Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 77

Ein wöchentliches Sammelwerk

MIDI für den Schneider

Unix-Directories

Der Minicomputer VAX

Das Micro-Marketing

computer kurs

Heft 77

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wöchentlich erscheinendes Sammelwerk. Die Gesamtzahl der Hefte ergibt ein vollständiges Computer-Nachschlagewerk. Damit Sie jede Woche Ihr Heft erhalten, bitten Sie Ihren Zeitschriftenhändler, Computer Kurs für Sie zu reservieren.

Zurückliegende Hefte
Ihr Zeitschriftenhändler besorgt Ihnen gerne zurückliegende Hefte. Sie können sie aber auch direkt beim Verlag bestellen.

Deutschland: Das einzelne Heft kostet DM 3,80. Bitte füllen Sie eine Postzahlkarte aus an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Computer Kurs

Österreich: Das einzelne Heft kostet öS 30. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs, Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Computer Kurs.

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen.

WICHTIG: Bei Ihren Bestellungen muß der linke Abschnitt der Zahlkarte Ihre vollständige Adresse enthalten, damit Sie die Hefte schnell und sicher erhalten. Überweisen Sie durch Ihre Bank, so muß die Überweisungskopie Ihre vollständige Anschrift gut leserlich enthalten.

SAMMELORDNER
Sie können die Sammelordner entweder direkt bei Ihrem Zeitschriftenhändler kaufen (falls nicht vorrätig, bestellt er sie gerne für Sie) oder aber Sie bestellen die Sammelordner für den gleichen Preis beim Verlag wie folgt:

Deutschland: Der Sammelordner kostet DM 12. Bitte füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

Österreich: Der Sammelordner kostet öS 98. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Sammelordner Computer Kurs.

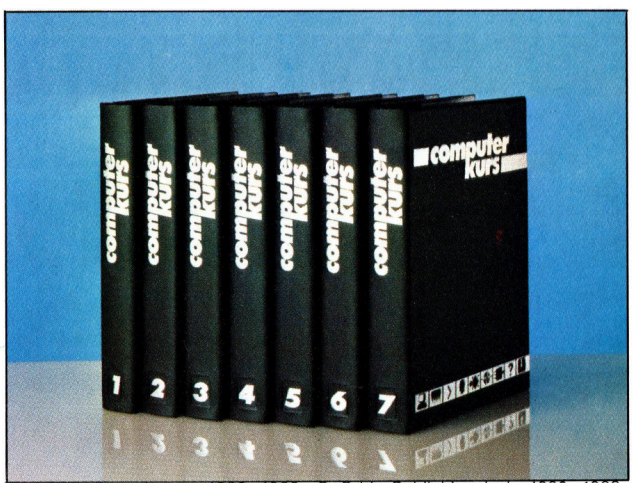
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Holger Neuhaus, Peter Aldick, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1



Druck: E. Schwend GmbH, Schmollerstraße 31, 7170 Schwäbisch Hall

Wer im Computerbereich eine Stellung sucht und keine Spezialausbildung mitbringt, hat beim kleineren Hardware-Einzelhändler noch die besten Aussichten auf einen Job. Das Problem einer solchen Beschäftigung liegt oft in der Überlastung mit täglichem Kleinkram; der Angestellte hat kaum Zeit für Fortbildung und Information über neue Produkte.

Im Vertrieb

Der komplizierte und unübersichtliche Computermarkt hat durch ständiges Wachstum eine Vielzahl von Stellen im Marketing- und Vertriebsbereich geschaffen. Beide Berufsfelder bieten interessante Aufstiegsmöglichkeiten und hohe Gehälter.

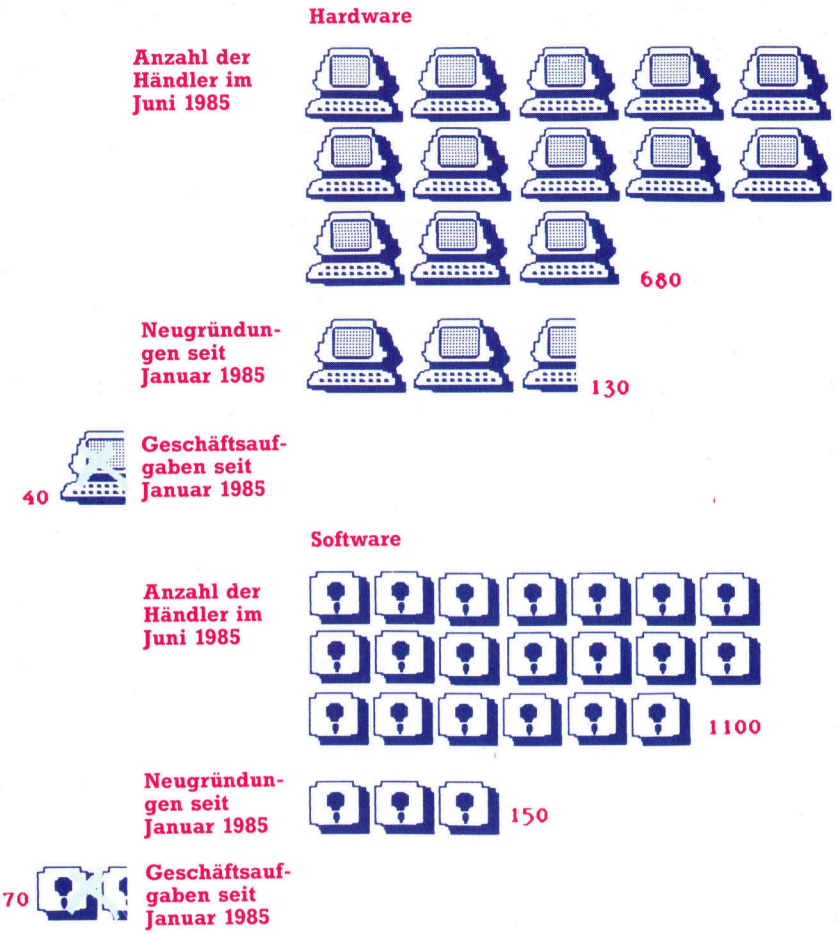
Die ungewöhnlichen Wachstumsraten im Computergeschäft beginnen sich langsam zu normalisieren, doch schrumpft damit auch der Zuwachs an offenen Stellen. Zwar werden immer noch dreimal soviele Unternehmen neu gegründet, wie sich (zuletzt und spektakulär Sinclair) aus dem Geschäft zurückziehen. Aber auch der beste Programmierer oder Vertriebsmann ist heute vor einer überraschenden Firmenschließung nicht sicher.

Arbeitsstellen im Vertriebs- oder Marketingbereich gehören zu den bestbezahlten Positionen dieser Branche. An der Spitze steht dabei der Vertriebsprofi, der mit kommerziellen Firmen über die Anschaffung größerer Anlagen oder umfangreicher Software-Systeme verhandelt. Er kann im Jahr um hunderttausend Mark verdienen – zumeist als Provision. Marketing-Angestellte, die keine Provision erhalten, verdienen weniger. Erst nach mehrjähriger Tätigkeit kann man den Sessel eines Marketing-Direktors erkämpfen, bei dem das Salär üblicherweise die 100 000-Markgrenze überschreitet. Vertrieb und Marketing sind häufig der Einstieg karrierebewußter Aufsteiger gewesen, die sich von hier bis an die Spitze bedeutender Unternehmen vorarbeiten konnten. Bis vor kurzem stammten beispielsweise noch sämtliche IBM-Manager aus dem Vertriebsbereich des Unternehmen.

In den meisten Laufbahnen ist es die schwierigste Aufgabe, die erste Stufe der Karriereleiter zu erklimmen; der Vertrieb macht hier keine Ausnahme. Üblicherweise erwarten die Firmen eine gewisse Berufserfahrung. Wie aber soll der Neuling sie erwerben, wenn ihm nirgends eine Chance geboten wird?

Personalberatungsfirmen empfehlen den Bewerbern um Vertriebsjobs meist den Besuch von Computer-Einführungskursen, die teilweise auch von den Arbeitsämtern gefördert werden. Der nächste Schritt ist eine Anstellung bei einem Unternehmen, das etwa Büroausstattung oder Computer verkauft. Hier wird sel-

Zuwächse beim Handel



Trotz eher pessimistischer Vorhersagen mancher Fachleute hat sich die Wachstumsrate bei den Computer-Händlern bis heute nicht verlangsamt. Es ist eher das stabile Wachstum als die Aussicht auf extreme Profite, die junge Unternehmen dazu veranlassen, sich auf diesem Markt zu etablieren. (Zahlen aus England laut NCC Microsystems Centre).

Vertrieb und Marketing

Marketing und Vertrieb werden häufig verwechselt: Der Vertriebsmann ist für den direkten Kontakt zum Kunden zuständig, während Marketing eher eine strategische Aufgabe ist: Die Verpackung des Produktes muß geplant werden, Namen müssen gefunden werden, und auch die Werbung bedarf sorgfältiger Planung.

In der Microcomputer-Industrie gibt es zwei Hauptbranchen – Hersteller und Wiederverkäufer. Vertriebsleute von Hard- und Softwareherstellern verkaufen hauptsächlich an Händler oder Handelsketten. Das Vertriebspersonal von Wiederverkäufern (Systemhäuser, Einzelhandel) wendet sich direkt an den Endabnehmer. Die Verkaufsgewohnheiten sind national sehr verschieden – während in den USA der schnelle Verkauf über den Ladentisch die Regel ist, werden in Europa meist noch Vorverhandlungen vereinbart. Oft geht einem Verkaufsabschluß auch eine Mailing-Aktion oder ein telefonisches Verkaufsgespräch voraus.

Zu Beginn der Computer-Ära wurde das Marketing noch recht stiefmütterlich behandelt: Es wurde planlos gefertigt und verkauft. Heute hat man dazugelernt. Der erfolgreichste Heimcomputer-Hersteller war 1985 die Firma Amstrad (in der BRD Schneider). An der Technologie gab es nichts Neues zu entdecken, das Design der Geräte entsprach jedoch den Wünschen der Abnehmer. Grundsätzlich ist das Erfassen vorhandener Wünsche eine der Hauptaufgaben des Marketing.

ten viel verdient, es können aber wichtige Erfahrungen gesammelt werden. Danach befindet man sich in einer besseren Position, wenn es an die Verhandlungen mit einem professionellen Computer-Handelsunternehmen oder gar einem Hersteller geht. Aber auch bei einer erfolgreichen Bewerbung bleibt die Bezahlung im ersten Jahr niedrig. Der Neuling wird als Vertriebsassistent einem erfahreneren Verkäufer zugeordnet und muß zuerst einmal lernen, wie Computer oder Software-Pakete vorgeführt werden.

Der Weg nach oben

Wer sich bewährt, hat gute Aufstiegsmöglichkeiten. Einjährige Erfahrung bei einer der größeren Firmen genügt oft schon für den Einstieg als „richtiger" Vertriebsmann, der bereits 30–40 000 Mark verdienen und diese durch Provisionen theoretisch noch verdoppeln kann. Im allgemeinen ist auch noch ein Firmenwagen für Außendienste drin.

Innerhalb von drei bis vier Jahren kann aus dem Anfänger ein Profi werden, dem auch wichtige Kunden anvertraut werden. Ein Unternehmen, das sowohl Ein- wie auch Mehrplatzsysteme verkauft, überläßt die teureren Anlagen natürlich den erfahrenen Mitarbeitern, und damit haben diese auch höhere Provisionseinnahmen. Der Verdienst kann jetzt bereits zwischen 50 000 und fast 100 000 Mark liegen, wenn der Erfolg beim Verkauf gut ist. Nach dem Erreichen des 30. Lebensjahres sollte versucht werden, auf den Sessel eines Vertriebsmanagers zu kommen. Die Arbeit bekommt organisatorischen Charakter: So müssen etwa die Vertriebsziele für die Mannschaft festgelegt werden. Wegen der Nähe zum Marketing-Bereich wird dieser Zeitpunkt oft gewählt, um in dieses Tätigkeitsfeld überzuwechseln.

Man darf nie vergessen, daß der Vertrieb ein unsicheres Gewerbe ist, speziell in der risikoreichen Computer-Branche. Nicht selten sind die Vertriebsziele kaum zu erfüllen. Wer mehrere Quartale unter dem Soll bleibt, findet sich leicht auf der Straße wieder. Für Sicherheitsfanatiker ist diese Arbeit ganz bestimmt nicht das Richtige!

Auch beim Marketing ist die Konkurrenz groß. Zwar geben sich manche Firmen mit einem Fachhochschul-Diplom zufrieden, überwiegend wird jedoch ein Universitätsabschluß gefordert. Besonderer Wert wird auf gute Noten in naturwissenschaftlichen Fächern wie

Kettenreaktion

Das britische Unternehmen Granada Business Centres in England ist eine stark expandierende Kette von Computer-Händlern, die sich auf den Verkauf von Ein- und Mehrplatzrechnern an kleine und mittelgroße Unternehmen spezialisiert hat. Als Grundvoraussetzung einer Neueinstellung nennt man bei Granada Verkaufserfahrung im gewerblichen Bereich, noch lieber wird Erfahrung im Verkauf von Computersystemen gesehen. Granada bildet seine zwischen dem 23. und dem 30. Lebensjahr eingestellten Vertriebsleute selbst aus. In jedem Geschäft gibt es einen Vertriebsbeauftragten für Großkunden und einen Vertriebsleiter, der die Aktivitäten des betreffenden Geschäftes überwacht. Ein Bezirksleiter ist für jeweils vier Geschäfte und die dazugehörige Servicemannschaft zuständig.

Die Marketingabteilung bei Granada ist klein. Auf die 60 Vertriebsleute kommen nur vier Marketingmanager, von denen jeder einen Bereich bearbeitet: Der eine ist für Ausstellungen zuständig, der nächste für Pressearbeit, der dritte für die Gestaltung der Läden und so weiter.

Das kleine Wunder

Psion ist ein englisches Unternehmen, das sich mit Software-Entwicklung und dem Bau von Handheld-Computern befaßt. Die Firma hat 70 Angestellte. Von Neulingen im Vertriebsbereich werden zwar Erfahrungen im Verkauf von technischen Geräten oder aus dem Bürofachhandel erwartet, spezielles Computerwissen setzt man jedoch nicht voraus. Psion schult seine Angestellten an den eigenen Produkten. Der Vertriebsmann arbeitet direkt mit Psion-Händlern und größeren Einkaufsgemeinschaften zusammen.

Ein Vertriebsbeauftragter ist bereits für einen festen Kundenstamm zuständig und überwacht die Arbeit der beigeordneten Assistenten.

Bei Psion werden nur Bewerber mit Hochschulabschluß im Marketing beschäftigt. Sie erstellen die Entwürfe für Werbung und Handbücher, organisieren Messestände und bearbeiten Anfragen der Presse.

Marketingleiter und Direktor sind für langfristige Planungen zuständig, sie instruieren etwa die Werbeagentur und lassen Marktforschung durchführen.

Kleine Unternehmen können oft flexibler sein. Sie bieten auch Chancen für solche Interessenten, die sich über ihren Berufsweg noch nicht klar sind. So stellte Psion 1983 etwa einen Programmierer ein, der später zur technischen Vertriebsberatung überwechselte und kurz darauf Vertriebsbeauftragter wurde. Ende 1985 war er dann in der Exportabteilung gelandet.

Marketing beinhaltet erheblich mehr als nur den Verkauf eines Produktes. Wer im Marketing tätig ist, begleitet eine Neuentwicklung vom Konzeptions- und Entwicklungsstadium an. Marketing-Fachleute müssen aufmerksam darüber wachen, daß neue Geräte nicht an den Bedürfnissen des Endabnehmers vorbeigeplant werden. Ein Beispiel für gutes Marketing bietet die Firma Schneider. Obwohl jeder ihrer Computer mit im Grunde bereits überholter Technik arbeitet, kann sie doch den Wunsch des Verbrauchers nach preisgünstigem Computereinsatz voll abdecken.

Mathematik oder Physik gelegt.

