D8                  RET   C             ;korrekt ausgefuehrt => Basic
81EA  32 F1 81            LD    (RUE),a       ;Fehler merken
81ED  C9                  RET

81EE  (0002)        FAR   defs  2             ;Adresse der DOS-Routine
81F0  (0001)        ROM   defs  1             ;ROM-Auswahl-Nummer
81F1  (0001)        RUE   defs  1             ;Rueckmeldung
81F2  (0002)        CMD   defs  2             ;der gesuchte Befehl

                  ;Formattabelle
                  ;Sie dient zur Erzeugung des speziellen
                  ;Formates bei der Benutzung von
                  ;RANDOMD1

81F4  00 00 C5 02         defb  &00,&00,&c5,&02
81F8  00 00 C1 02         defb  &00,&00,&c1,&02
81FC  00 00 C6 02         defb  &00,&00,&c6,&02
8200  00 00 C2 02         defb  &00,&00,&c2,&02
8204  00 00 C7 02         defb  &00,&00,&c7,&02
8208  00 00 C3 02         defb  &00,&00,&c3,&02
820C  00 00 C8 02         defb  &00,&00,&c8,&02
8210  00 00 C4 02         defb  &00,&00,&c4,&02
8214  00 00 C9 02         defb  &00,&00,&c9,&02

Fehler: 00000  Warnungen: 00000

Symboltabelle:

BCD4 BSUCH  81F2 CMD  81BD EINSP1  81C6 EINSP2
81EE FAR    81F0 ROM  81F1 RUE
```

7.5.7 Assembler-Listing: DISC2.EDI

```
                  ;*******************************
                  ;*          DISC2.EDI          *
                  ;*   Hilfsprogramm fuer alle   *
                  ;*     Diskettenstationen      *
                  ;*    (auch Vortex-X-Modul)    *
                  ;*******************************

akdrive equ   &a702:findcom equ   &bcd4
```

```
                org   &8218

8218  3A 42 82            ld    a,(drive)     ;gewuenschtes Laufwerk
821B  32 42 82            ld    (drive),a     ;merken
821E  21 3E 82            ld    hl,befehl     ;Befehl aus Tabelle
8221  CD D4 BC            call  findcom       ;Befehl im ROM suchen
8224  22 3F 82            ld    (romadr),hl   ;Romadresse merken
8227  79                 ld    a,c           ;Romselect
8228  32 41 82            ld    (romsel),a    ;merken

822B  21 42 82            ld    hl,drive      ;Adr fuer Drive # holen
822E  5E                  ld    e,(hl)        ;nach E uebergeben
822F  21 43 82            ld    hl,track      ;Adr fuer Track # holen
8232  56                  ld    d,(hl)        ;nach d uebergeben
8233  21 44 82            ld    hl,formse     ;Format + Sector
8236  4E                  ld    c,(hl)        ;in Reg C
8237  2A 46 82            ld    hl,(buffer)   ;Bufferadr in HL
823A  DF 3F 82            rst   3,romadr      ;weitermachen bei ROMADR
823D  C9                  ret                 ;und zurueck

823E  84          befehl  defb  &84
823F  (0002)      romadr  defs  2
8241  (0001)      romsel  defs  1
8242  (0001)      drive   defs  1
8243  (0001)      track   defs  1
8244  (0001)      formse  defs  1
8245  00          kopf    defb  &00           ;muss auf 00 bleiben!
8246  (0002)      buffer  defs  2

Fehler: 00000  Warnungen: 00000

Symboltabelle:

A702 AKDRIVE   823E BEFEHL   8246 BUFFER   8242 DRIVE   BCD4 FINDCOM
8244 FORMSE    8245 KOPF     823F ROMADR   8241 ROMSEL  8243 TRACK
```

7.6 Programmbeschreibung RANDOMD2
(für alle CPCs mit VORTEX-X-MODUL)

Für dieses zweite Programm mit direktem Zugriff gilt äquivalent das bei der Beschreibung RANDOMD1 stehende. Aber ein paar kleine Unterschiede sind vorhanden, auf die hier näher eingegangen werden soll.

Zunächst einmal ist festzustellen, daß anstelle der einseitigen Schneider-Laufwerke die 40 Spuren haben, nun eines mit 80 Spuren vorhanden ist, das auch noch zweiseitig arbeitet.

Um hier nun wieder eine etwas andere Programm-Technik zu wählen, ist auch der Sektor für das Inhaltsverzeichnis den Daten zum Opfer gefallen. Das heißt, ein Inhaltsverzeichnis wird überhaupt nicht benutzt. Das birgt allerdings auch Gefahren in sich: So könnte leicht eine Datendiskette mit Programmen überschrieben werden, deshalb also ganz besondere Vorsicht walten lassen und die Datendiskette ganz auffällig kennzeichnen, damit dies nicht geschieht.

Da die X-Module-Laufwerke sowohl im 3,5-Zoll- als auch 5,25-Zoll-Format erhältlich sind, sei an dieser Stelle erwähnt, daß das Programm mit einem 5,25-Zoll-Laufwerk an einem CPC 6128 getestet wurde. Ich sehe aber keinen Grund, weshalb es Schwierigkeiten mit anderen CPCs oder mit dem 3,5-Zoll-Laufwerk geben sollte!

Doch nun erst zur Frage der Kapazität. 160 Spuren mit je 9 Sektoren und pro Sektor 512 Byte, das ergibt eine Gesamtkapazität von 737280 Byte.

Eine Kapazität dieser Größenordnung für Daten, sollte eigentlich für Normalanwendungen ausreichen. Wem es nicht reicht, der sollte sich am besten nach einem anderen Computersystem umsehen oder den Kauf einer Harddisc erwägen, denn Disketten-Laufwerke mit noch größerer Kapazität wird man auch die nächste Zeit kaum zu vernünftigen Preisen erhalten können. Ein weiterer Ausweg wäre aber auch noch die Verteilung der Daten auf mehrere Disketten. Auch aus diesem Grunde ist im Programm wieder die "basis"-Festlegung möglich.

Noch einmal kurz zum Kapazitätswert zurück, dieser ist in Zeile 2460 des Maskengenerators zu finden.

Größtenteils gilt für dieses Programm das schon bei RANDOMD1 stehende. Aber einige Punkte sind doch so unterschiedlich, daß diese nun aufgeführt werden müssen.

Zunächst einmal wird die zweite Maschinenroutine für Direktzugriffe auf die Diskettenstation verwendet. Dadurch ergeben sich auch andere Übergabeweisen an das Maschinenprogramm.

Andererseits aber werden alle Sektoren als Datensektoren verwendet und deshalb erfolgt auch deren Berechnung etwas anders. In Zeile 35Ø wird eine Hilfszahl berechnet. Sie dient zur Bestimmung wieviele Sektoren - im ungünstigsten Falle - eingelesen werden müssen. Diese Anzahl wird dann auch immer eingelesen.

Bei den Definitionen haben sich, durch die Verwendung des anderen Maschinenprogrammes für das Diskettenhandling, einige Werte - gegenüber RANDAMD1.BAS - geändert.

Die Art des Maschinenprogramm-Ansprunges ist anders: Dies zeigt beispielsweise die Zeile 88Ø. Der Befehl des Diskettenzugriffes wird per POKE abgelegt. Auch die Sektor- und Track-Berechnung erfolgt in den Zeilen 19ØØ bis 2Ø4Ø anders.

Und zu "guter Letzt" zeigen die Zeilen 2Ø8Ø bis 215Ø auch noch ein anderes Aussehen.

Bei diesem Programm wird vorausgesetzt, daß ein CPC mit dem Erstlaufwerk "Schneider" und mit dem Zweitlaufwerk "X-Modul-Vortex" ausgestattet ist. Deshalb wird in die zuständige Speicherstelle mdrive der Wert eins geladen.

Aber ansonsten ist dieses Programm - bis auf kleine Unterschiede - wie RANDOMD1.BAS! Nur ein paar Punkte sind auch hier noch erwähnenswert, die Diskette wird "spezialformatiert" und dieser "Formatter" ist im Abschnitt der Hilfs- und Zusatzprogramme zu finden.

Die Assemblerlistings für dieses Programm finden Sie unter der Programmbeschreibung für RANDOMD1, da sie identisch sind.

Als Hexlader sind ebenfalls die des anderen Direktzugriffsprogrammes zu verwenden.

7.6.1 Programm RANDOMD2.BAS

```
100 '****************************************
110 '*                                      *
120 '*            RANDOMD2.BAS              *
130 '*______________________________________*
140 '*                                      *
150 '*   Copyright by Lothar Miedel 1986    *
160 '*                                      *
170 '****************************************
180 '
190 MODE 2:MEMORY &7FFF:basis=1:seclen=512
200 '
210 'Fenster definieren
220 WINDOW #0,2,79,5,22:WINDOW #1,2,79,1,1
230 WINDOW #2,61,78,1,1:WINDOW #3,2,78,3,3
240 WINDOW #4,60,78,24,24: WINDOW #5,2,50,24,24
250 WINDOW #6,60,78,15,20
260 '
270 'Rahmen zeichnen
280 DRAW 0,380:DRAW 639,380:DRAW 639,349:DRAW 0,349:DRAW 0,35
290 DRAW 639,35:DRAW 639,379:DRAW 639,0:DRAW 0,0
300 '-------------------------------------------------------------
310 PRINT#1,CHR$(24) "Random-Dateiverwaltung mit X-Modul"CHR$(24)
320 PRINT#3,"Lade das Maschinenprogramm":LOAD"randomd1.mpg",&8000
330 '
340 GOSUB 510:'Maskendaten holen
350 hilfszahl=off+seclen-1'           Hilfszahl zur Berechnung
360 anzsec=INT(hilfszahl/seclen)+1'   Anzahl der einzulesenden Sektoren
370 '-------------------------------------------------------------
380 'Definitionen
390 zeilem=&818E:mspalte=zeilem+1:manzahl=zeilem+2:stream=zeilem+3
400 linksm=stream+1:rechtsm=stream+2:memhig=&81B9:fchar=&81BB
410 filler=&8197:zeichen=&2A:ramdi=zeilem+7:ctaste=&8194:inv$=CHR$(24)
420 ramrd=&80BB:wed=&8110:ramwr=&80F1:fundpos=&8077:suchwort =&807F
430 suche =&8000:ramlen=&8071:laenge=&8073:aktstart=&8075:mbuffer=&8246
440 beginn=VAL(buffanf$):zeile=1:spalte=1:str=7:mformse=&8244
450 mbefehl=&823E:mdrive=&8242:mtrack=&8243:lesen=&84:schreiben=&85
460 '-------------------------------------------------------------
470 POKE memhig,&FF:POKE memhig+1,&9F:POKE ramdi,&0:POKE ramdi+1,&90
480 POKE fchar,zeichen:CALL filler:GOTO 690
490 '-------------------------------------------------------------
500 'Maskendaten lesen
510 PRINT#3,"Lese die Maskendaten
520 OPENIN"randomd2.msk"
530  INPUT #9,buffanf$,buffend$,eintraege,off,felder
540 buffanf=VAL(buffanf$)+2^16:buffend=VAL(buffend$)+2^16
550 IF buffend<0 THEN buffend=buffend+2^16
```