Eine Besonderheit des Marketing-Bereiches ist es, daß die Arbeitsprinzipien völlig unabhängig vom Produkt sind: Wer im Marketing von Waschpulver Erfolge vorweisen kann, hat auch in der Computer-Branche Chancen. Viele kleinere Computer-Firmen verfügen nicht über die Mittel, ihre Marketing-Spezialisten selbst auszubilden, zum Vorteil für denjenigen, der bereits zwei bis drei Jahre unter der Obhut eines Großunternehmens zugebracht hat.

Überzeugungsarbeit

Computer-Kenntnisse sind im Marketing kein Muß – aber natürlich ist Interesse an der Sache kein Fehler. Ein Marketing-Assistent mit ein- bis zweijähriger Berufserfahrung verdient ca. 40 000 Mark jährlich. Seine Aufgabe besteht darin, beispielsweise Handbücher und Prospekte zu texten, die Händler mit Informationen zu versorgen, Seminare und Pressekonferenzen abzuhalten und auch Kontakte zu Journalisten zu pflegen.

Der nächste Schritt führt zur Position des Marketing-Leiters. Er hält den Kontakt zur Werbeagentur, entscheidet über Messeaktivitäten und hat auch bei der Einführung neuer Produkte und der Planung einer Firmenstrategie bedeutenden Einfluß. Das Einkommen eines Marketingleiters überschreitet meist die 60 000 Mark, dazu wird er im allgemeinen auch mit einem Firmenwagen und anderen Extras ausgestattet. In kleinen Firmen bleibt dann nur noch der Aufstieg zum Marketing-Direktor.

Blätterrauschen

In unserem neuen Projekt entwerfen wir ein Tabellenkalkulationsprogramm für vier Microcomputer und stellen die wichtigsten Leistungsmerkmale eines „Spreadsheets“ vor.

Ein Tabellenkalkulationsprogramm ist das ideale Hilfsmittel, wenn große Datenmengen zu berechnen sind. Mit ihm können Sie Reihen und Spalten auf Knopfdruck addieren – eine hervorragende Eigenschaft für Verkaufsabrechnungen, für die Verarbeitung wechselnder Daten mit den gleichen mathematischen Formeln, zur Berechnung der Mehrwert- und Einkommenssteuer und ähnlicher Rechenvorgänge. Kurz, Sie können mit einem Kalkulationsprogramm alle Arbeiten ausführen, die Sie mit einem Stift, einem Blatt Papier und einem Rechner erledigen, nur bedeutend schneller und genauer. Ein weiterer Pluspunkt guter Kalkulationsprogramme ist die Möglichkeit, einmal eingegebene Formeln und Anweisungen immer wieder neu, mit anderen Daten, zu benutzen. Für ständig wiederkehrende Berechnungen, wie etwa wöchentliche Reise- oder Verkaufsabrechnungen, ist das überaus hilfreich. Beim Durchführen kleinerer Geschäfte können Sie etwa die bereits beim Wareneinkauf erstattete Mehrwertsteuer gegen die an die Finanzbehörde abzuführende gegenrechnen. Zuvor müssen Sie natürlich die Gesamtsumme der beim Verkauf berechneten Steuer errechnen und die beim Einkauf bezahlte davon subtrahieren. Mit dem Taschenrechner müßten Sie die Steueranteile für jeden Posten gesondert ermitteln und die Einzelergebnisse später summieren – eine Vorgehensweise, die gleichermaßen ermüdend wie anfällig für Fehler ist. Einem Kalkulationsprogramm geben Sie nur die Rechenoperationen zum Ermitteln der Mehrwertsteuer vor: in diesem Falle die Eintragungen einer Spalte durch 100 dividieren und mit 14 multiplizieren. Anschließend geben Sie die einzelnen Beträge ein, den Rest erledigt das Programm.

Die Anwendungsmöglichkeiten sind zahlreich und erstrecken sich im privaten Bereich von Haushalts- und Kraftfahrzeugkosten bis zu Lohnabrechnungen. Geschäftliche Anwendungen sind Faktura, Buchführung, Finanzierung, Vorhersagen und Planung, Kostenanalysen und Projektüberwachung, um einige zu nennen. Die enormen Vorteile für den Benutzer führen zu gesteigerter Produktivität, verminderter Fehlerquote und besserer Präsentation.

Das in dieser Serie erstellte Spreadsheet gibt dem BASIC-Programmierer einen Einblick in den Aufbau und die Funktion kommerzieller Tabellenkalkulationsprogramme. Zu jedem Teil des Programms geben wir detaillierte Erläuterungen über die Funktion und Arbeitsweise der jeweiligen Sektion. Das Programm ist für Computer der Typen Sinclair Spectrum, Commodore 64, Acorn B und Schneider CPC geschrieben. Jeden Programmteil drucken wir in einer Spectrum- und einer Microsoft-BASIC-Version mit Erläuterungen zur Übertragung auf andere Rechnersysteme.

Konzentration

Angesichts des komplexen Aufbaus eines Kalkulationsprogramms konzentrieren wir uns auf die wesentlichen Leistungsmerkmale. Resultierend daraus findet lediglich eine minimale Fehlerüberprüfung statt. Als Konsequenz daraus sollten Sie größtmögliche Sorgfalt bei der Benutzung des Programms, speziell beim Eingeben der Informationen, walten lassen, da jeder Fehler zum Zusammenbruch des Programms führt. Für die vernachlässigte Fehlerbehandlung stehen zwei Gründe: erstens der beschränkte Speicherplatz und zweitens die Arbeitsgeschwindigkeit.

Da das Programm in BASIC geschrieben wurde, ist die Größe des Arbeitsplatzes limitiert auf eine Matrix von 15 Spalten und 15 Reihen. Diese Einschränkung garantiert eine annehmbare Ablaufgeschwindigkeit und läßt dennoch ein übersichtliches Programmlisting zu. Aufgrund des Bildschirmaufbaus kann immer nur ein Teil („Fenster“) des Arbeitsblattes gezeigt werden. Auf dem C 64, dem Acorn B und Schneider-Computern umfassen diese Fenster fünf Reihen und sieben Spalten, auf dem Spectrum vier Reihen und sechs Spalten.

Jedes der 225 Felder des Kalkulationsprogramms trägt die Bezeichnung Zelle. Diese Zellen sind benannt (in der obersten Reihe) als A1, A2 usw. bis A15, B1 bis B15 (in der zweiten Reihe), bis zur fünfzehnten Spalte (O1, O2 und weiter bis O15). Zur Bewegung des Cursors innerhalb des Kalkulationsschemas benutzen Sie einfach die Cursor-Steuerungstasten. Der Name der angewählten Zelle ist auf der oberen linken Seite des Bildschirmes sichtbar. Die Zelle selbst ist invertiert.

Jede Zelle kann eine Information beinhalten: entweder beliebige numerische Daten oder eine Berechnungsformel. Die Daten erscheinen auf dem Bildschirm in der jeweils zugehörenden Zelle, wobei pro Zelle nur fünf Zeichen

zur Verfügung stehen. Enthält die angewählte Zelle eine Berechnungsformel, so wird diese in der Eingabezeile am unteren Rand des Bildschirms angezeigt. Zum Eingeben einer neuen Formel wählen Sie die Option „enter formula" und setzen die neue Formel ein. Die Berechnungsroutinen bearbeiten die Felder von links nach rechts. Als Operatoren verarbeitet das Programm +, −, *, /, ^ und die runden Klammern. Es erkennt Zellnamen und wirkliche oder ganze Zahlen als Operanden. Wenn Sie die Werte in die Zellen A1, A2 und A3 addieren und das Ergebnis in Zelle A4 ablegen möchten, schreiben Sie die Formel A1+A2+A3 in die Zelle A4. Zum Errechnen der Mehrwertsteuer, zum Beispiel des Wertes in B1, schreiben Sie B1*0.14 in Zelle B2 und die Formel B1+B2 in die Zelle B3.

Unser BASIC-Kalkulationsprogramm bietet viele Funktionen. Dazu gehören:

- Eine Hilfstafel wählen Sie über HELP an. Sie listet die für den Anwender verfügbaren Optionen auf.
- Mit der vorläufigen Speicherfunktion STORE sichern Sie Ihr Arbeitsblatt im derzeitigen Stadium auf Datenträger. Damit können Sie in das Arbeitsblatt unterschiedliche Werte eintragen.
- Ein vorübergehend gesichertes Arbeitsblatt können Sie mit RESTORE jederzeit wieder zurück in den Computer laden und damit den ursprünglichen Zustand wiederherstellen.
- Die Rechenfunktion CALCULATE bearbeitet (von links nach rechts und von oben nach unten) jeden Eintrag der Reihe nach ab. Diese Reihenfolge müssen Sie beim Eingeben berücksichtigen, da sonst fehlerhafte Ergebnisse auftreten könnten.
- Mit der CLEAR-Anweisung löschen Sie den gesamten Inhalt des Arbeitsblattes. – Vergewissern Sie sich vorher, ob Sie die Daten auf Band oder Diskette sicherten.
- Mit der REPLICATE-Funktion läßt sich eine Formel auf andere Zellen übertragen. Das ist besonders dann nützlich, wenn die entsprechende Formel oft im Arbeitsblatt erscheinen soll (z. B. bei Berechnungen der MWSt.).
- Ihre Daten sichern und laden Sie mittels der SAVE- und LOAD-Funktionen.
- Mit SAVE und LOAD können Sie auch nur die Berechnungsformeln sichern und laden. Das ist überaus nützlich für ständig benötigte Kalkulationsschemata (etwa wöchentliche Reisekostenabrechnungen).
- Der Cursor kann mittels der TAB-Funktion an jede gewünschte Position direkt gesetzt werden, wobei Sie den Weg nicht mit den Steuertasten vorgeben müssen.

In diesem Beitrag drucken wir die Routinen zum Bildschirmaufbau und zur Steuerung der Grafik für den Commodore 64. Die entsprechenden Programmteile für den Acorn B, den Spectrum und den Schneider CPC folgen in der nächsten Ausgabe. Nur diese Programmsektionen wurden speziell für die jeweiligen Computer geschrieben, da die Ansteuerung der Bildschirmgrafik unterschiedlich ist. Die restlichen Programmteile sind für alle Computer gleich (mit Ausnahme des Spectrum, dessen Stringhandling sich von den übrigen unterscheidet).

Steuerberechnung

			A2*14/114			A5*14/100
			B3*14/114			B5*14/100
			C3*14/114			C5*14/100
			D2*14/114			D5*14/100

	1. Einkauf	2. Preis	3. MwSt	4. Verkauf	5. Preis	6. MwSt
A	Micro	999.98	122.80	Kalk-Pgr	299.98	42.00
B	Drucker	624.98	76.75	Conv-Pgr	4000,00	560.00
C	Papier	50.00	6.14	Kalk-Pgr	299.98	42.00
D	Farbband	18.10	2.22	Druckutil	160.00	22.40
E	Stifte usw.	6.05	0.74			
F	Tisch	231.18	28.39			
G	Laufwerk	499.88	61.39			
H	Leuchte	62.18	7.64			
I	Rechenstab	143.30	17.60			
J	Sicherungen	2.50	0.31			
K	Porto	20.00	2.46			
L		MwSt aus	326.44			
M					MwSt ein	666.39
N					MwSt ges.	339.95

Dieses Beispiel zeigt eine typische Anwendung aus dem Bereich der Buchhaltung – die Berechnung der Mehrwertsteuer. Die Spalten 1, 2, 4 und 5 müssen vom Benutzer mit den entsprechenden Angaben ausgefüllt werden. Das Programm errechnet die Einzel- und Gesamtsummen in den Spalten 3 und 6. Die Summen in Spalte 3 ergeben sich aus dem jeweiligen Wert in Spalte 2 multipliziert mit 14 und dividiert durch 100. Die dazugehörige Programmformel sehen Sie über dem Arbeitsblatt. Oftmals wird eine Berechnungsformel in mehreren Zellen benötigt. Damit nun der Anwender nicht jede Zelle ausfüllen muß, bieten die meisten Kalkulationsprogramme eine REPLICATE-Funktion, mit der eine Formel in andere Zellen übertragen wird.

Grafik-routinen

Auch wenn der größte Anteil des Programmlistings für alle Computer gültig ist, benötigen die hier besprochenen Rechner unterschiedliche Befehle zum Aufbau der Bildschirme. Um dennoch ansprechende und übersichtliche Darstellungen zu erreichen, bringen wir zu jedem der Computer separate Routinen. Wir beginnen in dieser Ausgabe mit den Routinen für den Commodore 64. Die Programmsektionen steuern die Darstellung der Tabellenmatrix, der Daten in den Zellen und die Bewegung des Cursors. Nach dem Eingeben und Starten dieses Programmteils sollten Sie die Matrix auf dem Bildschirm sehen. Sie können den Cursor über das Arbeitsblatt wandern lassen, doch die eigentlichen Kalkulationsfunktionen sind noch nicht vorhanden. Sie sind Bestandteil der folgenden Listings.

Commodore 64

```
10 REM **** CBM 64 SPREADSHEET ****
15 REM ** SET UP CBM CHARACTER STRINGS *
*
20 FOR I=1 TO 7:L7$=L7$+CHR$(195):NEXT
30 L5$=LEFT$(L7$,5):T$=CHR$(178):B$=CHR$
(194)
40 S5$="      ":X$=CHR$(219):I$=CHR$(177)

100 GOSUB 3000:REM SETUP ARRAYS & VARIAB
LES
110 GOSUB 1000:REM PRINT SCREEN
120 GOSUB 1700:REM PRINT DATA ON SCREEN
130 GOSUB 1100:REM MAIN KEYBOARD SCANNIN
G ROUTINE
999 STOP
1000 PRINT CHR$(147);CHR$(145);CHR$(5):P
OKE 53280,6:POKE 53281,6
1005 PRINT "                C O L U M N S"

1006 PRINT
1007 PRINT "ROW     1.    2.    3.    4.
    5.    "
1010 PRINT "         "CHR$(176);L5$;T$;L5
$;T$;L5$;T$;L5$;T$;L5$;CHR$(174)
1020 FOR C=1 TO 7
1030 PRINT " "CHR$(C+64);".      "B$;S5$;
B$;S5$;B$;S5$;B$;S5$;B$;S5$;B$
1040 PRINT " ";L7$;X$;L5$;X$;L5$;X$;L5$;
X$;L5$;X$;L5$;CHR$(179)
1050 NEXT C
1060 PRINT " "CHR$(C+64);".      ";B$;S5$
;B$;S5$;B$;S5$;B$;S5$;B$;S5$;B$
1070 PRINT " ";L7$;I$;L5$;I$;L5$;I$;L5$;
I$;L5$;I$;L5$;CHR$(189)
1080 RETURN
1100 P$=CHR$(Y+64)+MID$(STR$(X),2,2):PRI
NT CHR$(19);"CELL:";P$;" "
1110 GET A$:IF A$="" THEN 1110
1120 IF A$=CHR$(29) THEN 1200:REM MOVE R
IGHT
1130 IF A$=CHR$(157) THEN 1300:REM MOVE
LEFT
1140 IF A$=CHR$(17) THEN 1400:REM MOVE D
OWN
1150 IF A$=CHR$(145) THEN 1500:REM MOVE
UP
1152 IF A$=CHR$(133) THEN GOSUB 6000:REM
 F1 PRINT HELP SCREEN
1155 IF A$=CHR$(137) THEN GOSUB 2000:REM
 F2 INPUT FORMULA
1158 IF A$=CHR$(134) THEN GOSUB 5150:REM
 F3 STORE CURRENT SHEET
1160 IF A$=CHR$(135) THEN GOSUB 2300:REM
 F5 CALCULATE SHEET
1165 IF A$=CHR$(13) THEN RETURN
1170 IF A$)"0" AND A$("9" THEN GOSUB 210
0:REM ENTER NUMERIC DATA
1180 IF A$=CHR$(139) THEN GOSUB 5000:REM
 F6 CLEAR SHEET
1185 IF A$=CHR$(138) THEN GOSUB 5100:REM
 F4 GET PREVIOUS SHEET
1187 IF A$="G" THEN GOSUB 5200
1188 IF A$=CHR$(136) THEN GOSUB 5700:REM
 F7 DECIDE COL OR ROW
1189 IF A$=CHR$(140) THEN GOSUB 7000:REM
 F8 LOAD/SAVE ROUTINES
1190 GOTO 1110:REM BACK TO START
1200 REM ****** MOVE RIGHT ******
1210 IF X=15 THEN 1100
1220 IF X=H2 THEN GOSUB 1600:X=X+1:H1=H1
+1:H2=H2+1:GOTO 1270
1230 GOSUB 1600:LET X=X+1:GOSUB 1650:GOT
O 1100
1270 GOSUB 1800:GOSUB 1700:GOTO 1100
1300 REM ****** MOVE LEFT ******
1310 IF X=1 THEN 1100
1320 IF X=H1 THEN GOSUB 1600:X=X-1:H1=H1
-1:H2=H2-1:GOTO 1370
1330 GOSUB 1600:LET X=X-1:GOSUB 1650:GOT
O 1100
1370 GOSUB 1800:GOSUB 1700:GOTO 1100
1400 REM ****** MOVE DOWN ******
1410 IF Y=15 THEN 1100
1420 IF Y=V2 THEN GOSUB 1600:Y=Y+1:V1=V1
+1:V2=V2+1:GOTO 1470
1430 GOSUB 1600:LET Y=Y+1:GOSUB 1650:GOT
O 1100
1470 GOSUB 1850:GOSUB 1700:GOTO 1100
1500 REM ****** MOVE UP ******
1510 IF Y=1 THEN 1100
1520 IF Y=V1 THEN GOSUB 1600:Y=Y-1:V1=V1
-1:V2=V2-1:GOTO 1570
1530 GOSUB 1600:LET Y=Y-1:GOSUB 1650:GOT
O 1100
1570 GOSUB 1850:GOSUB 1700:GOTO 1100
1600 REM ***** TURN CURSOR OFF *****
1610 CU=1023+40*(V(Y+1-V1)-1)+H(X+1-H1):
POKE CU,PEEK(CU)-128
1620 POKE CU+1,PEEK(CU+1)-128:POKE CU+2,
PEEK(CU+2)-128
1630 POKE CU+3,PEEK(CU+3)-128:POKE CU+4,
PEEK(CU+4)-128:RETURN
1650 REM ***** TURN CURSOR ON ******
1660 CU=1023+40*(V(Y+1-V1)-1)+H(X+1-H1):
POKE CU,PEEK(CU)+128
1670 POKE CU+1,PEEK(CU+1)+128:POKE CU+2,
PEEK(CU+2)+128
1680 POKE CU+3,PEEK(CU+3)+128:POKE CU+4,
PEEK(CU+4)+128
1690 GOSUB 1900:RETURN
1700 REM **** PRINT DATA IN SHEET ****
1710 FOR I=0 TO 7
1720 PRINT CHR$(19);:FOR C=1 TO V(I+1)-1
:PRINT CHR$(17);:NEXT C
1730 FOR J=0 TO 4
1735 P$=MID$(STR$(MAT(I+V1,J+H1)),2)
1740 PRINT TAB(H(J+1)-1);"      ";CHR$(14
5)
1745 PRINT TAB(H(J+1)+4-LEN(P$));P$;
1750 NEXT J,I
1760 GOSUB 1650:RETURN
1800 REM **** PRINT COLUMN NUMBER ***
1810 PRINT CHR$(19);CHR$(17);CHR$(17);CH
R$(17);
1820 FOR I=H1 TO H2:PRINT TAB(7+6*(I-H1)
)I;CHR$(157);". ";
1830 NEXT I:RETURN
1850 REM **** PRINT ROW NUMBERS *****
1860 PRINT CHR$(19);:FOR I=1 TO 5:PRINT
CHR$(17);:NEXT
1870 FOR C=V1 TO V2
1880 PRINT TAB(1)CHR$(C+64);CHR$(17)
1890 NEXT C:RETURN
1900 REM *** FORMULA OF CURRENT CELL **
1920 LET D$=F$((Y-1)*15+X)
1930 GOSUB 1950:REM MOVE CURSOR TO INPUT
 LINE
1935 PRINT CHR$(18);"FORMULA:
                "
1940 PRINT CHR$(145);CHR$(18)"FORMULA:";
D$
1945 PRINT CHR$(19):RETURN
1950 REM ***** CURSOR TO INPUT LINE ***
1960 PRINT CHR$(19);:FOR K=1 TO 22:PRINT
 CHR$(17);:NEXT K
1970 RETURN



3000 REM ******   SETUP ARRAYS *******
3010 DIM H(5),V(8),ST(20),ST$(20),E$(20)
,G$(20)
3020 FOR C=0 TO 4
3030 H(C+1)=6*C+10:REM CALC X POS
3040 NEXT C
3050 FOR C=1 TO 8
3060 V(C)=2*C+4:REM CALC Y-POS
3070 NEXT C
3075 X=1:Y=1
3080 H1=X:H2=X+4:V1=Y:V2=Y+7
3090 REM *******  SHEET ARRAYS ********
3100 DIM MAT(15,15):DIM MC(15,15)
3110 FOR I=1 TO 15:FOR J=1 TO 15
3120 MAT(I,J)=I*J
3130 NEXT J,I
3140 DIM F$(225)
3150 RETURN
```

MIDI: Musik liegt in der Luft

Das in einem früheren Kursabschnitt entstandene MIDI-Interface war für den Anschluß am Commodore 64 und Acorn B vorgesehen. Wir wollen die Schnittstelle nun an einem Rechner der Schneider-CPC-Serie betreiben.

Die Hardware der Schnittstelle unterscheidet sich auch in der Schneider-Version kaum von der für die beiden anderen Rechner verwendeten Schaltung. Unterschiede ergeben sich daraus, daß der Schneider nicht mit einer CPU der 65XX-Serie, sondern mit dem Microprozessor Z80 ausgestattet ist.

Um das IC mit dem Z80 zu koppeln, ist ein Zusatzbauteil erforderlich: ein Dreifach-NAND-Gatter mit je drei Eingängen. Die beiden Chip-Select-Leitungen CS0 und CS1 waren in der Original-Version auf +5 Volt gelegt. Die Chip-Enable-Leitung des ACIA ist zwar im Grunde das Gegenstück zur Z80 IORQ (I/O Request), muß aber über M1 durchgesteuert werden. Andernfalls könnte ein I/O-Zugriff auch während der Interrupt-Acknowledge-Phase des Z80 auftreten, bei der sowohl IORQ als auch M1 Low werden. Eine weitere Steuerung findet über den Punkt A7 statt.

Der Erweiterungsanschluß des Schneider verfügt nicht über eine geeignete Leitung für den direkten Anschluß von Ein- und Ausgabegeräten, also muß eine externe Codierung vorgenommen werden. Die I/O-Adressierung nutzt beim CPC alle 16 Adreßbits (der Inhalt von Register B wird bei den meisten Schreib-/Lesebefehlen des Z80 als obere acht Adreßbits ausgeben). Über den Erweiterungsbus sind die Adressen zwischen &F800 und &FBFF verfügbar, wobei die unteren acht Bits für Peripheriegeräte zwischen &E0 und &FE liegen müssen.

Das ist zum Glück nicht sehr schwierig, weil bei allen internen I/O-Adressen A10 auf +5 Volt liegt. Um unseren Adreßbereich anzusprechen, müssen wir nur ein Low auf A10 und Highs auf A5 bis A7 erkennen. Das läßt sich durch die Verbindung von A10 mit CS2 und von A5 bis A6 mit CS0 und CS1 erreichen. A7 wird verwendet, um das Enable-Signal (E) durchzuschalten, das damit nicht mehr aktiv werden kann, wenn A7 nicht auf High liegt.

Im Gegensatz zum 6502-Prozessor stellt der Z80 kein Read/Write-Signal zur Verfügung. Wir behandeln daher die Schreib-Leseregister des 6850 als unterschiedliche Adressen – die R/W-Leitung wird mit der Adreßleitung A9 verbunden. Die interne Register-Select-Leitung koppeln wir mit A8. Dadurch stehen die ACIA-Register mit den in der Tabelle angegebenen Adressen in Verbindung.

Man darf nicht vergessen, daß die Schreib- und Lesevorgänge zur ACIA über eine Adreßleitung gesteuert werden – damit kann nicht zu den Adressen der Read-Only-Register (&FAE0, &FBE0) geschrieben werden. Andernfalls könnte sowohl der Z80 als auch der ACIA gleichzeitig ein Byte auf den Systembus schikken. Noch gefährlicher ist ein Leseversuch an den Write-Only-Adressen (&F8E0, &F9E0) – ein Schreibversuch zum ACIA-IC würde völlig undienlich über einen Datenbus in undefiniertem Zustand laufen.

Die Prüfung der Platine entspricht dem Test der ursprünglichen Version. Wir beschreiben den Ablauf noch einmal mit den beim Schneider nötigen Veränderungen:

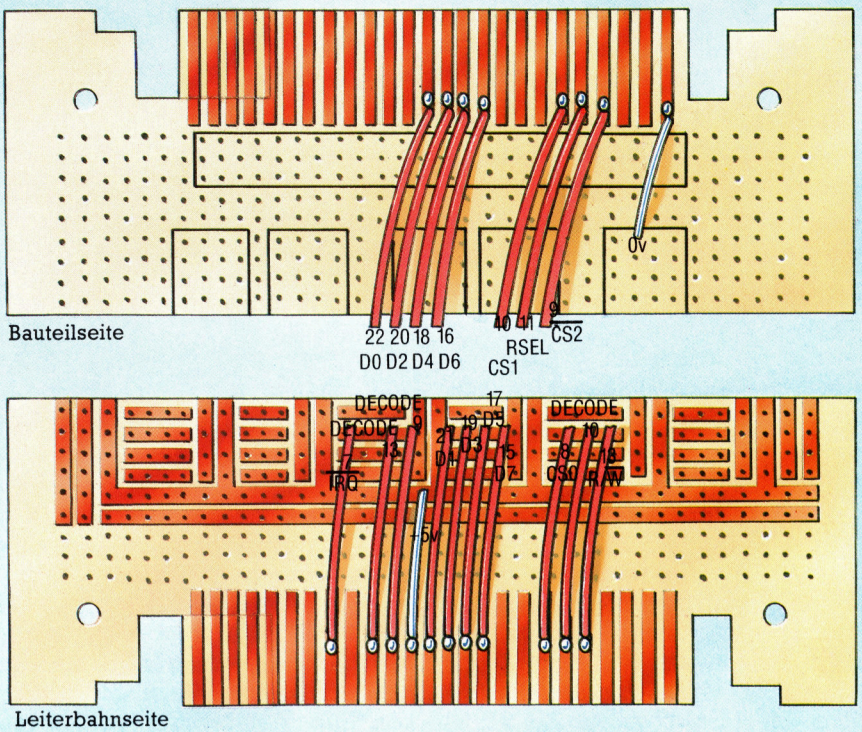
Liste der Bauteile

Die Bauteile bekommen Sie im gutsortierten Elektronik-Fachhandel. Eine preiswerte Bezugsquelle sind auch Versandhandlungen.

Anzahl	Bauteil
1	Styroflex-Kondensator 1 nF
2	Styroflex-Kondensator 100 nF
1	Widerstand 270 Ohm
3	Widerstände 220 Ohm
2	Widerstände 680 Ohm
2	5-polige 180-DIN-Buchsen für die Platinenmontage
1	MC68B50 ACIA-Chip
1	Optokoppler 6N139
1	Sechsfach-TTL-Invertierer 74LS04
1	Dreifach-NAND-Gatter mit je drei Eingängen; 74LS10N
1	24-poliger IC-Sockel
2	14-polige IC-Sockel
1	8-poliger IC-Sockel
1	Schwingquarz, 2,0 MHz
1	Diode 1N914
1	Spezialplatine mit Kontaktstreifen
2	50-polige Platinenstecker
30 cm	50-poliges Flachkabel

Die linke Seite der Platine wird mit dem Erweiterungsanschluß des Schneider CPC verbunden. Auf der Zeichnung sind die notwendigen Drahtbrücken für den Busanschluß des ACIA-ICs, des 74LS10-Decoders und der Stromversorgung zu sehen. Erst wenn alle Leitungen komplett sind, werden die vier ICs in ihre Sockel gesetzt (Auf richtige Lage der Markierung achten!). Jetzt fehlt noch die Verbindungsleitung, die aus dem 50-poligen Flachkabel mit zwei Platinensteckern zusammengebaut wird. Wenn die Leitung am Rechner und an der Platine aufgesteckt wird, sollte der Computer unbedingt ausgeschaltet sein. Danach geht's an den Test der MIDI-Schnittstellenplatine.

Kontaktbelegung



1. Verbinden Sie die IN- und die OUT-Buchse der Platine mit einem 5-poligen DIN-Kabel.
2. Initialisieren Sie den ACIA mit:
   OUT &F8E0,3

3. Die Festlegung von Taktfrequenz und Datenformat sowie die Abschaltung der Sende/Empfangsinterrupts besorgt der Befehl:
   OUT &F8E0,&16

4. Lesen Sie nun das ACIA-Statusregister aus:
   PRINT INP(&FAE0)

Hier sollte 2 ausgegeben werden.
5. Als nächstes übertragen Sie ein serielles Byte vom Ausgang der Schnittstelle zu ihrem Eingang:
   OUT &F9E0,x

x kann eine beliebige ganze Zahl zwischen 0 und 255 sein.
6. Mit erneutem Auslesen des Statusregisters wird geprüft, ob das Byte angekommen ist:
   PRINT INP(&FAE0)

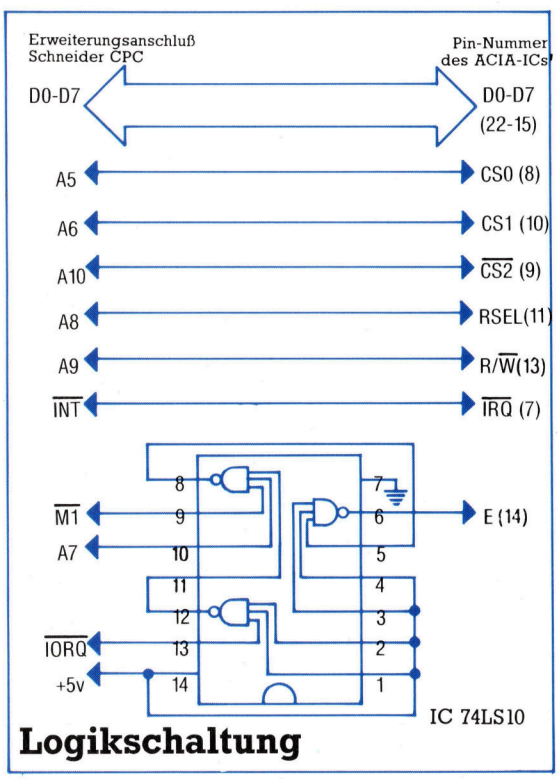
Jetzt sollte 3 angezeigt werden.
7. Ob das Byte auch ohne Verfälschung angekommen ist, erfahren Sie durch:
   PRINT INP(&FBE0)

Jetzt sollte das x von Schritt 5 erscheinen. Wiederholen Sie die Schritte 4 bis 7 mit verschiedenen x-Werten.