```
560 diff=buffend-buffanf+1
570 DIM fenam$(felder),feldlen(felder),xposi(felder),yposi(felder)
580 DIM feldinh$(felder)
590   FOR i = 1 TO felder
600     INPUT #9,fenam$(i)
610     INPUT #9,feldlen(i)
620     benutzt=benutzt+feldlen(i)
630     INPUT #9,xposi(i)
640     INPUT #9,yposi(i)
650   NEXT i
660 CLOSEIN:PRINT#3,inv$ "Maske gelesen - Ausgabe der Maske" inv$
670 RETURN
680 '-------------------------------------------------------------
690  FOR j = 1 TO felder
700      LOCATE xposi(j),yposi(j)
710      PRINT fenam$(j)
720 NEXT j
730 '-------------------------------------------------------------
740 PRINT #3,"Auf geht's !
750 LOCATE #1,47,1:PRINT#1,"Dateieintraege: "eintraege"*"off
760 PRINT#5,"Datendiskette einlegen und g fuer goto eingeben!
770 x$=LOWER$(INKEY$):IF x$<>"g" THEN 770
780 GOTO 920
790 '-------------------------------------------------------------
800 PRINT #3,inv$"GOTO ...."inv$
810 INPUT #5,"Eintragsnummer";nr$:enr$=LEFT$(nr$,6)
820 IF LEN(nr$)<1 THEN 810
830 IF ASC(LEFT$(nr$,1))>57 THEN 810
840 IF ASC(LEFT$(nr$,1))<48 THEN 810
850 enr=VAL(enr$):GOTO 1020
860 '
870 CLS #3:PRINT #3,inv$ "Daten gueltig! Bitte Ihre Eingabe!" inv$
880 befehl=lesen:POKE mbefehl,befehl
890 CLS #5:PRINT #5,"Eintragsnummer: "enr
900 '
910 x$=LOWER$(INKEY$):IF x$="" THEN 910
920 IF x$="e" THEN 1830
930 IF x$="f" THEN 1490
940 IF x$="g" THEN 800
950 IF ASC(x$)=224 THEN 1190
960 IF ASC(x$)<240 THEN 910
970 IF ASC(x$)>243 THEN 910
980 IF ASC(x$)=243 THEN enr=enr+1
990 IF ASC(x$)=242 THEN enr=enr-1
1000 IF ASC(x$)=240 THEN enr=enr+10
1010 IF ASC(x$)=241 THEN enr=enr-10
1020 nr=enr-basis
1030 IF enr<basis THEN enr=basis
```

```
1040 IF enr>=eintraege+basis THEN enr=eintraege+basis-1
1050 IF nr<0 THEN nr=0
1060 nr=enr-basis
1070 sflag=0:befehl=lesen:POKE mbefehl,befehl
1080 GOSUB 1870
1090 '
1100 'Leseschleife
1110 POKE stream,str:zz=0:FOR i = 1 TO felder
1120 beginn=begad+zz:zz=zz+feldlen(i)
1130 zeile=yposi(i):anzahl=feldlen(i)-1
1140 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
1150 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+16,zeile+4,zeile+4:CLS #7
1160 GOSUB 1740:CALL ramwr:NEXT i:GOTO 870
1170 '------------------------------------------------------------
1180 'Aendern
1190 PRINT #3,inv$"Bitte Ihre Eingaben!"inv$:PRINT #5,"Aendern":PRINT #4,enr
1200 POKE stream,str:zz=0:befehl=schreiben:POKE mbefehl,befehl
1210 FOR i = 1 TO felder
1220 beginn=begad+zz:zz=zz+feldlen(i):zeile=yposi(i):anzahl=feldlen(i)-1
1230 ramhigh=INT(beginn/256):ramlow=beginn-ramhigh*256
1240 WINDOW #7,xposi(i)+16,xposi(i)+feldlen(i)+17,zeile+4,zeile+4
1250 GOSUB 1800:CALL WED
1260 '
1270 anzahl=feldlen(i):GOSUB 1740:CALL ramrd:NEXT i:PRINT #5
1280 '
1290 POKE mtrack,track1:POKE mformse,sf1
1300 POKE mdrive,1:POKE mbuffer,0:POKE mbuffer+1,&90
1310 CALL &8218:IF track2>159 THEN 1410
1320 '
1330 POKE mtrack,track2:POKE mformse,sf2
1340 POKE mdrive,1:POKE mbuffer,0:POKE mbuffer+1,&92
1350 CALL &8218:IF track3>159 THEN 1410
1360 '
1370 POKE mtrack,track3:POKE mformse,sf3
1380 POKE mdrive,1:POKE mbuffer,0:POKE mbuffer+1,&94
1390 CALL &8218
1400 '
1410 CLS #4:GOTO 870
1420 '------------------------------------------------------
1430 'Fehlerbehandlung
1440 PRINT #3,inv$ "Fehler! "inv$;
1450 PRINT #5,inv$" Bitte Taste 'j' oder 'n' druecken!"inv$:PRINT CHR$(7);
1460 x$=LOWER$(INKEY$):IF x$="" THEN 1460
1470 IF x$<> "j" AND x$<> "n" THEN 1460 ELSE CLS #5:CLS #3:RETURN
1480 '-------------------------------------------------------------
1490 PRINT #3,inv$"Suchen: "inv$"  Bitte geben Sie die Zeichenfolge ein!  "
1500 INPUT #5,such$:IF LEN (such$)=0 THEN sw$=sw$ ELSE sw$=such$
1510 wlaenge=LEN(sw$):IF wlaenge>56 THEN sw$=LEFT$(sw$,56):GOTO 1510
```

```
1520 IF wlaenge<1 THEN 1490
1530 FOR i=1 TO wlaenge:POKE suchwort+i-1,ASC(MID$(sw$,i,1)):NEXT i
1540 POKE laenge,wlaenge:sflag=1
1550 altenr=enr:GOTO 1690
1560 stand=begad
1570 '
1580 sthi=INT(stand/256):stlo=stand-sthi*256:rest=off
1590 ramlhi=INT(rest/256):ramllo=rest-ramlhi*256
1600 '
1610 POKE aktstart,stlo:POKE aktstart+1,sthi
1620 POKE ramlen,ramllo:POKE ramlen+1,ramlhi
1630 '
1640 CALL suche:fs=PEEK(fundpos)+PEEK(fundpos+1)*256
1650 IF fs<>0 THEN 1020
1660 '
1670 PRINT #5,"Eintragsnummer: "enr
1680 IF enr>=eintraege THEN 1710
1690 nr=nr+1:enr=enr+1:IF nr > eintraege THEN 1710 ELSE GOSUB 1870
1700 GOTO 1560
1710 enr=altenr:PRINT #3,inv$"Nicht gefunden !"inv$:GOSUB 1450:GOTO 1020
1720 '-----------------------------------------------------------------
1730 'Lesen aus Window 7 und schreiben in RAM-Disk
1740 zeile=1:spalte=1:POKE zeilem,zeile
1750 POKE manzahl,anzahl:POKE mspalte,spalte
1760 POKE ramdi,ramlow:POKE ramdi+1,ramhigh
1770 IF PEEK(ctaste)=224 THEN i = felder
1780 POKE ctaste,0:RETURN
1790 '-----------------------------------------------------------------
1800 links=16:rechts=feldlen(i)
1810 POKE stream,str:POKE linksm,1:POKE rechtsm,rechts:RETURN
1820 '-----------------------------------------------------------------
1830 PRINT #3,"Programm beenden ???":GOSUB 1450
1840 IF x$="n" THEN GOTO 1080
1850 IF x$="j" THEN MODE 2:END
1860 '-----------------------------------------------------------------
1870 CLS #3:PRINT #3,inv$"Moment bitte!"inv$
1880 IF sflag=1 THEN PRINT#3,"Ich suche: "sw$
1890 '
1900 'Berechnung von Offset (off1) und Sectornummern (sf1 bis sf..x)
1910 '
1920 hnr=INT((nr*off)/seclen)
1930 secbytes=(hnr+anzsec)*seclen-(anzsec*seclen)
1940 datbytes=hnr*off
1950 off1=datbytes-secbytes
1960 IF off1>=off THEN off1=off1-off:GOTO 1960
1970 IF off1>seclen THEN off1=off1-seclen:hnr=hnr+1:GOTO 1930
1980 sf1=1+hnr MOD 9:sf2=1+(1+hnr) MOD 9:sf3=1+(2+hnr) MOD 9
1990 '
```

```
2000 bytstand1=nr*off:bytstand2=bytstand1+seclen
2010 bytstand3=bytstand2+seclen
2020 track1=INT((bytstand1)/(seclen*9))
2030 track2=INT((bytstand2)/(seclen*9))
2040 track3=INT((bytstand3)/(seclen*9))
2050 '
2060 POKE mbefehl,befehl
2070 '
2080 POKE mtrack,track1:POKE mformse,sf1:POKE mdrive,1:POKE mbuffer,0
2090 POKE mbuffer+1,&90:CALL &8218:IF track2>159 THEN 2170
2100 '
2110 POKE mtrack,track2:POKE mformse,sf2:POKE mdrive,1:POKE mbuffer,0
2120 POKE mbuffer+1,&92:CALL &8218:IF track3>159 THEN 2170
2130 '
2140 POKE mtrack,track3:POKE mformse,sf3:POKE mdrive,1:POKE mbuffer,0
2150 POKE mbuffer+1,&94:CALL &8218
2160 '
2170 PRINT #5,"Track1: "track1"  ";
2180 PRINT #5,"Sektor: "HEX$(sf1)"  ";
2190 buffoff=off1:begad=buffanf+off1
2200 RETURN
```

8 Verwaltung ist nicht alles!

8.1 Die Daten sollen weiterverarbeitet werden

Haben Sie nach mühevollen Eingaben Ihre Datei erstellt, dann soll mit diesen Daten auch irgendetwas angefangen werden. Die Tatsache alleine, Daten gespeichert zu haben, rechtfertigt den ganzen Aufwand bestimmt nicht.

Abgesehen, von schneller Suche nach irgendwelchen Kriterien will man auch vieles "schwarz auf weiß" haben und das auch möglichst noch in der Form, wie man es wünscht.

Ich kann nun unmöglich alle Kriterien, die bei Ihnen von Interesse sind, kennen. Auch ein Allroundprogramm um auf die von Ihnen erstellten Datenbänke zugreifen zu können und in jeder Form und Weise zu Papier zu bringen, gibt es nicht. Aber ich kann Ihnen die Wege aufzeigen, die beschritten werden können.

Nehmen wir den einfachsten Fall. Die Daten, die mittels des Programmes RAMDISK1 erfaßt wurden, sollen ausgedruckt werden.

Wie Sie wissen, werden die Daten nicht über Strings verwaltet und deswegen scheint es schwierig zu sein, diese auch ausgeben zu lassen; oder?

Kein Problem und dabei lernen Sie nun wieder einen Trick kennen, der gar nicht so ohne ist.

Wie Sie wissen, werden die Daten für das Programm RAMDISK1 unter dem Namen RAMDISK1.DAT auf die Diskette gespeichert. Sie stehen also allen weiteren Programmen zur Verfügung. Gleiches gilt auch für die Maskendaten.

Ein Zusatz- oder Hilfsprogramm kann deshalb sowohl auf die vorhandenen Daten zugreifen, als auch die Masken-

daten lesen. Deshalb gleich wieder ein Programm, das aufzeigt, wie mit den vorhandenen Daten gearbeitet werden kann. Der Name des Programmes ist SEQWAND1.BAS und das bezeichnet auch welche Aufgabe das Programm erfüllt. Es wandelt die binären Daten in ein sequentielles File um. Daß zu diesem Zweck Stringvariable erzeugt werden müssen, ist klar, aber in welcher Weise dies geschieht ist bestimmt für Sie interessant.

Daß nicht nur eine Umwandlung erfolgen kann, sondern daß diese Daten auch auf den Bildschirm oder Drucker ausgegeben werden können, hängt im großen und ganzen nur von dem ab, was Sie (evtl. noch zusätzlich) programmieren.

Ich bin einfach davon ausgegangen, daß ich mittels anderer Programme dann wieder auf ein sequentielles Datenfile zugreifen will und deswegen eine Umwandlung erfolgen soll.

Das nachstehende Listing erfüllt diesen Zweck für die Daten von RAMDISK1.

8.1.1 Basicprogramm: SEQWAND1

```
100 'SEQWAND1.BAS
110 '
120 MODE 2:MEMORY &2FFF:PRINT #1,"Bitte Datendiskette einlegen":CALL &BB06
121 GOSUB 500:PRINT #2,"Teil 1
130 LOCATE 1,3:PRINT"Lese das Binaerfile ":LOAD"ramdisk1.dat",&3000
140 PRINT"Daten gelesen!
150 start=&3000:adresse=start
160 PRINT"Eroeffne SEQ-File!
170 OPENOUT"ramdisk1.seq"
180 LOCATE 40,1:PRINT"Eintragsnummer: "
190 FOR i = 1 TO eintraege
200 CLS #1:PRINT #3,i
210 FOR x = 1 TO felder
220 b$=STRING$(feldlen(x),"*")
230 a$=b$
240 adlhig=INT(adresse/256):adllow=adresse-adlhig*256
250 a2=@ a$+1:a3=@ a$+2
260 POKE a2,adllow:POKE a3,adlhig
270 MID$(b$,1)=a$:a$=""
280 PRINT #1,b$:PRINT #9,b$:b$="":a=FRE("")
290 adresse=adresse+feldlen(x)
300 NEXT x:NEXT i
```

```
310 CLOSEOUT
320 CLEAR:MEMORY &9FFF
330 CLS #1:CLEAR
340 GOSUB 500
350 PRINT #2,"Teil 2
360 DIM feldinhalt$(eintraege,felder)
370 LOCATE 40,1:PRINT"Eintragsnummer: "
380 OPENIN"ramdisk1.seq"
390 FOR i = 1 TO eintraege
400 PRINT #3,i
410 FOR x = 1 TO felder
420 INPUT #9,feldinhalt$(i,x)
430 PRINT #1,feldinhalt$(i,x)
440 'Die naechste Zeile hilft gegen Stringspace full
450 FOR j = 2 TO 5:feldinhalt$(i,j)="":NEXT j
460 NEXT x:CLS #1:NEXT i
470 CLOSEIN
480 MODE 2:END
490 '----------------------
500 MODE 2:PRINT"Programm: SEQWAND1":PRINT:PRINT
510 WINDOW #1,1,79,3,23:WINDOW #2,25,35,1,1
520 WINDOW #3,60,66,1,1
530 PRINT #1,"Lese die Maskendaten":OPENIN"ramdisk1.msk"
540 INPUT #9,buffanf$,buffend$,eintraege,off,felder
550 DIM fenam$(felder),feldlen(felder)
560  FOR i = 1 TO felder
570    INPUT #9,fenam$(i),feldlen(i),xposi,yposi
580  NEXT i
590 CLOSEIN
600 RETURN
```

Was für Anfänger vollkommen unverständlich bei diesem Programm sein wird, ist die Tatsache, daß die Variablenzeiger für a$ erst geholt werden und dann in den Bereich der Daten umgestellt werden.

Diese Methode ist sehr schnell und jeder Umwandlung über die Methode PEEK-CHR$-Umwandlung (auf die ich hier gar nicht näher eingehen will) haushoch überlegen.

Voraussetzung um einen solchen Trick einzusetzen, ist detailierteres Wissen über die Arbeitsweise der CPCs. Aber auch derjenige, der dieses Wissen nicht hat, kann diesen Trick nun einsetzen, wenn er ein paar kleine Punkte berücksichtigt.

Vom Zeitpunkt der "Zeigerverstellung" an, bis zur "Löschung" der Variablen darf keine Garbage-Collect-Routine erfolgen. Die Garbage-Collect-Routine (GCR) ist eine Müllstring-Beseitigungs-Routine der CPCs. Im CPC werden bei der Stringver- und -bearbeitung laufend Strings erzeugt, die nicht mehr gültig bzw. nicht mehr aktuell sind. Wird dem CPC deshalb der Platz zu eng, dann ruft er automatisch eine Routine auf, die diesen "Müll" beseitigt. Durch diese Beseitigung aber werden die Variablenzeiger, also die Speicherstellen die dem CPC zeigen, wo er sie findet, ebenfalls automatisch korrigiert. Durch eine Verstellung von Variablenzeigern und nachfolgender Müllbeseitigung aber, würden die Variablenzeiger falsch gestellt werden!

Erste Voraussetzung ist also, es darf keine GCR auftreten. Als weiteres dürfen auch keine Stringmanipulationen vorgenommen werden, die bewirken, daß die aktuellen Stringinhalte "wandern"!

Und deshalb muß über die Funktion "MID$" gearbeitet werden, und zwar über die zweite Möglichkeit, diesen Befehl einzusetzen!

Außerdem wird im Programm nach der Löschung durch a$ ="" und b$ ="" sofort ein Zwangsaufruf der GCR durch a=FRE("") bewirkt, denn dann stimmen die Zeiger alle "zwangsweise". Die genauere Beschreibung würde nun aber zu weit führen, wie mit dieser Methode gearbeitet werden kann, zeigen die Zeilen 22Ø bis 28Ø sehr anschaulich.

Doch was geschieht in diesen Zeilen noch?

Das Programm wandelt alle Dateneinträge (alle Felder) in die Stringvariable b$ und gibt diese dann jeweils auf den Bildschirm und auf die Diskette aus. Für die Diskettenausgabe wurde vorher in Zeile 17Ø das sequentielle File "RAMDISK1.SEQ" geöffnet.

In Zeile 22Ø wird b$ immer auf die Länge des jeweiligen Feldeintrages gebracht, denn nur dadurch ist es möglich, mittels MID$ in der zweiten Art zu arbeiten,

[illegible]

nd 5ØØ Teil

Die eigentliche Routine geht nur bis Zeile 13Ø u bis 6ØØ, aber ich habe gleich noch einen zweiten hinzugefügt, der das File wieder einlesen kann.

In diesem Teil sehen Sie in Zeile 450 einen kleinen Dreh um "Stringspace full" zu vermeiden, denn wenn zu allen Variablen auch noch der Variablenzeiger kommt, dann reicht der Speicherplatz des CPCs nicht mehr. Das zeigt auch, daß unter den angegebenen Voraussetzungen mittels der Binärfiles mehr Daten verarbeitet werden können, als mittels der Verwaltung über die Stringvariablen!

Während des Programmlaufes werden Ihnen die ganzen umgewandelten Daten auf dem Bildschirm ausgegeben und am Ende steht auf der Diskette ein sequentielles File, das Sie beispielsweise für Ausdruckzwecke verwenden können. (Durch PRINT #8 hätte anstelle der Schirmausgabe auch gleich eine Druckerausgabe erfolgen können!)

Ich hoffe, daß Sie die Grundzüge des Programmes verstehen, um eigene "Weiterverarbeiter" programmieren zu können.

Wollen Sie zusätzlich zu den Daten auch noch die Feldnamen haben, dann zeigt Ihnen nachstehende Variante, wie dies machbar ist.

8.1.2 Basic-Listing: SEQWAND2.BAS

```
100 'SEQWAND2.BAS
110 '
120 MODE 2:MEMORY &2FFF:PRINT #1,"Datendiskette einlegen/Taste":CALL &BB06
130 GOSUB 540:PRINT #2,"Teil 1
140 LOCATE 1,3:PRINT"Lese das Binaerfile ":LOAD"ramdisk1.dat",&3000
150 PRINT #1,"Daten gelesen!
160 start=&3000:adresse=start
170 PRINT#1,"Eroeffne SEQ-File!":CLS #1
180 OPENOUT"ramdisk1.seq"
190 CLS #1:LOCATE 40,1:PRINT"Eintragsnummer: "
200 FOR i = 1 TO felder:PRINT #1,fenam$(i):NEXT i
210 FOR i = 1 TO eintraege
220 PRINT #3,i
230 FOR x = 1 TO felder
240 b$=STRING$(feldlen(x),"*")
250 a$=b$
260 adlhig=INT(adresse/256):adllow=adresse-adlhig*256
270 a2=@ a$+1:a3=@ a$+2
280 POKE a2,adllow:POKE a3,adlhig
290 MID$(b$,1)=a$:a$=""
300 PRINT #4,b$:PRINT #9,b$:b$="":a=FRE("")
310 adresse=adresse+feldlen(x)
```

```
320 NEXT x:CLS #4:NEXT i
330 CLOSEOUT
340 CLEAR:MEMORY &9FFF
350 CLEAR
360 GOSUB 540
370 PRINT #2,"Teil 2
380 DIM feldinhalt$(eintraege,felder)
390 LOCATE 40,1:PRINT"Eintragsnummer: "
400 OPENIN"ramdisk1.seq"
410 CLS #1:FOR i =1 TO felder:PRINT #1,fenam$(i):NEXT i
420 FOR i = 1 TO eintraege
430 PRINT #3,i
440 FOR x = 1 TO felder
450 INPUT #9,feldinhalt$(i,x)
460 IF LEFT$(feldinhalt$(i,x),1)="." THEN feldinhalt$(i,x)=" "
470 PRINT #4,feldinhalt$(i,x)
480 'Die naechste Zeile hilft gegen Stringspace full
490 FOR j = 2 TO 5:feldinhalt$(i,j)="":NEXT j
500 NEXT x:CLS #4:NEXT i
510 CLOSEIN
520 MODE 2:END
530 '----------------------
540 MODE 2:PRINT"Programm: SEQWAND2":PRINT:PRINT
550 WINDOW #1,1,79,3,23:WINDOW #2,25,35,1,1
560 WINDOW #3,60,66,1,1:WINDOW #4,16,79,3,23
570 PRINT #1,"Lese die Maskendaten":OPENIN"ramdisk1.msk"
580 INPUT #9,buffanf$,buffend$,eintraege,off,felder
590 DIM fenam$(felder),feldlen(felder)
600  FOR i = 1 TO felder
610    INPUT #9,fenam$(i),feldlen(i),xposi,yposi
620  NEXT i
630 CLOSEIN
640 PRINT #2,"Maskendaten gelesen!"
650 RETURN
```

Damit wäre die normale Umwandlung von Dateien für RAM-DISK1 abgeschlossen.

Für RAMDISK2 kann die Umwandlung auf die gleiche Weise geschehen. Nur die Bank-Umschaltung muß noch zusätzlich erfolgen. Aber da kann es dann Probleme mit der Speicherkapazität der Schneider-Disketten geben, denn alleine die Binärdaten belegen auf der Diskette 5*17 KByte, kommen zu diesen noch andere Files hinzu, dann wird es eng auf dem Massenspeichermedium. Sorgen Sie deshalb immer dafür, daß die Diskettenkapazität noch ausreichend ist.

Ein Sequentielles Datenfile benötigt immer mehr Platz als ein Binärfile in der von den Programmen verwendeten Art. Obwohl die binären Daten nur 5 * 17 KByte = 85 KByte benötigen, braucht ein sequentielles File dann 87 KByte oder mehr. Das hängt vor allem davon ab, wieviele Felder Sie haben, denn nach jedem Feldinhalt wird ein sogenannter Delimiter geschrieben, also ein Trennzeichen um die einzelnen Strings voneinander zu trennen.

Reicht die Kapazität bedingt durch diese Eigenarten nicht mehr aus, dann hilft evtl. ein zweites Laufwerk aus der Misere, wenn Sie ein sequentielles File mit dem kompletten Datenumfang haben wollen.

Aber meist wird dies ja gar nicht benötigt, es sollte nur gezeigt werden, wie die Wandlung erfolgen kann.

Viel interessanter ist schon, die Daten beispielsweise sortiert ausgeben zu können, oder mit den Daten zu rechnen. Für Rechenaufgaben muß nur die Umwandlung von gelesenen Datenstrings in Zahlvariable erfolgen und dann geht es ganz normal weiter.

8.2 Sortierte Daten

Nun aber zum Sortieren. Es wäre unsinnig, wenn durch jeden Sortiervorgang (für jedes Sortierkriterium) alle Daten "umgeschaufelt" werden würden. Schließlich hat man sich irgendwann ja einmal Gedanken gemacht, warum die Daten in der Reihenfolge stehen, wie Sie eingegeben wurden.

Einfacher ist es, eine (oder mehrere) Sortierdatei(en) erstellen zu lassen.

Wie dies geht, soll wieder durch Beispielprogramme gezeigt werden. Aber an dieser Stelle nun einmal erst die vielleicht von Ihnen schon in Gedanken gestellte Frage:

> Warum ist in all den Programmen dieses Buches bis jetzt noch nichts von Umlauten und DIN-Tastatur geschrieben worden?

Die Antwort ist: Beim Sortieren würde man enorm große Probleme bekommen.

Wer sich schon einmal den Aufwand gemacht hat, ein Sortierprogramm für deutsche Umlaute zu schreiben, der weiß was ich damit meine.

Das Problem ist eben, daß der ASCII-Zeichensatz mit eine der Grundlagen ist, auf dem alle Computer (auch die CPCs) basieren.

Es ist kein Problem, dem CPC Umlaute zu "lernen" und zwar so, daß sowohl die Tastenbelegung stimmt, als auch daß - auf den entsprechenden Tastendruck hin - das Umlautzeichen auf dem Bildschirm erscheint, aber der Programmieraufwand beim Sortieren ist enorm hoch.

Aber gut, ich wäre schon in der Lage gewesen ein derartiges Programm zu schreiben, auch wenn der Umfang etwas hoch gewesen wäre. Aber das bisher Erwähnte ist nicht der einzige Grund weshalb ich es nicht getan habe. Selbst wenn die Zeichen auf dem Bildschirm richtig erscheinen, sagt das noch gar nichts darüber aus, was ein angeschlossener Drucker zu Papier bringt!

Was hilft es denn, wenn auf dem Schirm die Zeichen so stehen, wie Sie es wünschen, der Drucker aber etwas völlig anderes zu Papier bringt. Ich aber kenne Ihren Drucker nicht und kann deshalb auch gar nicht helfen. Bitte haben Sie Nachsicht, daß die Sortierung der Umlaute nicht realisiert wurde. Wer sowohl auf dem Bildschirm, als auch auf dem Drucker Umlaute haben will, kann es mit dem Programm DINTAST.BAS versuchen. Da der CPC 664 und der CPC 6128 eine Druckwandel-Tabelle im Betriebssystem haben, könnte es durchaus sein, daß der Drucker bei Einstellung des deutschen Zeichensatzes alles so druckt, wie es auf dem Schirm steht.

Hier gleich das Programm für eine deutsche Tastatur.

Basic-Programm: DINTAST.BAS