ACIA-Registeradressen

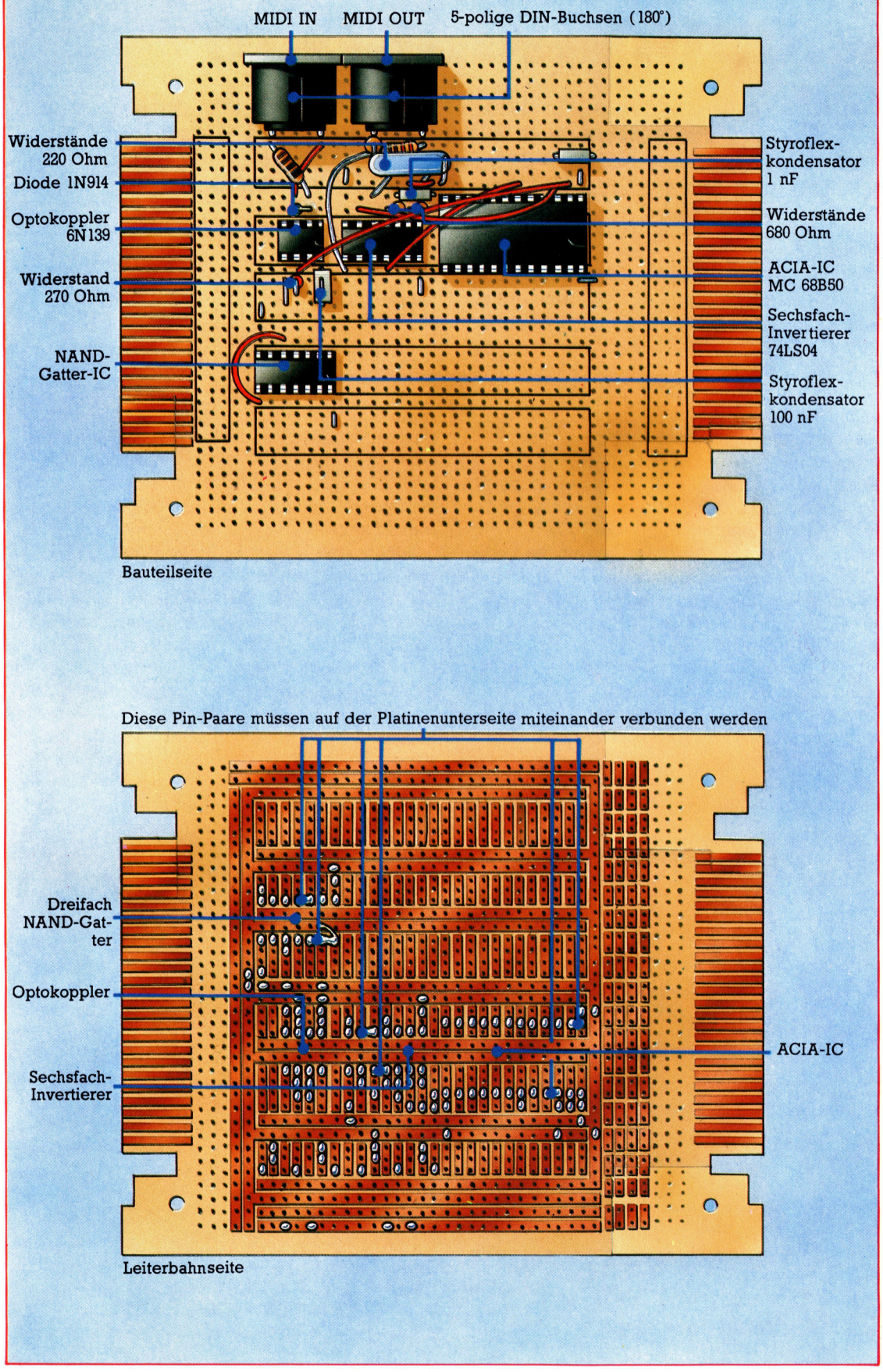
Adresse	Register	
&F8E0	Control	(WRITE ONLY)
&F9E0	Transmit data	(WRITE ONLY)
&FAE0	Status	(READ ONLY)

Die Ur-Version unserer MIDI-Schnittstelle wurde für Rechner mit einer 65XX-CPU entwickelt. Damit sie auch am Schneider CPC (Z80-Prozessor) läuft, müssen die Bus-Signale vorher umcodiert werden, bevor sie zum ACIA-Chip weitergeleitet werden können. Die erforderliche Logikschaltung ist im nebenstehenden Bild dargestellt. Für die Signalwandlung sind nur drei in einem IC integrierte NAND-Gatter nötig, die über je drei Eingänge verfügen.



Logikschaltung

Platinenlayout



Alle Bauteile der MIDI-Schnittstelle werden auf eine Spezialplatine montiert. Beginnen sie beim Aufbau mit den passiven Komponenten: Widerstände, Kondensatoren, IC-Sockel und DIN-Buchsen. Die Verbindungsleitungen auf der Platine lassen sich gut mit isolierter Wire-Wrap-Litze herstellen. Danach den Quarz und die Diode einlöten (Achtung, der farbige Kennungsring muß oben liegen). Die kleine Drahtbrücke unter dem Kondensator (neben dem Sechsfach-Invertierer) darf nicht vergessen werden!

Schneiders Maschinencode

Am Anfang dieser neuen Serie über das Betriebssystem der Schneider CPC 464 und 664 untersuchen wir zunächst, wie RAM und ROM ein- und ausgeblendet werden. Weiterhin verschaffen wir uns über einen speziellen Speicherbereich – den „Jumpblock" – Zugang zu den Firmware-Routinen.

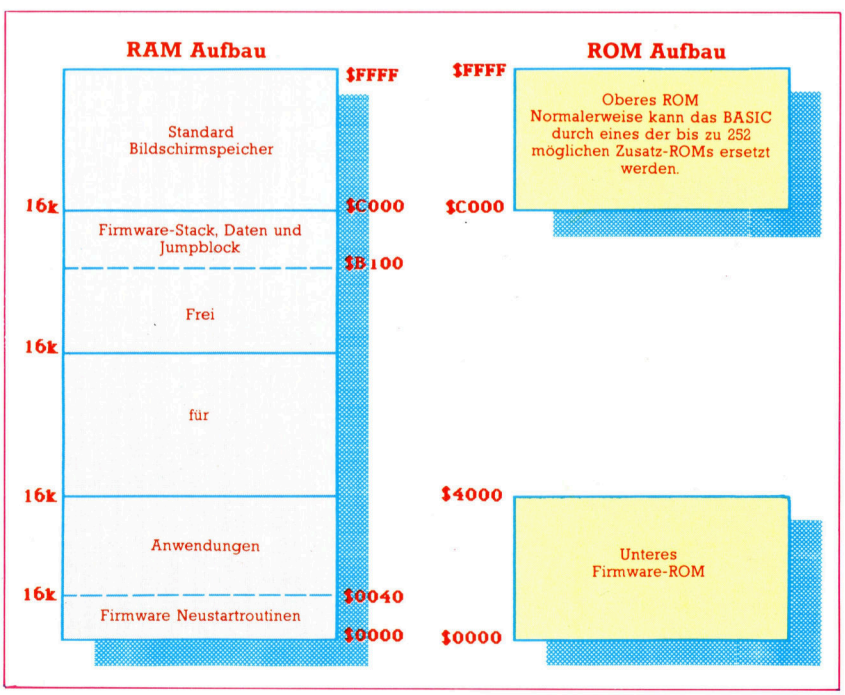
Die Schneider-Computer CPC 464 und 664 enthalten zwar keine außergewöhnlichen technischen Neuerungen, doch gleicht ein umfassend angelegtes Betriebssystem diesen Mangel aus, so daß die Geräte mit minimaler Hardware auskommen. Das Ergebnis ist ein preisgünstiger Micro mit hoher Leistung.

Der CPC 464 arbeitet mit einer Z80A CPU, 64 KByte RAM, 32 KByte ROM, einem 8255 PIA (Peripheral Interface Adaptor), einem AY-3-8912 Klanggenerator, einer 6845 CRTC (Bildschirmsteuerung) und einer speziellen Logikschaltung (ULA – Uncommitted Logic Array).

Das Betriebssystem, auch „Firmware" genannt, belegt die untere Hälfte des ROM – in der oberen Hälfte befindet sich die Schneider-Version von BASIC (das Locomotive BASIC).

Die Memory-Map zeigt, daß der 32 KByte umfassende ROM-Chip zwei logisch getrennte Bereiche zu je 16 KByte enthält, die sich über die ULA unabhängig voneinander ein- und ausschalten lassen. Die beiden Bereiche ersetzen die ersten und letzten 16 KByte des RAM, während die mittleren 32 KByte zum Schreiben und Lesen zur Verfügung stehen. Die ULA kann außerdem in das obere ROM bis zu 252 Zusatz-ROMs einschalten.

Die Memory-Map des Schneider zeigt, wie der 32K-ROM-Chip die oberen und unteren 16K des RAM „ausschaltet". Die Firmware kann beide RAM-Seiten ein- und ausblenden und auch prüfen, ob die bis zu 252 Zusatz-ROMs vorhanden sind. Die Bildschirmdaten lassen sich in jeden der vier 16K-RAM-Blöcke legen.



Der Bildschirm wird von einem 6845 Chip gesteuert und belegt 16 KByte RAM – normalerweise direkt unter der RAM-Obergrenze. Die Bildschirmdaten lassen sich jedoch in jeden der vier 16K-Blöcke übertragen, so daß mehrere Bildschirminhalte blitzschnell gegeneinander ausgetauscht werden können.

Wenn das obere ROM eingeschaltet ist – etwa bei aktiviertem BASIC – veranlaßt die ULA, daß alle Daten der oberen 16 KBytes aus dem ROM gelesen werden. Alles, was in diesen Block geschrieben werden soll, muß das ROM passieren, um das darunterliegende RAM (normalerweise der Bildschirmspeicher) erreichen zu können. Das obere ROM läßt sich auf mehrere Arten ausschalten, wenn die oberen 16 KBytes RAM gelesen werden sollen.

Auch das untere Firmware-ROM wird auf diese Weise angesprochen. Wie aber kann das Betriebssystem funktionieren, wenn das untere ROM ausgeschaltet ist? Die Lösung ist einfach: Das untere ROM wird von den Programmen nicht direkt angesprochen. Alle Aufrufe an das Betriebssystem gehen über einen „Jumpblock", der in dem festen Speicherbereich liegt. Er enthält Einträge, die einen Sprung (oder einen vergleichbaren Vorgang) auf die Routinen der Firmware auslösen. Durch den Jumpblock lassen sich die Adressen der Firmware-Routinen verändern, ohne daß bestehende Programme in Mitleidenschaft gezogen werden. Sie sollten Firmware-Routinen nur über den Jumpblock aufrufen.

Der Jumpblock liegt im dritten der 16K Blöcke und kann nicht wie ein ROM ein- und ausgeschaltet werden. Normalerweise läßt sich dieser Bereich auch nicht vom Anwender überschreiben. Es gibt jedoch Situationen, in denen der Jumpblock geändert werden muß, damit ein Teil des Betriebssystems verschoben oder ausgetauscht werden kann.

Alle Einsprungspunkte des Betriebssystems sind im Jumpblock eingetragen. Ein Zugriff ruft

zuerst den entsprechenden Eintrag des Jumpblocks auf. Dieser gibt einen RST-Befehl aus, der wiederum die Firmware-Routine aktiviert. Die Einträge sind jeweils drei Bytes lang und enthalten beim Einschalten des Systems einen RST-Befehl gefolgt von der Einsprungsadresse des Betriebssystems.

Wenn es nicht benötigt wird, kann das untere ROM ausgeschaltet werden. Im RAM befindet sich eine Kopie der Neustart-Routinen, die das Firmware-ROM vor jedem Zugriff wieder anschalten. Nach Ablauf der Betriebssystemroutinen werden die ROMs in ihren ursprünglichen Zustand versetzt. Damit aber ist die Frage gelöst, welche ROMs vor Aufruf einer Betriebssystemroutine eingeschaltet werden müssen. Die meisten Neustart-Befehle stellen zusätzliche Kommandos dar, die hauptsächlich das Schalten der ROMs steuern.

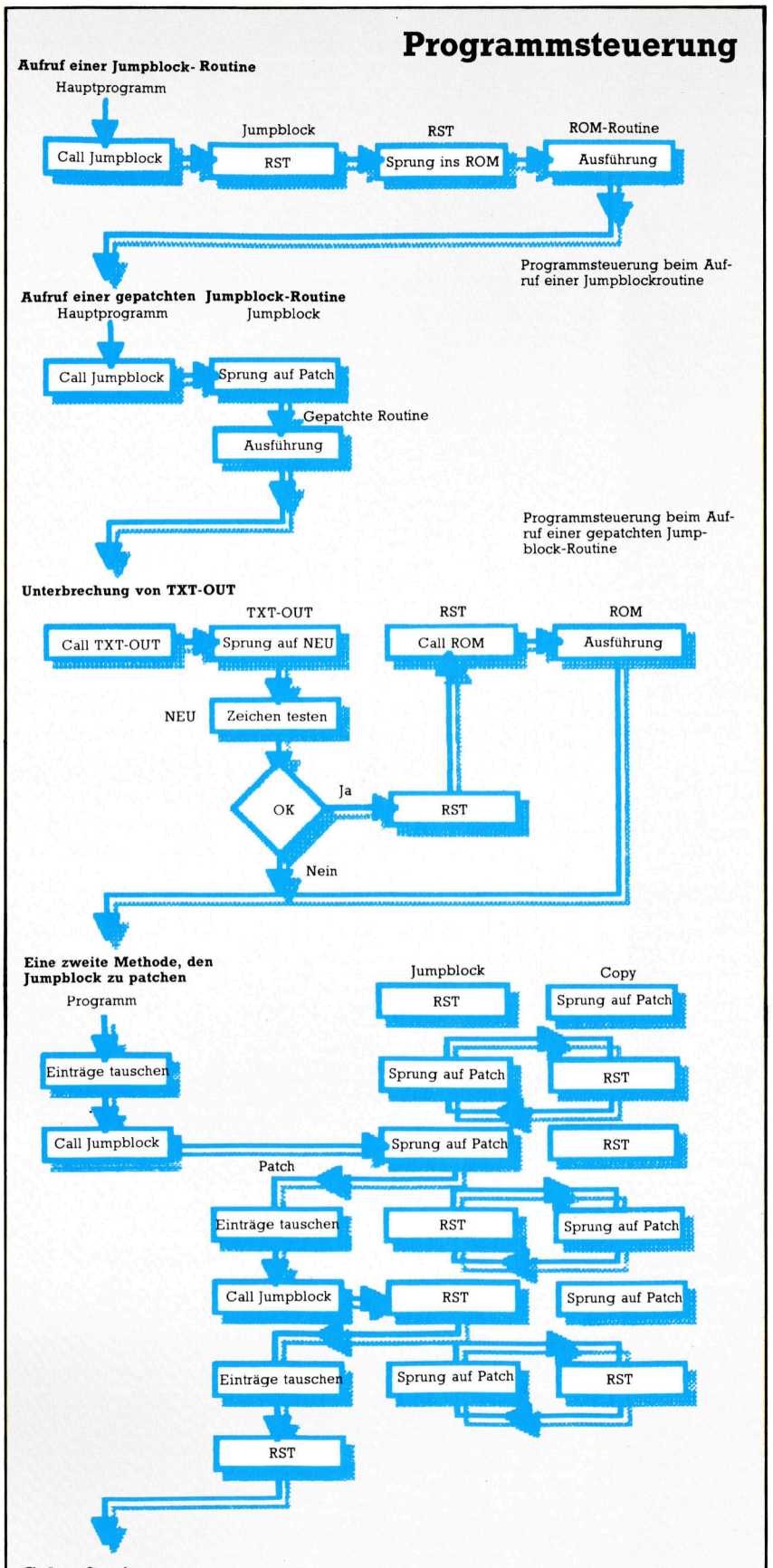
Per Jumpblock lassen sich anstelle der Systemroutinen auch andere Routinen einsetzen. So können Sie beispielsweise eigene Bildschirmmodule einfügen, indem Sie den entsprechenden Jumpblock-Eintrag auf die neue Routine zeigen lassen. Das Bild zur Programmsteuerung zeigt die unterschiedliche Steuerung bei einer normalen Befehlsübergabe an das Betriebssystem und bei einer vom Anwender eingesetzten Unterbrechung. Mit dieser Methode des „Patchens" wird auf dem CPC 464 und 664 auch zwischen den Cassetten- und Diskettenbefehlen umgeschaltet.