```
100 'DINTAST
110 '
120 SYMBOL AFTER 64
130 '
140 SYMBOL 64,&3C,&60,&3C,&66,&3C,&6,&3C,&0
150 SYMBOL 91,&C6,&10,&38,&6C,&C6,&FE,&C6,&0
160 SYMBOL 92,&C6,&0,&7C,&C6,&C6,&C6,&7C,&0
170 SYMBOL 93,&66,&0,&66,&66,&66,&66,&3C,&0
```

```
180 SYMBOL 123,&66,&0,&78,&C,&7C,&CC,&76,&0
190 SYMBOL 124,&66,&0,&3C,&66,&66,&66,&3C,&0
200 SYMBOL 125,&66,&0,&66,&66,&66,&66,&3E,&0
210 SYMBOL 126,&78,&C6,&C6,&FC,&C6,&C6,&D8,&0
220 '
230 KEY DEF 17,1,43,42,27:KEY DEF 19,1,35,39,29:KEY DEF 22,1,60,62,28
240 KEY DEF 24,1,94,96,30:KEY DEF 25,1,126,63:KEY DEF 26,1,125,93
250 KEY DEF 28,1,123,91:KEY DEF 29,1,124,92:KEY DEF 30,1,45,95
260 KEY DEF 31,1,46,58:KEY DEF 32,1,48,61,31:KEY DEF 39,1,44,59
270 KEY DEF 41,1,55,47:KEY DEF 43,1,122,90,26:KEY DEF 56,1,52,36,163
280 KEY DEF 57,1,51,64,&0:KEY DEF 71,1,121,89,25
```

Durch dieses Programm haben Sie eine deutsche Tastenbelegung und auch die Schirmdarstellung erfolgt in dieser Weise. Ein solches Programm aber vermindert den zur Verfügung stehenden Speicherplatz. Daran sollten Sie denken! Ein CPC wird nach der Aktivierung von Umlauten etc. durch obiges Programm nur noch &A0FB als HIMEM melden.

Nun zu den Sortierprogrammen. Ich habe dabei auf ein ebenfalls beim Franzis-Verlag erschienenes Buch zurückgegriffen. Die Sortierroutinen sind Abwandlungen aus "BASIC: Sortierprogramme" von Rudolf Busch.

Hierüber bin ich Ihnen eine Erklärung schuldig. Es gibt nur ein paar Sortieralgorithmen, auf denen eigentlich alle "Sortierer" aufbauen (Quicksort, Bubble-Sort, Hoare-Sort u.ä.). Diese Programme unterscheiden sich vor allem in der Geschwindigkeit des Sortierens bei vorgegebener Datenstruktur und auch in der Überschaubarkeit für den Anwender.

Da ich aber weder Ihre Datenstruktur kenne noch weiß, was Sie wie berücksichtigen wollen (z. Beispiel Umlautsortierung), erschien es mir nicht vorteilhaft, Ihnen ein Maschinenprogramm zu geben. Dadurch haben Sie einmal die Möglichkeit, die Sortier-Routine durch eine andere zu ersetzen und außerdem können Sie auch noch im entsprechenden Buch nachlesen wie sie funktionieren. Dabei habe ich zwei verschiedene Sortierprogramme benutzt. Das erste ist relativ langsam aber leicht zu verstehen. Beim zweiten wird es schon schwieriger.

8.2.1 Immer der Reihe nach
(Sortieren für RAMDISK1 und RAMDISK2

Zunächst ein Programm für RAMDISK1. Dabei legen Sie durch sortfeld fest, nach welcher Eintragsnummer sortiert werden soll. Dieses Programm ist etwas langsam mit seiner "Sortiererei" aber besser zu durchschauen, wodurch Sie evtl. den Vorteil haben, es für die Umlautsortierung abzuändern. Da Großbuchstaben aber andere ASCII-Zuordnungen haben, als kleine, wird im Programm einfach alles in Großbuchstaben gewandelt. Wer es klein haben will, muß den Befehl UPPER$ gegen LOWER$ austauschen.

8.2.2 Basic-Listing: SORT1.BAS

```
100 'SORT1.BAS
110 '
115 MODE 2:MEMORY &2FFF:PRINT"Datendiskette einlegen/Taste":CALL &BB06
120 GOSUB 1010:PRINT #2,"Teil 1":sortfeld=15
130 LOCATE 1,3:PRINT #4,"Lese das Binaerfile!":LOAD"ramdisk1.dat",&3000
140 PRINT #4,"Maskendaten gelesen!":start=&3000:adresse=start
150 PRINT #4,"Oeffne SEQ-File!":OPENOUT"sort1.seq"
160 PRINT #4,"SEQ-File geoeffnet!"
170 LOCATE 40,1:PRINT"Eintragsnummer: "
180 PRINT #4,"Lese Daten aus dem Speicher und schreibe Sortierdaten auf Disk!
190  FOR i = 1 TO eintraege
200   PRINT #3,i
210     FOR x = 1 TO felder
220      b$=STRING$(feldlen(x),"*"):a$=b$
230      adihig=INT(adresse/256):adilow=adresse-adihig*256
240      a2=@ a$+1:a3=@ a$+2
250      POKE a2,adilow:POKE a3,adihig
260      MID$(b$,1)=a$:a$=""
270      IF x=sortfeld THEN PRINT #1,b$:PRINT #9,b$:b$="":a=FRE("")
280      adresse=adresse+feldlen(x)
290     NEXT x
300  NEXT i:CLS #1
310 PRINT #4,"Schliesse Seq-File!":CLOSEOUT:PRINT #4,"Seq-File geschlossen!"
320 '
330 PRINT #4,"Wechsel zu Programmteil 2!"
340 CLEAR:MEMORY &9FFF:CLS #1:CLEAR:GOSUB 1010
350 DIM feldinhalt$(eintraege),enr$(eintraege),enr(eintraege)
360 PRINT #2,"Teil 2":LOCATE 40,1:PRINT"Eintragsnummer: "
370 PRINT #4,"Oeffne Seq-File!":OPENIN"sort1.seq"
380 PRINT #4,"Seq-File geoeffnet - lese Sortierdaten!"
390  FOR i = 1 TO eintraege
```

```
400   PRINT #3,i:INPUT #9,feldinhalt$(i)
410   feldinhalt$(i)=UPPER$(feldinhalt$(i)):enr(i)=i
420   PRINT #1,feldinhalt$(i),enr$(i)
430  NEXT i:CLOSEIN:PRINT #4,"Seq-File geschlossen!"
440 CLS #1
450 '
460 'Sortierteil
470 PRINT #4,"Wechsel zu Programmteil 3!":PRINT #2,"Teil 3"
480 PRINT #4,"Ich sortiere !":n=eintraege
490 x=n
500 y=0:x=x-1:IF x=0 THEN 590
510  FOR i = 1 TO n-1
520   IF feldinhalt$(i)<=feldinhalt$(i+1) THEN 560
530   hilfs$=feldinhalt$(i):h=enr(i)
540   feldinhalt$(i)=feldinhalt$(i+1):enr(i)=enr(i+1)
550   feldinhalt$(i+1)=hilfs$:enr(i+1)=h:y=y+1:PRINT#3,i
560  NEXT i
570 IF y<>0 THEN 500
580 '
590 PRINT #4,"Sortiervorgang beendet - Ausgabe der sortierten Daten!
600  FOR i = 1 TO n
610   PRINT #1,feldinhalt$(i),enr(i)
620  NEXT i
630 '
640 PRINT #4,"Ausgabe beendet - schreibe Sortier-Nummern als SEQ-File!"
650 PRINT #2,"Teil 4":OPENOUT"sort1.seq"
660  FOR i = 1 TO n
670   PRINT #9,enr(i):PRINT #3,i
680  NEXT i:CLOSEOUT
690 PRINT #4,"Seq-File geschrieben!
700 PRINT #4,"Wechsel zu Programmteil 5"
710 '
720 CLEAR:MEMORY &2FFF:GOSUB 1010
730 PRINT #2,"Teil 5"
740 PRINT #4,"Lese die Datei ein !"
750 LOAD"ramdisk1.dat",&3000
760 PRINT #4,"Datei eingelesen - lese sortierte Eintragsnummern und ...
770 FOR i = 1 TO 2000:NEXT i
780 PRINT#4,".. gebe die Eintraege sortiert auf dem Schirm und auf Disk aus!"
790 WINDOW #1,17,79,5,23:WINDOW #6,1,17,5,23
800 FOR i = 1 TO felder:PRINT #6,fenam$(i):NEXT i
810 OPENIN "sort1.seq"
820 OPENOUT"ramdat1.sor"
830  FOR i = 1 TO eintraege
840   INPUT #9,enr:PRINT #3,enr
850    adresse=&3000+(enr-1)*off
860     FOR x = 1 TO felder
870      b$=STRING$(feldlen(x),"*"):a$=b$
```