An den Bildschirm

Nehmen Sie an, Sie haben eine Routine namens SEND SCREEN geschrieben, die Text zum Bildschirm sendet, und die Sie statt der entsprechenden Betriebssystemroutine einsetzen wollen. Der Jumpblock-Eintrag für die Zeichendarstellung auf dem Bildschirm heißt TXT WR CHAR und wird über OBB5DH adressiert. Patch trägt einfach die Adresse der neuen Routine in den Jumpblock ein.

Die eben beschriebene Routine funktioniert ohne Probleme, wenn eine neue Routine das im Jumpblock eingetragene Original vollständig ersetzt. Schwierigkeiten gibt es, wenn der ursprüngliche Eintrag ebenfalls benötigt wird oder von der neuen Routine aus eine weitere Routine des Jumpblocks aufgerufen wird. Hier ein Beispielprogramm in BASIC:

```
10  MODE 1                    Bildschirm löschen
                              und Cursor nach links
                              oben
20  WHILE -1                  Endlosschleife star-
                              ten
30  INPUT "Zeichen-           Zeichencode holen
    code";C;
40  IF C<0 OR C>              Überprüfung des
    255 then ?:GOTO 30        Zeichens
50  ?CHR$(C);                 Zeichen senden
60  WEND                      Zurück zum Schlei-
                              fenanfang
```



Gehupft wie gesprungen
Über den Jumpblock kann der Anwender problemlos auf die Firmware zugreifen und Systemaufrufe leicht auf seine eigenen Routinen leiten. Dabei muß der Jumpblock „gepatcht" werden. Das Bild zeigt die Abläufe der verschiedenen Methoden.

Mit diesem Programm lassen sich alle Grafikzeichen und auch die Steuercodes anzeigen. Es gibt jedoch einen Code, der den Bildschirm abschaltet und damit das Problem „aufhängt". Sie können das Senden dieses Codes leicht verhindern, wenn Sie in Zeile 40 einen Test einsetzen. In unserem Beispiel werden wir den Test jedoch in Zeile 50 (in der das Zeichen angezeigt wird) durchführen, damit wir das Zeichen über den Jumpblock-Eintrag TXT OUTPUT (bei $BB5A) abfangen können, bevor es zum Bildschirm gesandt wird. Die Originalroutine verlangt, daß das auszugebende Zeichen im Register A gespeichert wird und alle Register vor dem Rücksprung gesichert werden.

Das folgende Programmlisting zeigt die einfachste Methode, einen Jumpblock-Patch einzurichten:

```
        txt_ou: equ  #BB5A              ;jblock output routine
        disabl: equ  21                 ;screen disable char
        ; New routine follows - char to be displayed is given in A
FE15    new:    cp   disable            ;is it disable VDU?
C8              ret  z                  ;if so ignore it
000000  entry:  defb 0,0,0              ;insert original entry
        ;The return address is already on the stack on entry
        ;to the routine, so no RET instruction required
        ;Existing jumpblock entry at txt_out should be copied to ENTRY
        ;Original entry should be replaced with JMP to NEW
```

Sie funktioniert bei allen im Jumpblock eingetragenen Standardroutinen. Bevor Sie eine Routine „patchen", müssen Sie jedoch sicherstellen, daß sie nicht bereits zuvor auf die gleiche Weise bearbeitet wurde. Einige Routinen des Jumpblock funktionieren nur von ihrer Originaladresse aus – wird der Eintrag kopiert, arbeitet sie nicht mehr korrekt.

Unser Bild zeigt auch die alternative Methode, einen Jumpblock zu patchen. Dabei wird der aktuelle Eintrag kopiert und an seine Stelle ein Sprung auf die neue Routine gesetzt. Die neue Routine stellt schließlich den ursprünglichen Eintrag wieder an seinen angestammten Platz und aktiviert ihn. Vor dem Rücksprung ins Programm wird der Jumpblock-Eintrag dann wieder auf die neue Routine gesetzt.

Unser zweites Maschinencodelisting arbeitet mit der zweiten Methode. Sie erscheint auf den ersten Blick zwar umständlicher, funktioniert aber bei allen Jumpblock-Einträgen.

```
        txt_ou: equ  #BB5D              ;jblock output routine
        disabl: equ  21                 ;screen disable char
FE15    new:    cp   disable            ;routine entry point
C8              ret  z                  ;ignore disable
        ;now switch old entry into jumpblock and call it
CD2529          call copy               ;switch entry
CD5DBB          call txt_out            ;send character
CD2529          call copy               ;copy the entry back
C9              ret
        ;
F5      copy:   push af                 ;must preserve all regs
C5              push bc
D5              push de
E5              push hl
0603            ld   b,3                ;no of bytes in entry
215DBB          ld   hl,txt_out         ;old entry
113F29          ld   de,txt_cpy         ;copy area
4E      cloop:  ld   c,(hl)             ;read current entry
1A              ld   a,(de)             ;and its replacement
77              ld   (hl),a             ;substitute it
79              ld   a,c
12              ld   (de),a             ;and save the old one
23              inc  hl                 ;point to next byte
13              inc  de
10F7            djnz cloop              ;do all three
E1              pop  hl
D1              pop  de
C1              pop  bc
F1              pop  af
C9              ret
        ; new jumpblock entry is store below until required
C3      txt_cp: defb #c3                ;code for jmp
1829            defw new                ;to substitute routine
```

Wichtig ist, daß das neue Modul wie die ursprüngliche Firmware-Routine reagiert. Alle Parameter, die an die Routine gehen und von dort empfangen werden, müssen daher die gleiche Definition haben wie in der Originalroutine. Unser Beispielprogramm sichert deshalb die zurückgelieferten Register vor Ablauf des Kopiervorganges.

Die neuen Routinen sollten unterhalb von HIMEM liegen. Die Aufteilung des Arbeitsspeichers zeigt, daß die unteren drei 16K-RAM-Blöcke für BASIC zur Verfügung stehen. Die untere Grenze ist fest eingebaut, während HIMEM die obere Grenze bestimmt. BASIC-Programme „wachsen" von der Untergrenze aus in Richtung HIMEM.

Da der für Maschinencode reservierte Platz nicht von BASIC überschrieben werden darf, muß HIMEM heruntergesetzt werden. HIMEM sollte jedoch nur um die Länge der Routine (plus seines Datenbereiches) herabgesetzt werden, damit BASIC nicht zuviel Platz weggenommen wird. Die Routine wird dann in den auf diese Weise „befreiten" Platz geladen.

Der Inhalt des Speicherbereiches zwischen #0000 und #003F ist in RAM und ROM identisch. Damit ist die Ausführung von RST unabhängig davon, ob das untere ROM ein- oder ausgeschaltet ist. Der Z80 verfügt über acht Neustart-Befehle, die hauptsächlich für die Speicherverwaltung eingesetzt werden. Die Tabelle zeigt ihre Funktionen.

Hexadresse	Funktion
0000 RST 0	Reset (Wie beim Anschalten des Gerätes)
0008 RST 1	Sprung auf eine Routine der unteren 16K. Die beiden Bytes hinter RST werden als 14-Bit-Adressen (0000 bis 3FFF) interpretiert. Bit 15 gesetzt = oberes ROM abgeschaltet, Bit 14 gesetzt = unteres ROM abgeschaltet.
0010 RST 2	Aufruf eines Zusatz-ROMs – hinter RST stehen zwei Bytes. Bit 14 und 15 zeigen auf eins der Zusatz-ROMs; Bit 0 bis 13 enthalten ein Offset, das auf #C000 addiert wird und so die Adresse der gewünschten Routine angibt.
0018 RST 3	Ruft eine beliebige Routine im ROM oder RAM auf. RST plus zwei Bytes zeigen auf eine Drei-Byte-Adresse. Dabei geben Byte 0/1 die Adresse an, während Byte 2 das ROM bezeichnet: #00–#FB = Nummer des Zusatz-ROM – oberes ROM an, unteres aus; #FC = oberes und unteres ROM an; #FD = oberes RAM an, unteres aus; #FE = oberes ROM aus, unteres an; #FF = beide ROMs aus.
0020 RST 4	Das Byte, auf das HL zeigt, aus dem RAM lesen – oberes und unteres ROM abschalten. Beim Rücksprung enthält A das gelesene Byte.
0028 RST 5	Firmware-Aufruf. Schaltet das untere ROM an und aktiviert die Routine, auf die die beiden Bytes hinter RST zeigen.
0030 RST 6	Neustart durch den Anwender. Bytes #30 bis #37 sind unbelegt.
0038 RST 7	Einsprungspunkt der Interrupts. Sollte möglichst nicht eingesetzt werden. Wenn Sie Interrupts des Modus 1 verwenden wollen, können Sie mit „Ereignissen" der Firmware arbeiten, auf die wir später genauer eingehen.

Im Großformat

Obgleich die Lücke zwischen Micro- und Minicomputern kleiner wird, kommen die Micros an die Leistungsfähigkeit ihrer größeren Brüder noch nicht heran. Als Beispiel für ein bewährtes Minicomputersystem stellen wir die VAX 11/780 vor.

Angesichts der Möglichkeiten moderner Microcomputer mit ihrer Vernetzbarkeit und ihrem vielseitigen Softwareangebot ist die Frage naheliegend, weshalb sich die großen und kostspieligen Minicomputersysteme von Firmen wie DEC oder Prime überhaupt noch verkaufen lassen. Täten es ein paar IBM PCs mit entsprechender Plattenkapazität nicht auch? Wohl kaum, denn das Leistungsvermögen der Micros reicht an das eines mittleren Minicomputers bei weitem nicht heran; der Abstand in Rechengeschwindigkeit, Speicherkapazität sowie Soft- und Firmware-Angebot ist nach wie vor enorm.

Zu den bekanntesten Minicomputern gehören die VAX-Rechner von DEC (Digital Equipment Corporation). Die VAX-Familie umfaßt Systeme unterschiedlicher Leistung, von der MicroVAX, die nicht wesentlich größer als ein üblicher Bürorechner ist, bis zum neuen Spitzenmodell 8800. Dieser Artikel beschäftigt sich mit der schon länger angebotenen „mittleren" VAX 11/780, deren Architektur sich aber nicht grundsätzlich von der anderer VAX-Versionen abhebt.

Schon äußerlich besteht keine Gefahr, die 11/780 mit einem Micro zu verwechseln: Die Maschine beansprucht einen 1 m breiten, 1½ m hohen und ¾ m tiefen Schrank. Und während die meisten kleineren Computer sich mit einer einzigen Leiterplatte, allenfalls noch einer zusätzlichen Erweiterungskarte begnügen, sind bei der 11/780 allein 16 Platinen für Speicherchips vorgesehen, abgesehen von den CPU- und Interface-Karten.

Auch bei der CPU ist es nicht mit einem Chip oder einer Leiterplatte getan. Alle VAX-Rechner verfügen als 32-Bit-Maschinen über einen Adreßraum von vier Gigabyte (vier Milliarden Byte), und eine solche RAM-Kapazität wäre trotz des Preisverfalls bei den Chips weder zu bezahlen noch unterzubringen. Daher wird zur vollen Nutzung der Adressierungsmöglichkeiten ein „virtueller" Speicher organisiert: Nur ein Bruchteil der vier Gigabyte liegt tatsächlich als RAM vor, der Löwenanteil dagegen als Platten- oder Bandspeicher. Hardwaremäßig wird jeweils geprüft, ob ein angefordertes Byte sich bereits im RAM befindet; andernfalls wird die entsprechende „Page" (Seite) automatisch vom Externspeicher geladen.

Das „Paging"-Verfahren beruht auf einer festen Speichereinteilung in Seiten zu 512 Byte. Ist das verfügbare RAM bei Abruf neuer Seiten voll belegt, werden bereits bearbeitete wieder peripher ausgelagert.

Eine Nummer größer

Die VAX-Rechner laufen unter dem DEC-Betriebssystem VMS (Virtuelles Mehrbenutzer-System) oder der Unix-Version „Ultrix 32". Das VMS verfügt über eine leistungsfähige Kommandosprache und einen großen Vorrat an Dienstprogrammen wie etwa Editierhilfen, Sortieralgorithmen und Routinen für die Datenfernübertragung. Als Sprache wird lediglich ein Makro-Assembler mitgeliefert, es sind jedoch Compiler für rund zwanzig Hochsprachen verfügbar. Der VAX-Assembler ist für Programmierer, die den Z80- oder 6502-Code kennen, so etwas wie eine Offenbarung. Die 11/780 bietet z. B. 16 Register zu 32 Bit, die bis auf vier reservierte (für Stack Pointer, Befehlszähler usw.) frei verwendbar sind. Und während Micros intern meist nur ganzzahlig rechnen können, gibt es hier Maschinenbefehle für eine 64-Bit-Gleitkomma-Arithmetik, außerdem für die Zeichen-Manipulation. Zu dem kompletten, 248 Kommandos umfassenden Befehlssatz ge-

Die Kosten für den VAX-Minicomputer 11/780 von DEC dürften wohl nur in größeren Firmen locker zu machen sein. Eine derartige Investition zahlt sich aber aus, wenn kontinuierliche Massenspeicherung von Daten, hohe Verarbeitungsgeschwindigkeit sowie große Flexibilität gefordert sind.

hören auch die üblichen Schiebe- und Vergleichsoperationen sowie eine Reihe spezieller Anweisungen für die CPU-Steuerung.

Um am VAX-Betrieb teilzunehmen, benötigt man zunächst für das Einloggen vom Systemoperator eine Benutzerkennung und ein Paßwort. Damit haben Sie Zugang zu bestimmten Systembereichen; an diese Kennung sind individuelle Befugnisse und auch Einschränkungen gebunden, so daß Sie nicht auf jede Datei und jede Systemfunktion zugreifen können. Genauso lassen sich eigene Daten bei Bedarf gegen unberechtigtes Lesen oder Überschreiben schützen.

Kommandoprozeduren

Die VAX-Kommandosprache DCL (Digital Command Language) bietet viele Möglichkeiten, die sonst nur in Hochsprachen zu finden sind. Sie können Variablennamen einführen, GOTO- und IF-Anweisungen ähnlich wie in BASIC verwenden und außerdem über eine Reihe „lexikalischer" Funktionen Systeminformationen wie Rechenzeiten und aktuelle Benutzernamen abfragen. Die meisten dieser Befehle finden nur in „Kommandoprozeduren" Verwendung; das sind Sequenzen von DCL-Befehlen, die wie ein Programm abgearbeitet werden. Für viele Zwecke reicht aber auch schon ein einziges Kommando. „Show Users" beispielsweise veranlaßt die Ausgabe einer Benutzerliste. Lexikalische Funktionen innerhalb von Kommandoprozeduren erlauben es u. a., an jeden Benutzer eine Mitteilung zu senden.

Bei der Mehrzahl der Befehle gibt es verschiedene Modifikationen. So löscht DELETE*.* sämtliche Dateien einer Directory, soweit sie nicht geschützt sind, während DELETE/CONFIRM*.* zusätzlich für jedes File eine Löschfreigabe verlangt. Die vollständige Befehlsliste ist sehr umfangreich und enthält neben den üblichen Kommandos für den Umgang mit Dateien auch solche zur Umgestaltung der Programmierumgebung und zur Kommunikation mit anderen Benutzern.

Die Verbindung des Rechners mit der Außenwelt erfolgt über zwei Schnittstellen: Der „Unibus" (ein DEC-Standard) dient zum Anschluß von Terminals, langsamen Platten- und Bandspeichern sowie Druckern und einer Vielzahl von Spezialgeräten wie Plottern und Steuerungen, die zum DEC-Programm gehören. Sehr viel höhere Übertragungsraten (im Bereich von 10 MByte/s) gestattet der „Massbus", der beispielsweise für Plattenlaufwerke verwendet wird.

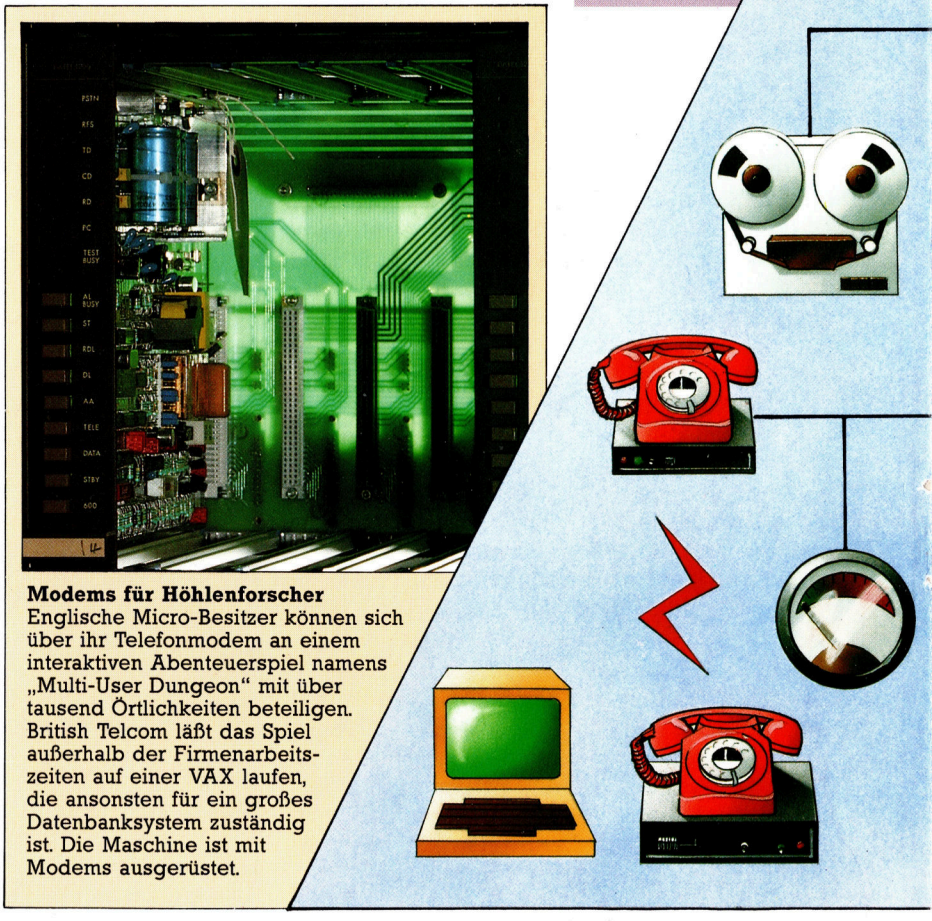
Die kleinste 11/780-Konfiguration verfügt über zwei MByte RAM und acht Terminalanschlüsse. Für einen sinnvollen Betrieb sind zumindest noch zwei Plattenlaufwerke mit einer Minimalkapazität von je zwei MByte erforderlich. Die Anlage läßt sich auf acht Laufwerke und einen Arbeitsspeicher von acht MByte erweitern, durch Einbau zusätzlicher Interface-Karten außerdem auf 64 Terminals.

Das VMS-Betriebssystem eignet sich hervorragend für die Programmentwicklung: Die Kommandosprache ist logisch strukturiert, und für alle Befehle gibt es jederzeit aufrufbare Help-Files. Ferner sind praktische Bibliotheksroutinen für die Übernahme in eigene Programme verfügbar. Daher entfällt weitgehend die Notwendigkeit, Dinge wie Bildschirmmasken oder mathematische Algorithmen selbst zu programmieren. Auch erspart die gemeinsame Benutzung der gleichen Plattendateien durch die Angehörigen von ganzen Software-Mannschaften all den Ärger, der bei der Programmentwicklung auf getrennten Microcomputern dadurch entsteht, daß jemand in seinen Dateien Änderungen vornimmt, ohne die anderen zu informieren.

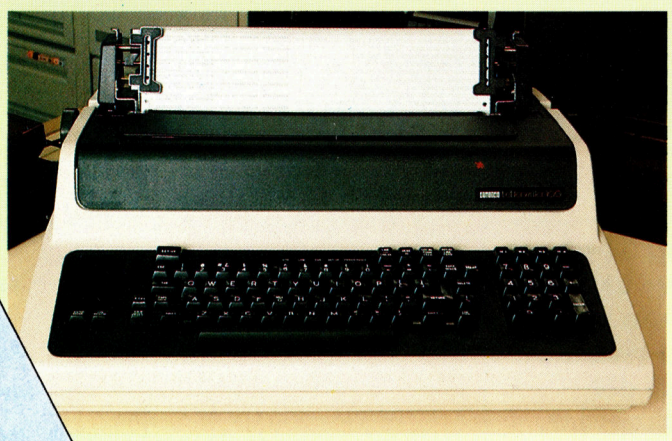
Insgesamt bietet ein derartiger Minicomputer eine sehr benutzerfreundliche Programmierumgebung, und dank des hohen Datendurchsatzes laufen auch aufwendige Programme wesentlich schneller als auf jedem Micro. Das hat natürlich seinen Preis: Die VAX 11/780 kostet in Standardausführung schon so viel wie eine Villa in bester Wohnlage. Auch die laufenden Kosten sind wegen des elektrischen Anschlußwerts von 6 kW zuzüglich Klimaanlage und wegen der Wartungsgebühren nicht unerheblich. Dafür wird perfekter Service garantiert.

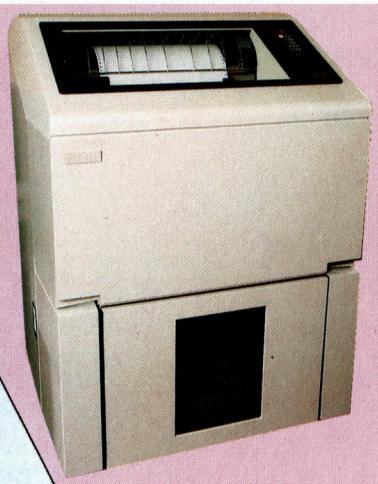
Spitzenposition
Der Systemoperateur der VAX-Anlage verkehrt mit dem Computer über den Konsolschreiber rechts im Bild, der wie eine Fernschreibmaschine betrieben wird und auch die Dialogantworten des Systems oder Zustandsmeldungen ausdruckt. Auf diese Weise entsteht ein vollständiges Protokoll der Zwiesprache des Operators mit der betriebenen Anlage.

Zeilendrucker
Über den DEC-Unibus kann an die VAX ein schneller Zeilendrucker angeschlossen werden, der pro Sekunde an die zwanzig Zeilen mit 132 Zeichen zu Papier bringt.



Modems für Höhlenforscher
Englische Micro-Besitzer können sich über ihr Telefonmodem an einem interaktiven Abenteuerspiel namens „Multi-User Dungeon" mit über tausend Örtlichkeiten beteiligen. British Telcom läßt das Spiel außerhalb der Firmenarbeitszeiten auf einer VAX laufen, die ansonsten für ein großes Datenbanksystem zuständig ist. Die Maschine ist mit Modems ausgerüstet.





Mini-System
Die VAX kommuniziert mit der Peripherie über zwei Busse: Der „Massbus" ist ein Hochgeschwindigkeitsbus, der eine Datenrate im Bereich von 10 MB/s erlaubt und schnelle Band- oder Plattenlaufwerke bedienen kann; der „Unibus" wird für langsamere Peripheriegeräte verwendet. Ohne zusätzlichen Aufwand können acht Terminals an der 11/780 betrieben werden, außerdem Drucker und Plotter. Der Unibus trägt auch insofern zur Flexibilität des VAX-Systems bei, als er die Anbindung an Kommunikationsnetze wie auch den Einsatz industrieller Prozeßsteuerungen ermöglicht.

DEC VAX 11/780

ABMESSUNGEN:
1181 × 1537 × 762 mm

ZENTRALEINHEIT:
VAX-CPU von DEC

SPEICHER:
2–8 MByte RAM, bis zu 4 Gigabyte virtueller Speicher adressierbar

BILDSCHIRM:
Mindestens 25 Textzeilen zu 80 Zeichen; hochauflösende Grafik mit bis zu 1024 × 1024 Pixeln bei über 16 Millionen Farbtönen (abhängig von der Terminal-Ausführung)

SCHNITTSTELLEN:
Massbus für schnelle Platten- und Bandspeicher; Unibus für alle anderen Geräte einschl. Drucker, Plotter und Terminals (über RS232-Interface)

VERFÜGBARE SPRACHEN:
Assemblerpaket wird mitgeliefert, Compiler für fast alle gängigen Hochsprachen erhältlich

DOKUMENTATION:
Die Grundausstattung an System-Handbüchern einschl. der Dokumentation für die Kommandosprache und die Dienstprogramm-Bibliothek umfaßt 30 Bände und belegt ein Zweimeter-Regal; inbegriffen sind ein knappes Nachschlagewerk und Benutzer-Einweisungen.

STÄRKEN:
Die VAX bietet eine äußerst produktive Umgebung für die Software-Entwicklung und die Datenbankorganisation. Auch die Vernetzung von VAX-Rechnern ist vorgesehen.

SCHWÄCHEN:
So ein System ist nicht gerade billig und verursacht zudem hohe laufende Kosten, weil Raumklimatisierung, Luftfilterung und der Abschluß eines Wartungsvertrags unumgänglich sind.

Hoch die Karten!

Wir begannen unser Spiel 17+4 mit der Kartendarstellung und den Routinen zum Mischen auf dem Commodore 64. Nun richten wir unser Augenmerk auf die gleichen Routinen für Schneider, Sinclair und Acorn B. Die prinzipiellen Unterschiede liegen in der Darstellung der Sonderzeichen. Außerdem müssen wir die Kartendarstellung anders aufbauen.

Karten mischen und darstellen

Schneider CPC

Hauptprogramm

```
10 REM **** Amstrad Pontoon ****
20 GOSUB 500:REM init array et
50 REM **** Game Loop ****
55 GOSUB 600:REM init game
```

Aufbau der Felder

```
500 REM **** init arrays etc ****
505 INK 0,16:pink=0:INK 1,3:red=1:INK 2,
0:black=2:INK 3,26:white=3
512 bk$=STRING$(7,207)
513 br$=CHR$(149):li$=STRING$(7,154)
515 DIM x(2),y(2):REM next card position
s
520 su$=CHR$(228)+CHR$(227)+CHR$(226)+CH
R$(229):REM suit symbols
530 DIM cn$(13):FOR i=1 TO 13:READ cn$(i
):NEXT i:REM read number data
540 DIM cd$(13):FOR i=1 TO 13:READ cd$(i
):NEXT i:REM read pattern data
560 DIM dk(52,2):REM card deck
570 GOSUB 3000:REM shuffle pack
580 BORDER 9:PAPER pink:REM screen colou
rs
590 RETURN
600 REM **** init game ****
605 CLS
610 hp(1)=1:hp(2)=1
620 x(1)=0:y(1)=0:x(2)=20:y(2)=0
625 bt=0:sb=0:GOSUB 4200:REM print bet b
ox
630 RETURN
```

Darstellung der Karten

```
900 REM **** print at ****
910 LOCATE tx+1,ty+1:RETURN
1000 REM **** display card ****
1010 GOSUB 1050:GOSUB 1100:RETURN
1050 REM **** card blank ****
1055 tx=x(pl):ty=y(pl):GOSUB 900:REM pos
ition
1060 PEN white:PRINT CHR$(150);li$;CHR$(
156)
1070 FOR i=1 TO 9:tx=x(pl):ty=ty+1:GOSUB
 900:PRINT br$;SPACE$(7);br$:NEXT i
1080 tx=x(pl):ty=ty+1:GOSUB 900:PRINT CH
R$(147);li$;CHR$(153)
1090 RETURN
1100 REM **** card detail ****
1120 tx=x(pl)+2:ty=y(pl)+1:GOSUB 900:REM
 position
1125 ct$=MID$(su$,su,1):REM select suit
type
1127 co=red:IF su>2 THEN co=black
1128 PEN co
1130 FOR i=1 TO 19 STEP 3
1140 cc$=MID$(cd$(cn),i,3):cl$=""
1142 FOR j=1 TO 3:c$=SPACE$(2)
1144 IF MID$(cc$,j,1)="1" THEN c$=ct$+SP
ACE$(1)
1146 cl$=cl$+c$:NEXT j
1150 tx=x(pl)+2:ty=ty+1:GOSUB 900:PRINT
cl$:NEXT i
1160 REM **** add corner labels ****
1170 tx=x(pl)+1:ty=y(pl)+1:GOSUB 900:PRI
NT cn$(cn):REM number
1180 ty=ty+1:GOSUB 900:PRINT ct$:REM sui
t
1190 tx=x(pl)+7:ty=y(pl)+9:GOSUB 900:PRI
NT cn$(cn):REM bottom number
1192 x(pl)=x(pl)+2:y(pl)=y(pl)+1
1195 RETURN
1200 REM **** display card back ****
1210 GOSUB 1050:GOSUB 1250:RETURN
1250 REM **** card back ****
1255 tx=x(pl):ty=y(pl):GOSUB 900:REM pos
ition
1260 FOR i=1 TO 9:tx=x(pl):ty=ty+1:GOSUB
 900:PRINT br$;:PEN red:PRINT bk$;:PEN w
hite:PRINT br$:NEXT i
1270 x(pl)=x(pl)+2:y(pl)=y(pl)+1
1280 RETURN
1300 REM **** deal a card ****
1310 cn=dk(dp,1):su=dk(dp,2)
1320 dp=dp+1:IF dp>52 THEN dp=1:REM bump
 deck
1330 IF fl=1 THEN GOSUB 1200:RETURN:REM
display back of card
1335 GOSUB 1000:RETURN:REM display card
2000 REM **** card number data ****
2010 DATA A,2,3,4,5,6,7,8,9,T,J,Q,K
2100 REM **** card display data ****
2110 DATA "00000000010000000000"
2120 DATA "00001000000000010000"
2130 DATA "00001000001000010000"
2140 DATA "00010100000000101000"
2150 DATA "00010100001000101000"
2160 DATA "00010100010100101000"
2170 DATA "00010101010100101000"
2180 DATA "00010101010101101000"
2190 DATA "10100010101010100101"
2200 DATA "10101010100010101010"
2210 DATA "00000000010000000000"
2220 DATA "00000000010000000000"
2230 DATA "00000000010000000000"
```

Den Stapel mischen

```
3000 REM **** shuffle the deck ****
3005 RANDOMIZE TIME:dp=1
3007 FOR i=1 TO 52:dk(i,1)=0:NEXT i
3010 FOR i=1 TO 4:FOR j=1 TO 13
3020 ep=INT(RND(1)*52)+1:REM select entr
y point
3030 IF dk(ep,1)=0 THEN 3050
3040 ep=ep+1:IF ep>52 THEN ep=1
3045 GOTO 3030
3050 dk(ep,1)=j:dk(ep,2)=i:NEXT j,i
3060 RETURN
```

Acorn B

Geben Sie den Befehl PAGE=&0E00 ein, bevor Sie das Programm eingeben oder laden.