```
880       adihig=INT(adresse/256):adilow=adresse-adihig*256
890       a2=@ a$+1:a3=@ a$+2
900       POKE a2,adilow:POKE a3,adihig
910       MID$(b$,1)=a$:a$="":a=FRE(0)
920         PRINT #1,b$:PRINT #9,b$
930       adresse=adresse+feldlen(x)
940     NEXT x:CLS #1
950   NEXT i
960 CLOSEIN:CLOSEOUT
970 CLS #2:CLS #3:CLS #6:PRINT #1,"Programmlauf beendet!"
980 LOCATE 1,25:PRINT"Bitte eine Taste druecken":CALL &BB06
990 MODE 2:END
1000 '----------------------
1010 MODE 2:PRINT"Programm: Sort1":PRINT:PRINT
1020 WINDOW #1,1,79,5,24:WINDOW #2,25,35,1,1
1030 WINDOW #3,60,66,1,1:WINDOW #4,1,79,3,3
1040 PRINT #4,"Lese die Maskendaten":OPENIN"ramdisk1.msk"
1050 INPUT #9,buffanf$,buffend$,eintraege,off,felder
1060 DIM fenam$(felder),feldlen(felder)
1070  FOR i = 1 TO felder
1080     INPUT #9,fenam$(i),feldlen(i),xposi,yposi
1090 fenam$(i)=LEFT$(fenam$(i),15)
1100  NEXT i
1110 CLOSEIN:PRINT#4,"Maskendaten gelesen, File geschlossen!"
1120 RETURN
```

Dieses Programm dient vor allem zu Demonstrationszwecken, genauso wie SORT2.BAS. Es besteht aus insgesamt 5 Programmteilen, die unterschiedliche Aufgaben erledigen. Die kleine Beschreibung ist eine Mischung von beiden Programmen. Teilweise treffen die Aussagen nur für SORT2.BAS zu.

Der erste Teil lädt die Maskendaten und das Datenfile. Sortfeld bestimmt, nach welchem Feld sortiert werden soll. Wünschen Sie aber die Sortierung über zwei Felder hinweg (z.B. Postleitzahl und Ort) dann müssen diese beiden Feldinhalte zusammengefaßt werden.

Der erste Teil schreibt die Daten der Sortierfelder als sequentielles File auf die Diskette, die in sortfeld festgelegt wurde. Nach diesem Vorgang wird Platz im Speicher benötigt, um den Sortiervorgang durchführen zu können. Deshalb wird auf sehr überlistende Art weiterverfahren. Die echten Daten sind im Moment ja nicht wichtig und deshalb kann der reservierte Speicher hierfür, freigegeben werden.

Dann werden die in der normalen Reihenfolge geschriebenen Daten des Sortierfeldes eingelesen (Teile 2) und sortiert (Teil 3). Der Platz hierfür ist ja nun da!

Zu Teil 3 gehört auch noch, daß die nun sortierten Daten auf dem Bildschirm ausgegeben werden. (Kann als Kontrolle dienen, ob das Programm fehlerfrei ist!)

Teil 4 überschreibt dann das alte SEQ-File. Aber nicht mit den sortierten Daten, nein, denn das ist der Trick, mit den Eintragsnummern, die ebenfalls mitsortiert wurden!

Diese Methode ist sehr speicherplatz-sparend auf der Diskette. Während das erste File z. B. 10 KByte auf der Diskette belegte, benötigt das File dann nur noch ca. 2 KByte. Das alte File wurde durch Überschreiben zu einem .BAK-File und kann dann gelöscht werden! Es wäre auch anders möglich, aber Sie sollen ja verschiedene Möglichkeiten kennenlernen.

Anschließend folgt der letzte Teil (5) der nun folgendes macht. Der Speicher wird wieder begrenzt. Die Daten werden wieder eingelesen und nun wird mit zwei gleichzeitig geöffneten Files gearbeitet. Von einem werden die Eintragnummern geholt (diese stehen in der sortierten Folge auf der Diskette) und in das zweiter File werden die kompletten Daten geschrieben. Nach dem gesamten Programmlauf stehen deshalb einmal die sortierten Nummern und zum anderen die komplette Datei (sortiert als SEQ-File) auf der Diskette.

Bei RAMDISK2 wird es auf der Diskette noch "enger", aber es geht! Getestet wurde mit 225 Einträgen (je 363 Zeichen pro Eintrag) und einer Länge des Sortierkriteriums von 40 Zeichen.

Das nachfolgende Programm bietet auch dem Auge ein schöneres Bild!

8.2.3 Basic-Listing: SORT2.BAS (für RAMDISK2)

```
100 'SORT2.BAS fuer CPC 6128
110 '
120 MEMORY &3FFF:MEMORY HIMEM-1:OPENOUT"dummy":CLOSEOUT
130 MODE 2:PRINT"Datendisk einlegen - Taste":CALL &BB06
140 GOSUB 1540:PRINT #2,"Teil 1":sortfeld=15
```

```
150 bswitch=&9FF0
160 LOCATE 1,3:PRINT #4,"Lese die Binaerfiles!"
170 CALL bswitch,0:LOAD"ramdisk2.da1",&4000
180 CALL bswitch,4:LOAD"ramdisk2.da2",&4000
190 CALL bswitch,5:LOAD"ramdisk2.da3",&4000
200 CALL bswitch,6:LOAD"ramdisk2.da4",&4000
210 CALL bswitch,7:LOAD"ramdisk2.da5",&4000
220 bank=0:CALL bswitch,bank
230 '
240 PRINT #4,"Maskendaten gelesen!":start=&4000:adresse=start
250 PRINT #4,"Oeffne SEQ-File!":OPENOUT"sort6128.seq"
260 PRINT #4,"SEQ-File geoeffnet!"
270 LOCATE 40,1:PRINT"Eintragsnummer: "
280 PRINT #4,"Lese Daten aus dem Speicher und schreibe Sortierdaten auf Disk!
290 '
300 adresse=&4000
310 FOR i = 1 TO felder
320  IF i=sortfeld THEN i=felder +1:GOTO 340
330  adresse=adresse+feldlen(i)
340 NEXT i
350 merkad=adresse
360 '
370  FOR i = 1 TO eintraege
380   IF adresse >= &7FFF THEN adresse=merkad:GOTO 500
390   PRINT #3,i
400      b$=STRING$(feldlen(sortfeld),"*"):a$=b$
410      adlhig=INT(adresse/256):adllow=adresse-adlhig*256
420      a2=@ a$+1:a3=@ a$+2
430      POKE a2,adllow:POKE a3,adlhig
440      MID$(b$,1)=a$:a$=""
450      PRINT #1,b$:PRINT #9,b$:b$="":a$="":a=FRE("")
470      adresse=adresse+off
480  NEXT i:CLS #1:GOTO 570
490 '
500 IF bank=7 THEN CALL bswitch,0:GOTO 570
510 IF bank=6 THEN bank=7
520 IF bank=5 THEN bank=6
530 IF bank=4 THEN bank=5
540 IF bank=0 THEN bank=4
550 CALL bswitch,bank:GOTO 390
560 '
570 PRINT #4,"Schliesse Seq-File!":CLOSEOUT:PRINT #4,"Seq-File geschlossen!"
580 '
590 CALL bswitch,0
600 PRINT #4,"Wechsel zu Programmteil 2!"
610 CLEAR:MEMORY &9FEF:CLS #1:CLEAR:GOSUB 1540
620 DIM feldinhalt$(eintraege),enr$(eintraege),enr(eintraege)
630 PRINT #2,"Teil 2":LOCATE 40,1:PRINT"Eintragsnummer: "
```

```
640 PRINT #4,"Oeffne Seq-File!":OPENIN"sort6128.seq"
650 PRINT #4,"Seq-File geoeffnet!"
660 PRINT #4,"Lese Sortierdaten!"
670  FOR i = 1 TO eintraege
680   PRINT #3,i:INPUT #9,feldinhalt$(i)
690   feldinhalt$(i)=UPPER$(feldinhalt$(i)):enr(i)=i
700   PRINT #1,feldinhalt$(i),enr$(i)
710  NEXT i:CLOSEIN:PRINT #4,"Seq-File geschlossen!"
720 CLS #1
730 '
740 'Sortierteil
750 PRINT #4,"Wechsel zu Programmteil 3!":PRINT #2,"Teil 3"
760 PRINT #4,"Ich sortiere !":n=eintraege
770 LOCATE 40,1:PRINT"Sortiervorgang:"
780 DIM b(n)
790 z=1:b(1)=1:b(2)=n
800 li=b(z):re=b(z+1):z=z-2
810 i=li:j=re
820 m=INT((li+re)/2):gw$=feldinhalt$(m)
830 IF feldinhalt$(i)>=gw$ THEN 850
840 i=i+1:GOTO 920
850 IF feldinhalt$(j)<=gw$ THEN 870
860 j=j-1:GOTO 920
870 IF i>j THEN 920
880 h$=feldinhalt$(i):feldinhalt$(i)=feldinhalt$(j):feldinhalt$(j)=h$
890 h=enr(i):enr(i)=enr(j):enr(j)=h
900 i=i+1:PRINT# 3,i
910 j=j-1
920 IF i<=j THEN 830
930 IF i>=re THEN 950
940 z=z+2:b(z)=i:b(z+1)=re
950 re=j
960 IF li<re THEN 810
970 IF z<>-1 THEN 800
980 '
990 PRINT #4,"Sortiervorgang beendet - Ausgabe der sortierten Daten!
1000  FOR i = 1 TO n
1010   PRINT #1,feldinhalt$(i),enr(i)
1020  NEXT i
1030 '
1040 PRINT #4,"Ausgabe beendet - schreibe Sortier-Nummern als SEQ-File!"
1050 PRINT #2,"Teil 4":OPENOUT"sort6128.seq"
1060  FOR i = 1 TO n
1070   PRINT #9,enr(i):PRINT #3,i
1080  NEXT i:CLOSEOUT
1090 PRINT #4,"Seq-File geschrieben!
1100 PRINT #4,"Wechsel zu Programmteil 5"
1110 '
```

```
1120 CLEAR:MEMORY &3FFF:GOSUB 1540
1130 WINDOW #1,17,79,6,24:WINDOW #6,1,17,6,24
1140 WINDOW #5,8,11,25,25:WINDOW #7,60,68,25,25
1150 LOCATE 40,1:PRINT"Eintragsnummer:"
1160 LOCATE 1,25:PRINT"Bank: ":LOCATE 40,25:PRINT"Nummer in der Bank: "
1170 bswitch=&9FF0:CALL bswitch,0
1180 PRINT #2,"Teil 5"
1190 PRINT #4,"Lese den ersten Teil der Datei wieder ein !"
1200 LOAD"ramdisk2.da1",&4000
1210 PRINT #4,"Datei eingelesen - lese sortierte Eintragsnummern und ...
1220 FOR i = 1 TO 2000:NEXT i
1230 PRINT#4,".. gebe die Eintraege sortiert auf dem Schirm und auf Disk aus!"
1240 FOR i = 1 TO felder:PRINT #6,fenam$(i):NEXT i
1250 OPENOUT"sort6128.sor":OPENIN "sort6128.seq"
1260  FOR i = 1 TO eintraege
1270   INPUT #9,enr:PRINT #3,enr;"/"i
1280   block=INT((enr-1)/anzprobank)
1290   IF block=0 THEN bank=0
1300   IF block=1 THEN bank=4
1310   IF block=2 THEN bank=5
1320   IF block=3 THEN bank=6
1330   IF block=4 THEN bank=7
1340   CALL bswitch,bank
1350   IF enr>anzprobank THEN enr=enr-anzprobank:GOTO 1350
1360   PRINT #5,bank:PRINT #7,enr
1370 '
1380   adresse=&4000+(enr-1)*off
1390     FOR x = 1 TO felder
1400      b$=STRING$(feldlen(x),"*"):a$=b$
1410      adlhig=INT(adresse/256):adllow=adresse-adlhig*256
1420      a2=@ a$+1:a3=@ a$+2
1430      POKE a2,adllow:POKE a3,adlhig
1440      MID$(b$,1)=a$
1450        PRINT #1,b$:PRINT #9,b$:a$="":b$="":a=FRE("")
1470      adresse=adresse+feldlen(x)
1480     NEXT x:CLS #1
1490  NEXT i:CLS #1:CLS #4:PRINT#4,"Schliesse die Files!":CLOSEIN:CLOSEOUT
1500 CLS #2:CLS #3:CLS #6:PRINT #1,"Programmlauf beendet!"
1510 LOCATE 1,25:PRINT"Bitte eine Taste druecken":CALL &BB06
1520 MODE 2:END
1530 '----------------------
1540 MODE 2:PRINT"Programm: SORT2 (6128)":PRINT:PRINT
1550 d$="*.bak":|ERA,d$
1560 WINDOW #1,1,79,5,24:WINDOW #2,25,35,1,1
1570 WINDOW #3,60,75,1,1:WINDOW #4,1,79,3,3
1580 PRINT #4,"Lese die Maskendaten":OPENIN"ramdisk2.msk"
1590 INPUT #9,buffanf$,buffend$,eintraege,off,felder
1600 anzprobank=INT(eintraege/5)
```