Hauptprogramm

```
10 REM BBC PONTOON
15 MODE 1
20 GOSUB 500
50 REM
55 GOSUB 600
```

Aufbau der Felder

```
500 REM
510 SP$="":FOR I=1 TO 39:SP$=SP$+" ":N
EXT I
512 BK$="":LI$="":FOR I=1 TO 7:LI$=LI$
```

```
+CHR$(195):BK$=BK$+CHR$(166):NEXT I
  513 BR$=CHR$(194)
  515 DIM X(2),Y(2)
  520 SU$=CHR$(211)+CHR$(218)+CHR$(216)+
CHR$(193)
  530 DIM CN$(13):FOR I=1 TO 13:READ CN$
(I):NEXT
  535 CN$(10)=CHR$(128)
  540 DIM CD$(13):FOR I=1 TO 13:READ CD$
(I):NEXT
  560 DIM DK(52,2)
  570 GOSUB 3000
  571 COLOUR 131:COLOUR 0
  572 VDU 23,166,165,66,90,36,90,165,90,
66
  573 VDU 23,195,0,0,0,255,255,0,0,0
  574 VDU 23,194,24,24,24,24,24,24,24,24
  575 VDU 23,193,16,56,124,254,254,238,8
4,56
  576 VDU 23,216,24,60,90,255,255,90,24,
60
  577 VDU 23,218,16,56,124,254,254,124,5
6,16
  578 VDU 23,211,36,126,255,255,255,126,
60,24
  579 VDU 23,213,0,0,0,15,31,28,24,24
  580 VDU 23,201,0,0,0,240,248,56,24,24
  581 VDU 23,202,24,24,28,31,15,0,0,0
  582 VDU 23,203,24,24,56,248,240,0,0,0
  583 VDU 23,128,206,219,219,219,219,219
,206,0
  590 RETURN
  600 REM
  601 DB=0:CS=0
  605 CLS
  620 X(1)=0:Y(1)=0:X(2)=20:Y(2)=0
  630 RETURN
```

Darstellung der Karten

```
  900 REM **** PRINT AT ****
  910 PRINT TAB(TX,TY);:RETURN
 1000 REM
 1010 GOSUB 1050:GOSUB 1100:RETURN
 1050 REM
 1055 TX=X(PL):TY=Y(PL):GOSUB 900:REM PO
SITION
 1060 COLOUR 0:PRINT TAB(TX,TY);CHR$(213
);LI$;CHR$(201)
 1070 FOR I=1 TO 9:PRINT TAB(TX,TY+I);BR
$;"        ";BR$:NEXT I
 1080 PRINT TAB(TX,TY+10);CHR$(202);LI$;
CHR$(203)
 1090 RETURN
 1100 REM
 1120 TX=X(PL)+2:TY=Y(PL)+1:GOSUB 900
 1125 CT$=MID$(SU$,SU,1)
 1127 COLOUR 1:IF SU>2 THEN COLOUR 0
 1130 FOR I=1 TO 19 STEP 3
 1140 CC$=MID$(CD$(CN),I,3):CL$=""
 1142 FOR J=1 TO 3:C$=CHR$(9)+CHR$(9)
 1144 IF MID$(CC$,J,1)="1" THEN C$=CT$+C
HR$(9)
 1146 CL$=CL$+C$:NEXT J
 1150 PRINT TAB(X(PL)+2,TY+((I+3)/3));CL
$:NEXT I
 1170 TX=X(PL)+1:TY=Y(PL)+1:GOSUB 900:PR
INT CN$(CN)
 1180 TY=TY+1:GOSUB 900:PRINT CT$
 1190 TX=X(PL)+7:TY=Y(PL)+9:GOSUB 900:PR
INT CN$(CN)
 1192 X(PL)=X(PL)+2:Y(PL)=Y(PL)+1
 1195 RETURN
 1200 REM
 1210 GOSUB 1050:GOSUB 1250:RETURN
 1250 REM
 1255 TX=X(PL):TY=Y(PL)+1:GOSUB 900
 1260 FOR I=1TO9:PRINT TAB(TX,I);BR$;BK$
;BR$:NEXT I
 1270 X(PL)=X(PL)+2:Y(PL)=Y(PL)+1
 1280 RETURN
 1300 REM
 1310 CN=DK(DP,1):SU=DK(DP,2)
 1320 DP=DP+1:IF DP>52 THEN DP=1
 1330 IF FL=1 THEN GOSUB 1200:RETURN
 1335 GOSUB 1000:RETURN
 2010 DATA A,2,3,4,5,6,7,8,9,T,J,Q,K
 2110 DATA"00000000001000000000"
 2120 DATA"00001000000000001000"
 2130 DATA"00001000001000001000"
 2140 DATA"00010100000000101000"
 2150 DATA"00010100001000101000"
 2160 DATA"00010100010100101000"
 2170 DATA"00010101010100101000"
 2180 DATA"00010101010101010000"
 2190 DATA"10100010101010100101"
 2200 DATA"10101010100010101010"
 2210 DATA"00000000001000000000":REM J
 2220 DATA"00000000001000000000":REM Q
 2230 DATA"00000000001000000000":REM K
```

Den Stapel mischen

```
3000 REM
3005 R=RND(-TIME):DP=1
3007 FOR I=1 TO 52:DK(I,1)=0:NEXT I
3010 FOR I=1 TO 4:FOR J=1 TO 13
3020 EP=INT(RND(1)*52)+1
3030 IF DK(EP,1)=0 THEN 3050
3040 EP=EP+1:IF EP>52 THEN EP=1
3045 GOTO 3030
3050 DK(EP,1)=J:DK(EP,2)=I:NEXT J,I
3060 RETURN
```

Sinclair Spectrum

Hauptprogramm

```
10>REM **** SPECTRUM PONTOON ****
15 PRINT "PLEASE WAIT..."
16 POKE 23658,8
20 GO SUB 500: REM INIT ARRAYS ETC
50 REM **** GAME LOOP ****
55 GO SUB 600: REM INIT GAME
```

Aufbau der Felder

```
500>REM **** INIT ARRAYS ETC ****
510 LET S$="": FOR I=1 TO 25: LET S$=S$
+" ": NEXT I
511 LET S$=S$+S$( TO 14)
512 LET B$="": LET L$="": FOR I=1 TO 7:
LET L$=L$+CHR$ 145: LET B$=B$+CHR$ 144:
NEXT I
513 LET D$=CHR$ 146
515 DIM X(2): DIM Y(2): REM NEXT CARD P
OSITIONS
520 LET U$=CHR$ 147+CHR$ 148+CHR$ 149+C
HR$ 150: REM SUIT TYPES
530 DIM N$(13): FOR I=1 TO 13: READ N$(
I): NEXT I: REM READ NUMBER DATA
540 DIM C$(13,21): FOR I=1 TO 13: READ
C$(I): NEXT I: REM READ PATTERN DATA
560 DIM K(52,2): REM SET UP CARD DECK A
RRAY
570 GO SUB 3000: REM SHUFFLE PACK
580 BORDER 4: PAPER 7: INK 9: CLS
581 REM **** SET UP UDG'S ****
582 FOR L=USR "A" TO USR "L"+7
583 READ US: POKE L,US: NEXT L
590 RETURN
600 REM **** INIT GAME ****
602 LET DB=0: LET CS=0
605 CLS
620 LET X(1)=0: LET Y(1)=0: LET X(2)=14
: LET Y(2)=0
630 RETURN
```

Darstellung der Karten

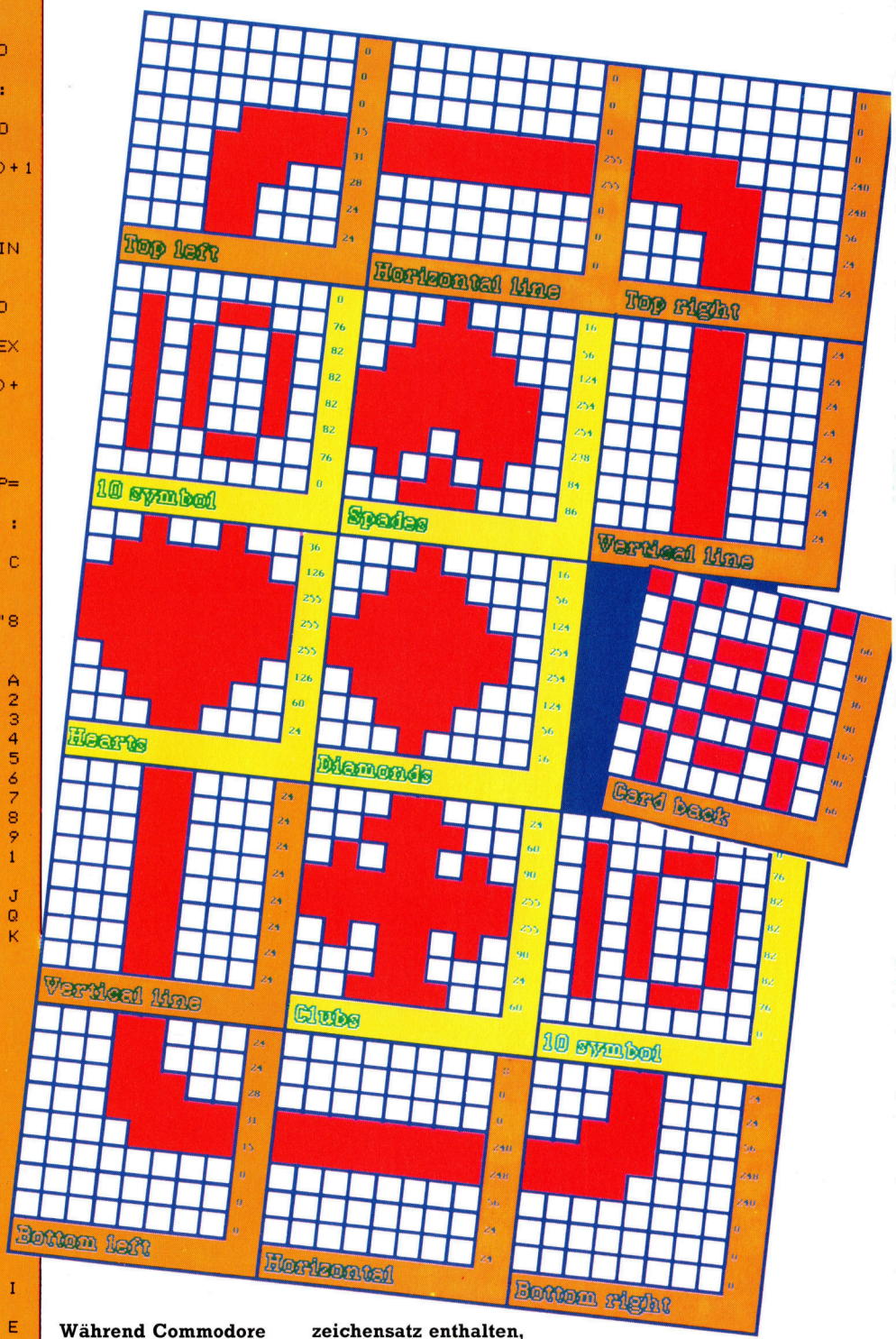
```
 900>REM **** PRINT AT ****
 910 PRINT AT TY,TX;: RETURN
1000 REM **** DISPLAY CARD ****
1010 GO SUB 1050: GO SUB 1100: RETURN
1050 REM **** CARD BLANK ****
1055 LET TX=X(PL)+1: LET TY=Y(PL): GO SU
B 900: REM POSITION
1060 PRINT AT Y(PL),TX;CHR$ 151;L$;CHR$
152
1070 FOR I=1 TO 9: PRINT AT Y(PL)+I,TX;C
HR$ 146;"        ";CHR$ 146: NEXT I
1080 PRINT AT 10+Y(PL),TX;CHR$ 153;L$;CH
R$ 154
1090 RETURN
1100 REM **** CARD DETAIL ****
1120 LET TX=X(PL)+2: LET TY=Y(PL)+2: GO
SUB 900: REM POSITION
1125 LET T$=U$(SU): REM SELECT SUIT TYPE
```

```
1127 INK 0: IF SU>2 THEN  INK 2: REM SEL
ECT COLOUR
1130 FOR I=1 TO 19 STEP 3
1140 LET Z$=C$(CN)(I TO I+2)
1142 FOR J=1 TO 3: LET W$="   "
1143 PRINT " ";
1144 IF Z$(J)="1" THEN  PRINT T$;
1145 IF Z$(J)="0" THEN  PRINT " ";
1146 NEXT J
1150 PRINT AT ((I+3)/3)+TY,X(PL)+2;: NEX
T I
1160 REM **** ADD CORNER LABLES ****
1170 LET TX=X(PL)+2: LET TY=Y(PL)+1: GO
SUB 900: PRINT N$(CN): REM POSITION
1180 LET TY=TY+1: GO SUB 900: PRINT T$:
REM SUIT
1190 LET TX=X(PL)+8: LET TY=Y(PL)+9: GO
SUB 900: PRINT N$(CN): REM POSITION
1192 LET X(PL)=X(PL)+2: LET Y(PL)=Y(PL)+1
1193 INK 0
1195 RETURN
1200 REM **** DISPLAY CARD BACK ****
1210 GO SUB 1050: INK 1: GO SUB 1250: IN
K 0: RETURN
1250 REM **** CARD BACK ****
1255 LET TX=X(PL)+2: LET TY=Y(PL)+1: GO
SUB 900: REM POSITION
1260 FOR I=1 TO 9: PRINT AT I,TX;B$: NEX
T I
1270 LET X(PL)=X(PL)+2: LET Y(PL)=Y(PL)+
1
1280 RETURN
1300 REM **** DEAL A CARD ****
1310 LET CN=K(DP,1): LET SU=K(DP,2)
1320 LET DP=DP+1: IF DP>52 THEN  LET DP=
1: REM BUMP DECK
1330 IF FL=1 THEN  GO SUB 1200: RETURN :
 REM DISPLAY BACK OF CARD
1335 GO SUB 1000: RETURN : REM DISPLAY C
ARD
2000 REM **** CARD NUMBER DATA ****
2010 DATA "A","2","3","4","5","6","7","8
","9",CHR$ 155,"J","Q","K"
2100 REM **** CARD DISPLAY DATA ****
2110 DATA "00000000001000000000": REM A
2120 DATA "00001000000000010000": REM 2
2130 DATA "00001000001000010000": REM 3
2140 DATA "00010100000000101000": REM 4
2150 DATA "00010100001000101000": REM 5
2160 DATA "00010100010100101000": REM 6
2170 DATA "00010101010100101000": REM 7
2180 DATA "00010101010101101000": REM 8
2190 DATA "10100010101010100101": REM 9
2200 DATA "10101010100010101010": REM 1
0
2210 DATA "00000000001000000000": REM J
2220 DATA "00000000001000000000": REM Q
2230 DATA "00000000001000000000": REM K
2240 REM **** UDG DATA ****
2250 DATA 165,66,90,36,90,165,90,66
2251 DATA 0,0,0,255,255,0,0,0
2252 DATA 24,24,24,24,24,24,24,24
2253 DATA 16,56,124,254,254,238,84,56
2254 DATA 24,60,90,255,255,90,24,60
2255 DATA 16,56,124,254,254,124,56,16
2256 DATA 36,126,255,255,255,126,60,24
2257 DATA 0,0,0,15,31,28,24,24
2258 DATA 0,0,0,240,248,56,24,24
2259 DATA 24,24,28,31,15,0,0,0
2260 DATA 24,24,56,248,240,0,0,0
2261 DATA 0,76,82,82,82,82,76,0
```

Den Stapel mischen

```
3000>REM **** SHUFFLE THE DECK ****
3005 RANDOMIZE : LET DP=1
3007 FOR I=1 TO 52: LET K(I,1)=0: NEXT I
3010 FOR I=1 TO 4: FOR J=1 TO 13
3020 LET EP=INT (RND*52)+1: REM SELECT E
NTRY POINT
3030 IF K(EP,1)=0 THEN  GO TO 3050
3040 LET EP=EP+1: IF EP>52 THEN  LET EP=
1
3045 GO TO 3030
3050 LET K(EP,1)=J: LET K(EP,2)=I: NEXT
J: NEXT I
3060 RETURN
```

All diese Versionen des Kartenspiels zeichnen die Umrandungen der Spielkarten mit Grafikzeichen. Für den Acorn B und den Spectrum müssen wir diese Zeichen zunächst definieren. Im Acorn-Programm nutzen wir hierfür VDU 23-Kommandos in den Zeilen 572 bis 583. Die Spectrum-Version enthält die Karten-Daten als DATA-Zeilen (2240 bis 2261) und liest sie während der Initialisierung (Zeile 581 bis 583) mit der READ-Anweisung ein.



Während Commodore 64 und Schneider die für die Umrandung der Karten benötigten Grafikzeichen im Standardzeichensatz enthalten, müssen wir diese Zeichen auf dem Acorn B und dem Spectrum selbst definieren.

Ahnengalerie

Bei Mehrplatz-Betriebssystemen wie Unix müssen Dateiverwaltung und Inhaltsverzeichnisse sehr leistungsfähig sein. Wir analysieren die Baumstruktur und sehen uns einige wichtige Befehle an.