```
1610 DIM fenam$(felder),feldlen(felder)
1620  FOR i = 1 TO felder
1630    INPUT #9,fenam$(i),feldlen(i),xposi,yposi
1640 fenam$(i)=LEFT$(fenam$(i),15)
1650  NEXT i
1660 CLOSEIN:PRINT#4,"Maskendaten gelesen, File geschlossen!"
1670 RESTORE:FOR i = &9FF0 TO &9FF6:READ x$:x=VAL("&"+x$):POKE i,x:NEXT
1680 RETURN
1690 DATA dd,7e,00,cd,5b,bd,c9
```

Da die Bildschirmausgaben sehr viel Zeit kosten, hier nun eine Kurzversion für die Benutzung mit einem Laufwerk.

8.2.4 Basic-Listing: SORTKURZ.BAS

```
100 MODE 2:PRINT"Sortierprogramm SORTKURZ
110 PRINT"Bitte Datendisk einlegen - Taste druecken":CALL &BB06
120 MEMORY &3FFF:GOSUB 640:sortfeld=15
130 PRINT"Lese Binaerfiles":CALL bswitch,0:LOAD"ramdisk2.da1",1a
140 CALL bswitch,4:LOAD"ramdisk2.da2",1a:CALL bswitch,5:LOAD"ramdisk2.da3",1a
150 CALL bswitch,6:LOAD"ramdisk2.da4",1a:CALL bswitch,7:LOAD"ramdisk2.da5",1a
160 bank=0:CALL bswitch,bank:OPENOUT"sort6128.seq":adresse=1a
170 FOR i = 1 TO felder:IF i=sortfeld THEN i=felder +1:GOTO 190
180 adresse=adresse+feldlen(i)
190 NEXT i:merkad=adresse
200 FOR i = 1 TO eintraege:IF adresse >= &7FFF THEN adresse=merkad:GOTO 260
210 b$=STRING$(feldlen(sortfeld),"*"):a$=b$
220 ad1hig=INT(adresse/256):ad1low=adresse-ad1hig*256
230 a2=@ a$+1:a3=@ a$+2:POKE a2,ad1low:POKE a3,ad1hig
240 MID$(b$,1)=a$:a$="":PRINT #9,b$:b$="":a=FRE(""):adresse=adresse+off
250 NEXT i::GOTO 300
260 IF bank=7 THEN CALL bswitch,0:GOTO 300
270 IF bank=6 THEN bank=7 ELSE IF bank=5 THEN bank=6
280 IF bank=4 THEN bank=5 ELSE IF bank=0 THEN bank=4
290 CALL bswitch,bank:adresse=merkad:GOTO 210
300 CLOSEOUT:CALL bswitch,0:CLEAR:MEMORY &9FEF:CLEAR:GOSUB 640
310 DIM feldinhalt$(eintraege),enr$(eintraege),enr(eintraege)
320 OPENIN"sort6128.seq":FOR i = 1 TO eintraege:INPUT #9,feldinhalt$(i)
330 feldinhalt$(i)=UPPER$(feldinhalt$(i)):enr(i)=i:NEXT i:CLOSEIN
340 PRINT"Ich sortiere!":n=eintraege:DIM b(n):z=1:b(1)=1:b(2)=n
350 li=b(z):re=b(z+1):z=z-2
360 i=li:j=re:m=INT((li+re)/2):gw$=feldinhalt$(m)
370 IF feldinhalt$(i)>=gw$ THEN 390
380 i=i+1:GOTO 440
390 IF feldinhalt$(j)<=gw$ THEN 410
```

```
400 j=j-1:GOTO 440
410 IF i>j THEN 440
420 h$=feldinhalt$(i):feldinhalt$(i)=feldinhalt$(j):feldinhalt$(j)=h$
430 h=enr(i):enr(i)=enr(j):enr(j)=h:i=i+1:j=j-1
440 IF i<=j THEN 370 ELSE IF i>=re THEN 460
450 z=z+2:b(z)=i:b(z+1)=re
460 re=j:IF li<re THEN 360
470 IF z<>-1 THEN 350
480 PRINT"Ich schreibe die sortierten Eintragsnummern!
490 OPENOUT"sort6128.seq":FOR i = 1 TO n:PRINT #9,enr(i):NEXT i:CLOSEOUT
500 CLEAR:MEMORY &3FFF:GOSUB 640:CALL bswitch,0
510 PRINT"Ich lade den ersten Datenblock!":LOAD"ramdisk2.da1",la
520 OPENIN "sort6128.seq":OPENOUT"rd6128.sor"
530 PRINT"Lesen des sortierten Files und Schreiben als SEQ-File!
540 FOR i = 1 TO eintraege:INPUT #9,enr:block=INT((enr-1)/anzprobank)
550 IF block=0 THEN bank=0 ELSE IF block=1 THEN bank=4
560 IF block=2 THEN bank=5 ELSE IF block=3 THEN bank=6
570 IF block=4 THEN bank=7
580 CALL bswitch,bank
590 IF enr>anzprobank THEN enr=enr-anzprobank:GOTO 590
600 adresse=la+(enr-1)*off:FOR x = 1 TO felder:b$=STRING$(feldlen(x),"*"):a$=b$
610 adlhig=INT(adresse/256):adllow=adresse-adlhig*256:a2=@ a$+1:a3=@ a$+2
620 POKE a2,adllow:POKE a3,adlhig:MID$(b$,1)=a$:a$="":a=FRE(0):PRINT #9,b$
630 adresse=adresse+feldlen(x):NEXT x:NEXT i:CLOSEIN:CLOSEOUT:MODE 2:END
640 OPENIN"ramdisk2.msk":INPUT #9,buffanf$,buffend$,eintraege,off,felder
650 anzprobank=INT(eintraege/5):DIM fenam$(felder),feldlen(felder)
660  FOR i = 1 TO felder:INPUT #9,fenam$(i),feldlen(i),xposi,yposi
670 fenam$(i)=LEFT$(fenam$(i),15):NEXT i:la=&4000:bswitch=&9FF0:CLOSEIN
680 RESTORE:FOR i = bswitch TO bswitch+6:READ x$:x=VAL("&"+x$):POKE i,x:NEXT
690 a$="*.bak":|ERA,@a$
700 RETURN
710 DATA dd,7e,00,cd,5b,bd,c9
```

Da ich schon des öfteren hingewiesen habe, daß auch mit zwei Laufwerken gearbeitet werden kann, hier zur Demonstration eine Programmversion mit zwei Laufwerken. Es spielt dabei keine Rolle, ob das zweite Laufwerk zur 3-Zoll- oder zur 5,25-Zoll-Familie gehört.

8.2.5 Basic-Listing: SORTK2.BAS

```
100 MODE 2:PRINT"Sortierprogramm SORTK2
110 PRINT"Bitte Datendisketten einlegen - Taste druecken":CALL &BB06
120 |B:a$="*.*":|ERA,@a$:|A
130 MEMORY &3FFF:GOSUB 670:sortfeld=15
140 PRINT"Lese Binaerfiles":CALL bswitch,0:|A:LOAD"ramdisk2.da1",la
```

```
150 CALL bswitch,4:LOAD"ramdisk2.da2",la:CALL bswitch,5:LOAD"ramdisk2.da3",la
160 CALL bswitch,6:LOAD"ramdisk2.da4",la:CALL bswitch,7:LOAD"ramdisk2.da5",la
170 bank=0:CALL bswitch,bank:!B:OPENOUT"sort6128.seq":adresse=la
180 PRINT"Schreibe Datenfile zum Sortieren
190 FOR i = 1 TO felder:IF i=sortfeld THEN i=felder +1:GOTO 210
200 adresse=adresse+feldlen(i)
210 NEXT i:merkad=adresse
220 FOR i = 1 TO eintraege:IF adresse >= &7FFF THEN adresse=merkad:GOTO 280
230 b$=STRING$(feldlen(sortfeld),"*"):a$=b$
240 adlhig=INT(adresse/256):adllow=adresse-adlhig*256
250 a2=@ a$+1:a3=@ a$+2:POKE a2,adllow:POKE a3,adlhig
260 MID$(b$,1)=a$:a$="":PRINT #9,b$:b$="":a=FRE(""):adresse=adresse+off
270 NEXT i::GOTO 320
280 IF bank=7 THEN CALL bswitch,0:GOTO 320
290 IF bank=6 THEN bank=7 ELSE IF bank=5 THEN bank=6
300 IF bank=4 THEN bank=5 ELSE IF bank=0 THEN bank=4
310 CALL bswitch,bank:adresse=merkad:GOTO 230
320 CLOSEOUT:CALL bswitch,0:CLEAR:MEMORY &9FEF:CLEAR:GOSUB 670
330 DIM feldinhalt$(eintraege),enr$(eintraege),enr(eintraege)
340 !B:OPENIN"sort6128.seq":FOR i = 1 TO eintraege:INPUT #9,feldinhalt$(i)
350 feldinhalt$(i)=UPPER$(feldinhalt$(i)):enr(i)=i:NEXT i:CLOSEIN
360 PRINT"Ich sortiere!":n=eintraege:DIM b(n):z=1:b(1)=1:b(2)=n
370 li=b(z):re=b(z+1):z=z-2
380 i=li:j=re:m=INT((li+re)/2):gw$=feldinhalt$(m)
390 IF feldinhalt$(i)>=gw$ THEN 410
400 i=i+1:GOTO 460
410 IF feldinhalt$(j)<=gw$ THEN 430
420 j=j-1:GOTO 460
430 IF i>j THEN 460
440 h$=feldinhalt$(i):feldinhalt$(i)=feldinhalt$(j):feldinhalt$(j)=h$
450 h=enr(i):enr(i)=enr(j):enr(j)=h:i=i+1:j=j-1
460 IF i<=j THEN 390 ELSE IF i>=re THEN 480
470 z=z+2:b(z)=i:b(z+1)=re
480 re=j:IF li<re THEN 380
490 IF z<>-1 THEN 370
500 PRINT"Ich schreibe die sortierten Eintragsnummern!
510 OPENOUT"sort6128.seq":FOR i = 1 TO n:PRINT #9,enr(i):NEXT i:CLOSEOUT
520 CLEAR:MEMORY &3FFF:GOSUB 670:CALL bswitch,0
530 !A:PRINT"Ich lade den ersten Datenblock!":LOAD"ramdisk2.da1",la
540 !B:OPENIN "sort6128.seq":OPENOUT"rd6128.sor"
550 PRINT"Lesen des sortierten Files und Schreiben als SEQ-File!
560 FOR i = 1 TO eintraege:INPUT #9,enr:block=INT((enr-1)/anzprobank)
570 IF block=0 THEN bank=0 ELSE IF block=1 THEN bank=4
580 IF block=2 THEN bank=5 ELSE IF block=3 THEN bank=6
590 IF block=4 THEN bank=7
600 CALL bswitch,bank
610 IF enr>anzprobank THEN enr=enr-anzprobank:GOTO 610
620 adresse=la+(enr-1)*off:FOR x = 1 TO felder:b$=STRING$(feldlen(x),"*"):a$=b$
```

```
630 adlhig=INT(adresse/256):adllow=adresse-adlhig*256:a2=@ a$+1:a3=@ a$+2
640 POKE a2,adllow:POKE a3,adlhig:MID$(b$,1)=a$:a$="":a=FRE(Ø):PRINT #9,b$
650 adresse=adresse+feldlen(x):NEXT x:NEXT i:CLOSEIN:CLOSEOUT:MODE 2
660 CAT:|A:CAT:END
670 |A:OPENIN"ramdisk2.msk":INPUT #9,buffanf$,buffend$,eintraege,off,felder
680 anzprobank=INT(eintraege/5):DIM fenam$(felder),feldlen(felder)
690  FOR i = 1 TO felder:INPUT #9,fenam$(i),feldlen(i),xposi,yposi
700 fenam$(i)=LEFT$(fenam$(i),15):NEXT i:la=&4000:bswitch=&9FFØ:CLOSEIN
710 RESTORE:FOR i = bswitch TO bswitch+6:READ x$:x=VAL("&"+x$):POKE i,x:NEXT
720 |B:a$="*.bak":|ERA,@a$
730 RETURN
740 DATA dd,7e,00,cd,5b,bd,c9
```

Damit glaube ich, wurden genügend Wege aufgezeigt, wie Daten umgewandelt und sortiert werden können. Auch für die Programme RANDOMD1 und RANDOMD1 sind diese Methoden einsetzbar. Ob Sie allerdings - gerade beim Programm RANDOMD2 - noch genügend Speicherplatz für den Sortiervorgang im CPC oder für die Files auf der Diskette haben, kann ich leider nicht aussagen. Falls Sie Platzmangel haben, hilft dann die Längenbegrenzung der zu sortierenden Einträge weiter (b$=LEFT$(b$,x)).

8.3 Backup für Vortex-X-Modul-Laufwerke

Bei Dateien ist es sehr wichtig, daß ausreichender Datenschutz gewährleistet ist. Ich meine damit nicht den Datenschutz, von dem sonst immer gesprochen wird, nämlich der unbefugten Benutzung von Daten, sondern die Sicherheit, daß bei Defekt oder Verlust des Datenträgers nicht alles wieder neu eingegeben werden muß.

FÜr das Programm RANDOMD2 aber bedeutet das, daß für BACKUP-Disketten insgesamt 72Ø KByte kopiert werden müssen. Eine Menge, die nicht immer Freude bereitet. Dies werden vor allem die CPC-Besitzer bemerken, die nur einen CPC mit 64 KByte RAM besitzen.

Aber auch bei diesen sollten die Sicherheitskopien nicht vernachlässigt werden.

Ein "Viertelstündchen" für die Sicherheitskopien pro "Arbeitstag" ist immer noch besser als tagelange Neueingaben.

Nun ein paar Worte zum ersten Programm. Auch hier habe ich wieder berücksichtigt, daß nichts oberhalb von &9FFF geschieht. Erweiterungen für den CPC dürften deshalb nicht beeinflußt werden und auch umgekehrt.

Wichtig zu vermerken ist noch folgendes:

> Durch viele CP/M-Kopierprogramme werden die ersten Spuren, die ja für CP/M reserviert sind, nicht kopiert!

Auf diese werden durch den Programmlauf jener Hilfsprogramme anstelle von Daten CP/M-Sektoren geschrieben.

Die nachfolgenden Programme verhalten sich anders. Sie lesen jeden Sektor vom Original (der Quelldiskette) und schreiben sie auf die Zieldiskette.

Deshalb können die beiden Programme DUPLI1 und DUPLI2 immer dann eingesetzt werden, wenn Komplett-Kopien von Disketten erstellt werden sollen. Es muß also nicht immer erst CP/M gestartet werden (eine Sache, die mich immer wieder bei vielen Hilfsroutinen stört)!

Denken Sie aber immer daran wenn Sie Kopien erstellen, daß Sie dann, wenn Sie echte CP/M-Disketten kopieren und diese weitergeben das COPYRIGHT verletzen.

Bei beiden Programmen stehen die Daten für das Maschinenprogramm im Basiclisting selbst. Diese sind wieder die Daten für den Diskettendirekt-Zugriff (DISC2) nur für einen anderen Speicherbereich assembliert.

Die Programme teilen immer mit, was Sie zu tun haben. Das Programm DUPLI1 arbeitet im Prinzip wie das Programm RANDOMD2. DUPLI2 ist eine Kombination von RAMDISK2 mit RANDOMD2. Deshalb entfallen nähere Programmbeschreibungen für diese Hilfsroutinen.

Aber einen Punkt sollten Sie beherzigen: Solange Sie sich nicht sicher sind, daß das Programm fehlerfrei abgeschrieben wurde, sollten Sie niemals mit "wichtigen" Disketten arbeiten und vor allem sollte auch auf der Quelldiskette der Schreibschutz "eingeschaltet" sein. Es könnte ja sein, daß Sie versehentlich von der Zieldiskette lesen lassen und dann auf die Quelldiskette schreiben wollen. Gut, wenn dann der Schreibschutz vorhanden ist!

Beide Programme arbeiten unter VDOS2. Das heißt, Sie laden das Programm DUPLI1 von Ihrer Programmdiskette (beispielsweise vom eingebauten 3-Zoll-Laufwerk beim CPC 664), der Kopierzugriff erfolgt aber auf das X-Laufwerk.

Die Zieldisketten müssen bereits formatiert sein, denn beide Kopierprogramme enthalten keinen Formatierteil.

8.3.1 Basic-Listing: DUPLI1.BAS

```
100 'DUPLI1.BAS
110 '
120 MEMORY &2FCF:CALL &BB4E:MODE 2
130 PRINT"DUPLI1.BAS - Backup-Programm fuer Vortex-5,25-Zoll (28K) (c) by L.M
140 LOCATE 20,12:PRINT"Buffer:":LOCATE 1,25:PRINT"JOB:"
150 '
160 jobfenster=7:Lesefenster=1:Schreibfenster=2:bufferfenster=3
170 spurfenster=4:bufferfenster=5
180 '
190 WINDOW #jobfenster,5,79,25,25:WINDOW #Lesefenster,5,60,4,9
200 WINDOW #Schreibfenster,5,60,15,20:WINDOW #spurfenster,74,79,1,1
```

```
210 WINDOW #bufferfenster,28,35,12,12
220 '
230 PRINT #jobfenster,"Lesen des Maschinen/Programmes!
240 GOSUB 860:PRINT #jobfenster,"Maschinen-Programm erzeugt
250 '
260 'Festlegungen und Definitionen fuer die Maschinen-Routine
270 mdrive=&2FFA:mbefehl=&2FF6:mbuffer=&2FFE:mformse=&2FFC
290 mtrack=&2FFB:mpgstart=&2FD0
300 '
310 'Befehle fuer den Diskettenzugriff
320 lesen=&84:schreiben=&85
330 '
340 'sonstige Festlegungen
350 buffstart=&3000:seclen=512:secanzges=160:warte=&BB06
360 job1$="Bitte die Quelldiskette einlegen und eine Taste druecken!"+CHR$(7)
370 job2$="Bitte die Zieldiskette einlegen und eine Taste druecken!"+CHR$(7)
380 inv$=CHR$(24)
390 '
400 PRINT #lesefenster,inv$:CLS #lesefenster
410 PRINT #schreibfenster,inv$:CLS #schreibfenster
420 '
430 FOR teil = 0 TO 28:CLS # lesefenster:CLS # schreibfenster
440 'Lesen
450 PRINT #Jobfenster,job1$:CALL warte:befehl=lesen
460 PRINT # jobfenster,"Lese Quelldiskette .....
470    buffer=buffstart
480    FOR spur = 0 TO 5
490      track=teil*6+spur:IF track=secanzges THEN 580
500      PRINT #lesefenster,"Spur    :"track"     ";
510           FOR Sektor=1 TO 9
520            PRINT #lesefenster,sektor;" ";
530            GOSUB 730
540           NEXT sektor:PRINT #lesefenster
550    NEXT spur
560 '
570 'Schreiben
580 PRINT #Jobfenster,job2$:CALL warte:befehl=schreiben
590 PRINT # jobfenster,"Schreibe Zieldiskette .....
600    buffer =buffstart
610    FOR spur = 0 TO 5
620       track=teil*6+spur:IF track=secanzges THEN 840
630       PRINT #Schreibfenster,"Spur    :"track"     ";
640           FOR Sektor=1 TO 9
650            PRINT #schreibfenster,sektor;" ";
660           GOSUB 730
670           NEXT sektor:PRINT #schreibfenster
680    NEXT spur
690 '
```

```
700 NEXT teil:END
710 '
720 'Disketten-Zugriffe
730 PRINT #bufferfenster,HEX$(buffer)
740 PRINT #spurfenster,track
750 POKE mtrack,track:POKE mdrive,1
760 bufferhigh=INT(buffer/256):bufferlow=buffer-bufferhigh*256
770 POKE mbuffer,bufferlow:POKE mbuffer+1,bufferhigh
780 POKE mformse,sektor
790 POKE mbefehl,befehl
800 CALL mpgstart
810 buffer=buffer+seclen
820 RETURN
830 '
840 MODE 2:PRINT"Die Diskette wurde dupliziert":END
850 '
860 a= 12240:e= 12279:zb=2000
870 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 900
880 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
890 ps=0:d$="":IF i=e THEN 930 ELSE i=i-1:zb=zb+1:GOTO 920
900 d$="&"+d$
910 POKE i,VAL(d$):ps=ps+VAL(d$)
920 IF i < e THEN NEXT i
930 RETURN
940 '
2000 DATA 3A,FA,2F,32,FA,2F,21,F6,2F,CD,D4,BC,22,F7,2F,79,&0822
2001 DATA 32,F9,2F,21,FA,2F,5E,21,FB,2F,56,21,FC,2F,4E,2A,&0667
2002 DATA FE,2F,DF,F7,2F,C9,84,&047F
```

Diejenigen, die gezwungen sind, mit diesem Programm zu arbeiten, (weil sie nur einen 64-KByte-CPC haben) werden die CPC 6128-Besitzer beneiden, denn mit dem nachstehenden Programm für diesen CPC-Typ braucht der Diskettentausch-Vorgang nur je neunmal durchgeführt zu werden. Also neunmal "Schreiben" und neunmal "Lesen".

Das Programm ist im wesentlichen so wie eine Kombination von RAMDISK2 mit RANDOMD2. Volle 80 KByte werden dabei als Datenbuffer benutzt.

8.3.2 Basic-Listing: DUPLI2.BAS

```
100 MEMORY &2FCF:CALL &BB4E:MODE 2:inv$=CHR$(24)
110 PRINT inv$"DUPLI2 - Disketten-Duplizier-Programm (c) by Lothar Miedel"inv$
120 LOCATE 2,3:PRINT"Spur:":LOCATE 65,3:PRINT"Bank: "
130 LOCATE 20,3:PRINT"Laufende Sektor-Nummer:
```

```
140 LOCATE 60,5:PRINT"Bufferadresse:
150 LOCATE 1,25:PRINT"JOB:
160 '
170 spurfenster=1:lfdsecfenster=2:sektorfenster=3:bankfenster=4
180 secanzfenster=5:bufferfenster=6:jobfenster=7
190 '
200 buffstart=&4000:buffer=buffstart:seclen=512
210 j1$="Bitte":j2$="diskette einlegen und eine Taste druecken!"
220 '
230 WINDOW #spurfenster,8,14,3,3
240 WINDOW #lfdsecfenster,46,52,3,3
250 WINDOW #sektorfenster,2,77,10,19
260 WINDOW #bankfenster,75,79,3,3
270 WINDOW #bufferfenster,75,79,5,5
280 WINDOW #jobfenster,6,79,25,25
290 '
300 PRINT #jobfenster,"Erzeuge das Maschinen-Programm!
310 GOSUB 990:PRINT #jobfenster,"Maschinen-Programm erzeugt!
320 '
330 'Festlegungen und Definitionen fuer die Maschinen-Routine
340 mdrive=&2FFA:mbefehl=&2FF6:mbuffer=&2FFE:mformse=&2FFC
350 mtrack=&2FFB:mpgstart=&2FD0:blsw=&3000:warte=&BB06:flush=&BD3D
360 '
370 'Befehle fuer den Diskettenzugriff
380 lesen=&84:schreiben=&85:befehl=lesen:b$="Lese"
390 '
400 PRINT #sektorfenster,inv$:CLS #Sektorfenster
410 PRINT #spurfenster,track
420 PRINT #bufferfenster,HEX$(buffer)
430 '
440 PRINT #bankfenster,bank
450 PRINT #jobfenster,j1$;" Quell";j2$:CALL &BB06
460 PRINT #jobfenster,b$;" Diskettensektoren
470 alti=1:befehl=lesen
480 FOR i = 1 TO 1440
490 secnr=i-1:PRINT #lfdsecfenster,i
500 track=INT(secnr/9)
510 sektor=(secnr MOD 9)+1
520 GOSUB 880
530 pruefzahl= (secnr MOD 32)+1
540 IF alttrack = track THEN 560
550 PRINT #spurfenster,track:alttrack=track:PRINT #sektorfenster,"  ";
560 PRINT #sektorfenster,sektor" ";
570 buffer=buffer+seclen:PRINT #bufferfenster,HEX$(buffer)
580 GOSUB 620
590 NEXT i:PRINT #jobfenster,"Diskette dupliziert!"
600 END
610 '
```

```
620 IF pruefzahl <>32 THEN 670
630 block=block+1:buffer=buffstart
640 IF block=5 THEN block=0: GOSUB 690
650 IF block=0 THEN bank=block ELSE bank=block+3
660 PRINT #bankfenster,bank
670 RETURN
680 '
690 PRINT #bankfenster,inv$:PRINT #bankfenster,bank
700 LOCATE 35,5:PRINT #jobfenster,"Wechsel: "wechsel+1:wechsel=wechsel+1
710 IF befehl=lesen THEN PRINT #Jobfenster,j1$;" Ziel";j2$
720 IF befehl=schreiben THEN PRINT #Jobfenster,j1$;" Quell";j2$
730 BORDER 23:CALL flush:CALL warte:PRINT #bankfenster,bank
740 CLS #sektorfenster:BORDER 1
750 '
760 IF befehl=schreiben THEN befehl=lesen:b$="Lese":GOTO 800
770 IF befehl=lesen THEN befehl=schreiben:b$="Schreibe":i=i-160
780 sektor=(i MOD 9)
790 '
800 CLS #sektorfenster
810 PRINT #jobfenster,b$;" Diskettensektoren
820 IF sektor=9 THEN 840
830 FOR k=1 TO sektor:PRINT #Sektorfenster,"    ";:NEXT k
840 PRINT #bankfenster,inv$
850 RETURN
860 '-----------------------------------------------
870 'Disketten-Zugriffe
880 PRINT #bufferfenster,HEX$(buffer)
890 PRINT #spurfenster,track
900 CALL blsw,bank
910 POKE mtrack,track:POKE mdrive,1
920 bufferhigh=INT(buffer/256):bufferlow=buffer-bufferhigh*256
930 POKE mbuffer,bufferlow:POKE mbuffer+1,bufferhigh
940 POKE mformse,sektor
950 POKE mbefehl,befehl
960 CALL mpgstart
970 RETURN
980 '
990 a= 12240:e= 12295:zb=2000
1000 FOR i =a TO e:READ d$:IF LEFT$(d$,1)<>"&" THEN 1030
1010 IF ps<>VAL(d$) THEN PRINT"Fehler Zeile "zb:END
1020 ps=0:d$="":IF i=e THEN 1060 ELSE i=i-1:zb=zb+1:GOTO 1050
1030 d$="&"+d$
1040 POKE i,VAL(d$):ps=ps+VAL(d$)
1050 IF i < e THEN NEXT i
1060 '
1070 RETURN
1080 '
2000 DATA 3A,FA,2F,32,FA,2F,21,F6,2F,CD,D4,BC,22,F7,2F,79,&0822
```

```
2ØØ1 DATA 32,F9,2F,21,FA,2F,5E,21,FB,2F,56,21,FC,2F,4E,2A,&Ø667
2ØØ2 DATA FE,2F,DF,F7,2F,C9,84,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,&47f
2ØØ3 DATA dd,7e,ØØ,cd,5b,bd,c9,&Ø4Ø9
```

Am Ende dieser beiden Listings kann ich mich nur wiederholen:

BACKUP.....BACKUP.....BACKUP!

Ich weiß es ist lästig, dauernd Disketten tauschen zu müssen, aber irgendwann macht es sich auch bei Ihnen einmal bezahlt.

Gerade "Profis", die schon lange mit Computern arbeiten und es eigentlich wissen sollten, nehmen die Sache mit der Sicherheit oft auf die leichte Schulter.

"Es ist mir schon lange nichts mehr passiert!"

Diesen Satz aussprechen zu können, ist schön und zeigt auch, daß derjenige gut mit seinem Rechnersystem umgehen kann, aber wenn es eben passiert, dann ist es zu spät!

8.4 Disketten in Form gebracht
(Spezial-Formate für Disketten)

Bei den Programmen für den RANDOM-Zugriff werden Disketten in unterschiedlichem Maße mehr genutzt, als es sonst geschieht.

Das wichtigere dabei ist aber, daß die Disketten so vorbereitet werden, daß das Füllzeichen geschrieben wird.

Deshalb sind auch spezielle Diskettenformate erforderlich.

Hier das erste Formatier-Programm, das die 3-Zoll-Disketten vorbereitet.

8.4.1 Basic-Listing: SPEZFORM.BAS

```
100 'SPEZFORM.BAS
110 '
120 MODE 1:PRINT"Spezialformatier-Programm"
130 PRINT"=========================":PRINT:PRINT
140 '
150 MEMORY &7FFF
160 LOAD"randomd1.mpg",&8000
170 PRINT"Bitte die zu formatierende Diskette
180 PRINT"einlegen und eine Taste druecken!
190 CALL &BB06:PRINT:PRINT
200 suche=&81BD:exe=&81D4:formtab=&81F4:buffer=&9000:xx=0:usernummer=0
210 extdpb1=&A890:extdpb2=&A8D0
220 fillbyte=&2E:lesen=&84:schreiben=&85:formatieren=&86:drive=0
230 IF drive=0 THEN extdpb=extdpb1 ELSE extdpb=extdpb2
240 '
250 'Komplette Diskette formatieren
260 befehl=formatieren:CALL suche,befehl
270 IF drive=0 THEN POKE extdpb+19,fillbyte
280 FOR i = 0 TO 39
290 FOR j=0 TO 8:POKE (formtab+j*4),i:NEXT j:track=i
300 CALL exe,formtab,&C1,track,drive:NEXT i
310 '
320 'Bufferdaten vorbereiten
330 name$="DATEI!!!":ext$="DAT":schutz=ASC(LEFT$(ext$,1))+&80
340 MID$(ext$,1,1)=CHR$(schutz):gesamtname$=name$+ext$
350 FOR i = 0 TO 10:w(i)=ASC(MID$(gesamtname$,i+1,1)):NEXT i
360 block=2:adresse=buffer-1
370 FOR i = buffer TO buffer+&170:POKE i,0:NEXT i
```

```
380 '
390 FOR x=0 TO 11:adresse=adresse+1:POKE adresse,usernummer
400 '
410 FOR i = 1 TO 11:adresse=adresse+1:POKE adresse,w(i-1):NEXT i
420 adresse=adresse+4:FOR i = 1 TO 16:adresse=adresse+1
430 POKE adresse,block:block=block+1:IF block=&B4 THEN x=12:i=17:GOTO 440
440 NEXT i:NEXT x
450 'Bufferinhalt in 1. Sektor schreiben
460 befehl=schreiben:CALL suche,befehl
470 CALL exe,buffer,&C1,0,0
480 PRINT"Der CAT-Befehl zeigt, dass die Diskette
490 PRINT"Spezialformat hat und nun 359 Sektoren
500 PRINT"zur Verfuegung stehen!
510 'Fillerbyte wieder korrigieren
520 POKE extdpb,&E5
530 IF drive=1 THEN |B
540 CAT
550 IF drive=1 THEN |A
```

Neben dem Füllzeichen wird die 3-Zoll-Diskette auch so vorbereitet, daß unter normalen Bedingungen kein File mehr darauf abgespeichert werden kann. Eine zusätzliche Maßnahme, die Daten zu schützen.

Wird beim vorstehenden Programm der Wert von drive auf 1 geändert, dann erfolgt die Formatierung in einem zusätzlich angeschlossenem Zweitlaufwerk. Dadurch sehen Sie auch, daß auch bei der Dateiverwaltung RANDOMD1 mit zwei Laufwerken gearbeitet werden kann. Es muß nur im Basiclisting RANDOMD1 ebenfalls geändert werden.

Auch beim Programm RANDOMD2 sollen die Füllzeichen bereits bei der Formatierung geschrieben werden. Hierzu dient VORFORMX.BAS.

Wenn Sie beide Programmlängen vergleichen, werden Sie bemerken, daß das nun folgende wesentlich kürzer ist. Dies hat zwei Gründe. Erstens kann ein Formatier-Aufruf bei den X-Modul-Laufwerken durch einen RSX-Befehl erfolgen und zum anderen wird kein Sektor mit einem Inhalt bedacht, wie dies bei SPEZFORM.BAS der Fall ist.

8.4.2 Basic-Listing:VORFORMX.BAS

```
100 'VORFORMX.BAS - Programm zum Spezial-"Formatieren" fuer Vortex-X-Modul-LW
110 '
120 MODE 2:PRINT:PRINT"Programm zur Spezialformatierung fuer VORTEX-X-Laufwerke
130 '
140 POKE &ABD8,&2E:|FORMAT,2:POKE &ABD8,&E5:|B:CAT:|A
150 '
160 PRINT"Diskette liegt nun in Spezialformat vor!"
```

Zur Bedienung des letzteren Programmes sei hier nur erwähnt, daß unter VDOS gearbeitet wird und das eingebaute Schneider-Laufwerk das Laufwerk 1 ist. CPC 464-Besitzer ohne Schneider-Laufwerk müssen ggf. noch Anpassungen machen. In der vorliegenden Form arbeitet es sicher mit dem CPC 6128.

8.5 Schlußwort

Wenn Sie dieses Buch durchgearbeitet haben, werden Sie nicht nur einiges über Dateien und deren Verwaltung gelernt haben, sondern auch den CPC in vielen Dingen besser programmieren können.

Es konnte und kann von mir nicht auf alle Fragen Antwort gegeben werden. Versuchen Sie anhand der vielen Beispielprogramme dann einen Weg zu finden, der auch Ihr Problem löst.

Mit den "superschnellen" Maschinenroutinen, die abgedruckt sind, werden sie bestimmt Verwaltungen aufbauen, die einem professionellen Programm in nichts nachstehen was die Geschwindigkeit und die Kapazität bei der Datenverwaltung betrifft.

Bei den Hilfs- und Zusatzprogrammen können auch noch viele Minuten gespart werden, wenn die Teile, die vor allem zur Demonstration dienen und Bildschirm-Ausgaben veranlassen, gestrichen werden. Man muß nicht immer alles sehen, aber daß der CPC etwas tut, das sollte immer angezeigt werden!

Ich wünsche nochmals viel Erfolg bei der Verwaltung von Daten mit Ihrem CPC.

Bezugsquelle:

Die Diskette zu diesem Buch ist erhältlich bei:

Franzis-Software-Service
Postfach 37Ø12Ø
8ØØØ München 37
Tel.: Ø89-5117-331

In diesem Buch werden alle Fragen einer Dateiverwaltung mit dem CPC behandelt. Das Spektrum reicht vom einfachen Wörterbuch über eine komfortable Verwendung von Datei-Masken bis zur schnellen Ramdisk. Dabei kommen Basic und Assembler als Programmiersprachen zum Einsatz.

Zu allen Bereichen werden umfangreiche und leistungsfähige Programme angeboten. Es kam dem Autor bei seinen Darstellungen vor allem darauf an, daß der Leser auch mit schwierigen Programmen umzugehen lernt. Das Buch ist also eine echte, sehr aktuelle Trickkiste für den CPC.

Der Autor ist ein erfahrener CPC-Spezialist. Wer das Buch gelesen hat, kann erfolgreich an die Umsetzung seiner eigenen Ideen herangehen. Dabei werden individuelle Lösungen entstehen, die sich vor professionellen nicht zu verstecken brauchen.

03800
9 783772 386916
ISB N 3-7723-8691-1 DM +038.00

Franzis'

Franzis' Miedel **Schneider CPC: Dateiverwaltung**

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/