In der letzten Folge wurde erklärt, daß bei dem Unix-Mehrplatzsystem jeder Anwender über einen eigenen Speicherbereich (Directory) verfügt, dessen Programme und Daten durch ein Schlüsselwortsystem vor fremdem Zugriff geschützt sind. Mehrplatzsysteme besitzen weiterhin ein „öffentliches Directory" mit Systemprogrammen, die jedem Anwender zur Verfügung stehen. Außerdem gibt es sogenannte „Systemdirectories" mit Informationen über die Zugriffsberechtigungen der Anwender und ihre Directories. Schließlich müssen Mehrplatzsysteme die Fähigkeit haben, mehr als ein Directory gleichzeitig bearbeiten zu können, da sich die Anwender sonst gegenseitig stören.

Aus diesen Gründen werden Mehrplatzsysteme für eine große Zahl Directories ausgelegt. Jeder Anwender soll sich im eigenen Directory frei bewegen können und Zugang zu den Programmen des öffentlichen Directory haben. Nur wenige dürfen diese Programme jedoch auch verändern können.

Es muß möglich sein, sich Einblick in andere Directories zu verschaffen und Dateien von einem Inhaltsverzeichnis zum anderen zu kopieren (auch ohne sie verändern zu können). In der Directorystruktur von Unix können Anwender neue Verzeichnisse anlegen und auch Directories löschen. Es gibt Befehle für den Zugang zu anderen Directories und auch für das Übertragen von Dateien.

Unix-Directories weisen eine Baumstruktur auf, die zur Organisation von Daten mit komplexen Querbeziehungen dient. Jedes Inhaltsverzeichnis kann Dateien und eine Reihe von Sub-Directories enthalten. Obwohl Unix Dateien und Sub-Directories oft völlig gleich behandelt, bringt es die beiden Typen nie durcheinander. Jedes Sub-Directory ist „Kind" des „Mutter"-Directory, in dem sein Name eingetragen ist. Jedes Verzeichnis besitzt daher genau ein Mutter-Directory und kann eine unbegrenzte Zahl von „Kindern" haben.

Wenn wir die Kette aufwärts verfolgen, erreichen wir schließlich ein Directory ohne Mutter. Es wird „Wurzel" (Root) genannt und mit / bezeichnet. Jedes andere Directory läßt sich über seinen „Pfadnamen" ansprechen. Ein Pfadname besteht aus einer Liste von Directories, die bei der Wurzel anfangen, und deren Elemente durch / getrennt werden.

Namensfreiheit

Unser Bild zeigt ein vereinfachtes Directorysystem von Unix. Der vollständige Name des Directory „John" lautet /usr/dept1/John. Auch Dateinamen bestehen aus dem Pfadnamen ihres Directory gefolgt von /Dateiname. Zum Abruf einer bestimmten Datei muß normalerweise jedoch nicht der gesamte Pfadname angegeben werden. Er wird oft abgekürzt oder kann – im eigenen Directory – auch völlig weggelassen werden.

Der Befehl pwd (Print Working Directory) zeigt den Namen des aktuellen Directory.

Unix bietet bei der Wahl der Datei- und Directorynamen mehr Freiheit als andere Betriebssysteme. Die Namen können bis zu 14

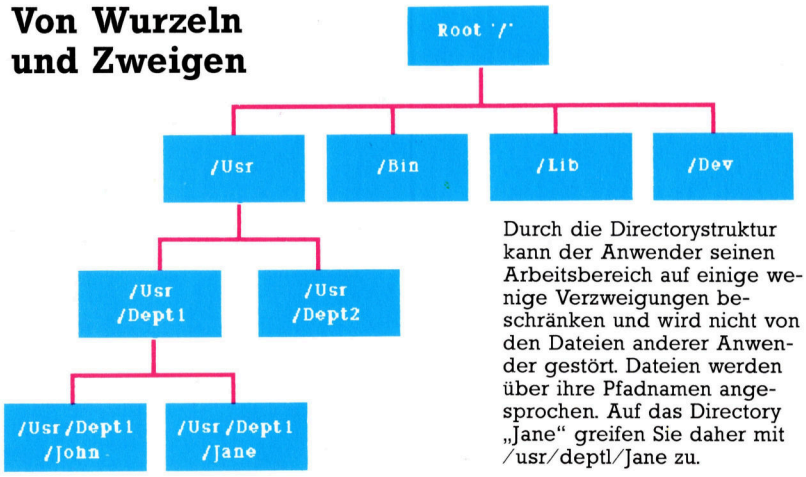
Optionen für Listen

Der Befehl ls hat folgende Optionen:

- **–a** Listet alle Einträge, auch Systemdateien
- **–c** Listet Dateien in Reihenfolge ihrer Anlage
- **–l** Vollständiges Listing
- **–m** Stromausgabe – durch Kommas getrennt
- **–r** Umgekehrte Reihenfolge
- **–s** Dateilänge in Blöcken
- **–F** Markiert Directories mit / und ausführbare Programme mit *
- **–R** Listet auch Inhalt aller Sub-Directories

Die Optionen lassen sich kombinieren (–a und –r etwa zu –ar).

Von Wurzeln und Zweigen



Durch die Directorystruktur kann der Anwender seinen Arbeitsbereich auf einige wenige Verzweigungen beschränken und wird nicht von den Dateien anderer Anwender gestört. Dateien werden über ihre Pfadnamen angesprochen. Auf das Directory „Jane" greifen Sie daher mit /usr/deptl/Jane zu.

Zeichen enthalten, darunter auch Leerzeichen. Sie sollten jedoch Zeichen vermeiden, die Unix für Spezialzwecke reserviert hat:

/\"'*;—[]()~{}<>?$!

Unix kann mit sogenannten „Wildcards" auch ganze Gruppen von Datei- oder Directorynamen ansprechen. Hier die Wildcard-Zeichen:

? spricht ein beliebiges einzelnes Zeichen an.

* spricht eine Zeichengruppe an, schließt aber den Punkt als erstes Zeichen eines Namens aus, um damit Systemdateien wie .login oder .cshrc vor Löschen zu schützen.

[] (mit einer Liste von Zeichen im Inneren der Klammer) spricht ein einzelnes Zeichen an, das im Klammerinneren aufgeführt ist:

[a,e,f],[A—Z], etc.

Der Befehl ls zeigt Dateien und Sub-Directories eines Directory an. Wie viele andere Unix-Befehle besitzt er eine Reihe von Optionen (siehe Kasten). Zwei weitere Befehle listen den Inhalt einer Datei: cat (kann Dateien auch zusammenfügen) und more (stellt den Dateiinhalt bildschirmweise dar).

Der Anwender kann seine eigenen Sub-Directories beliebig anlegen und auch wieder löschen. So richtet mkdir ein neues Sub-Directory des aktuellen Directory ein, und rmdir löscht es, wenn darin keine Dateien oder Sub-Directories mehr vorhanden sind.

Das aktuelle Directory kann mit cd gewechselt werden. Wenn Sie in ein Sub-Directory des alten Directory gehen, geben Sie nur den neuen Namen an, bei jedem anderen Directory brauchen Sie jedoch den vollen Pfadnamen. cd ohne Directorynamen führt zu dem Directory zurück, in dem Sie sich angemeldet hatten (das „Home-Directory").

Der untenstehende Kasten enthält einen Unix-Dialog, in dem einige der aufgeführten Befehle eingesetzt sind. Die Kommentare (in geschweiften Klammern) erscheinen in dem eigentlichen Dialog natürlich nicht.

Dialog mit Unix

```
Berkeley 4.2 Vax/Unix (infsc3)
Type <Ctrl-D> to disconnect

login:com-mcc
Password:
{Das eingegebene Passwort wird nicht auf dem Bildschirm angezeigt}
You are a Normal user (Class 3)
Jobs : 19  Superiors : 2  Maximum : 21
Last login: Fri Oct 18 11:45:37 on ttyn05

          Welcome to the Information Sciences VAX/UNIX System.

%pwd
/mnt/com/com-mcc          {dies ist das „Home-Directory"}
%cd fred                  {Wechsel in ein Sub-Directory}
%ls                       {Alle Dateien dieses Directory listen}
rec.c   receive   rx.p   transmit   tx.p
%ls -a
      rec.c   receive   rx.p   transmit   tx.p

{Die beiden zusätzlichen Namen (. und ..) bezeichnen dieses Directory und das Mutter-Directory}

%ls -l
total 42
-rw-rw-r--  1 com-mcc     502   Sep 17 12:07 rec.c
-rwxr-xr-x  1 com-mcc   18432   Oct 21 11:02 receive
-rw-r--r--  1 com-mcc    1068   Oct 18 14:44 rx.p
-rwxr-xr-x  1 com-mcc   19456   Oct 21 11:03 transmit
-rw-r--r--  1 com-mcc    1244   Oct 21 11:01 tx.p

{Die „rwx" Gruppen zeigen die Zugangsrechte des Directory-inhabers (1), anderen Mitgliedern der Anwendergruppe (2) und allen anderen Benutzern (3) an. r bedeutet – Lesen ist möglich; w bedeutet – Schreiben ist möglich, und x bedeutet – Ausführung ist möglich. Ein Strich besagt, daß der Zugang dieses Typs nicht möglich ist.}

%ls ?x.p                  {Einsatz von „Wildcard"-Zeichen}
rx.p   tx.p
%ls r*
rec.c   receive   rx.p
%ls *.[pc]
rec.c   rx.p   tx.p

%mkdir mike               {Neues Sub-Directory anlegen}
%ls -l
total 43
drwxr-xr-x  2 com-mcc      24   Oct 21 11:10 mike
-rw-rw-r--  1 com-mcc     502   Sep 17 12:07 rec.c
-rwxr-xr-x  1 com-mcc   19456   Oct 21 11:03 transmit
-rw-r--r--  1 com-mcc    1244   Oct 21 11:01 tx.p

{Das 'd' zeigt das neue Sub-Directory an}

%cd mike                  {Vom aktuellen Directory in das neue wechseln}
%pwd
/mnt/com/com-mcc/fred/mike
%cd ..                    {Zurück zum Mutter-Directory}
-rwxr-xr-x  1 com-mcc   18432   Oct 21 11:02 receive
-rw-r--r--  1 com-mcc    1068   Oct 18 14:44 rx.p
%pwd
/mnt/com/com-mcc/fred
%cp rx.p mike             {die Datei "rx.p" in das neue Directory kopieren}
%mv tx.p mike             {die Datei "tx.p" in das neue Directory stellen}
%ls                       {"tx.p" wurde aus diesem Directory genommen}
mike   rec.c   receive   rx.p   transmit
%cd mike
%ls
rx.p   tx.p
%who                      {Wer arbeitet sonst noch mit dem System?}
root              console Oct 21 08:07
com-rgd           ttyn00  Oct 21 09:45
ccs-klf           ttyn01  Oct 21 09:45
cs4bc             ttyn04  Oct 21 10:57
com-mcc           ttyn05  Oct 21 10:58
ccs-mdb           ttyn06  Oct 21 10:06
cs4cy             ttyn07  Oct 21 11:06
com-ah            ttyn08  Oct 21 11:00
com-jh1           ttyn10  Oct 21 11:05
cs4bg             ttyn11  Oct 21 11:05
%finger com-mcc           {Information über den anderen Anwender}
Login name : com-mcc    Real name : curtis
Department : Not known
Directory : /mt/com/com-mcc
Shell : /bin/csh
Logged in at 10:58 on Mon Oct 21 on ttyn05   36 secs idle   No Plan.
%logout                   {Dialog beenden}
Host sent disconnect
```

MIDI in — MIDI out

Wir vervollständigen unsere MIDI-Schnittstelle für den Schneider CPC mit der dafür maßgeschneiderten Steuer-Software. Unser Maschinenprogramm nutzt die im Schneider-Betriebssystem vorgesehenen Interrupts.

Der kleinste Zeitabstand zwischen zwei Midi-Meldungen beträgt 320 µs. Daher muß die Verarbeitungsschleife für die eingehenden Daten sehr viel schneller durchlaufen werden, damit der Rechner die Signale verarbeiten und trotzdem rechtzeitig für das nächste Byte empfangsbereit sein kann.

Ein gutes MIDI-Programm sollte die eingehenden Daten intensiv bearbeiten können. Dazu würde etwa die Fähigkeit gehören, gleichzeitig Daten zu senden und zu empfangen, sie auf dem Bildschirm darzustellen und auch Zusammenschnitte aus mehreren Kanälen zu erzeugen. Ein Hindernis bleibt immer die kurze Zeitspanne von 320 µs. Zum Glück gibt es aber ein Verfahren, mit dem der Rechner auf eingehende Daten „horcht", gleichzeitig aber andere Daten bearbeitet.

Um das Prinzip eines solchen Programms zu verdeutlichen, möchten wir Ihnen eine Routine vorstellen, die MIDI-Daten einliest und sie gleichzeitig auf dem Bildschirm darstellt. Die Daten werden in hexadezimaler Darstellung zeilenweise angezeigt. Dazu muß jedes eingehende MIDI-Byte in zwei Hexadezimalziffern im ASCII-Code umgewandelt werden. Diese Ziffern werden dann zum Bildschirm geschickt und mit zwei Leerzeichen abgeschlossen, um sie vom nächsten Byte unterscheidbar zu machen. An das letzte Byte einer abgeschlossenen MIDI-Message muß ein Return angehängt werden. Betriebssystemroutinen wie etwa die Tastaturabfrage wirken sich bei einer Übertragungsrate von 320 µs bereits störend aus: Es ist möglich, daß die gewünschte Verarbeitung länger dauert, als erlaubt.

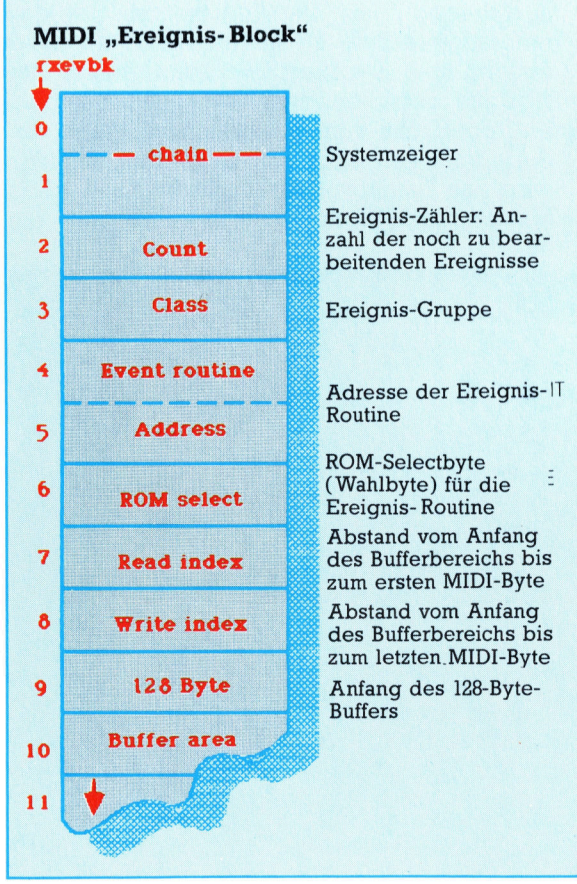
Datensicherheit

Um zu verhindern, daß Daten verlorengehen, müssen wir unsere Verarbeitungsschleife regelmäßig mit einem Interrupt ablösen, der das ACIA-Statusregister abfragt und alle neu empfangenen Daten in einen Bufferbereich schafft. Das Programm im Vordergrund kann diese Daten später wieder abholen und verarbeiten, ohne daß Signale übersehen werden. Der Interrupt-Abstand müßte allerdings kleiner als 320 µs sein, was das Programm nur unnötig komplizieren würde.

Als Alternative zu einem periodischen Interrupt können wir auch die Fähigkeit des ACIA-ICs nutzen, eigene Interrupts zu erzeugen. Das geschieht immer dann, wenn das Empfangsdatenregister voll ist. Dazu muß natürlich die interne Interrupt-Steuerung des Schneider abgeschaltet werden, um die ACIA-Interrupts nicht mit anderen Interrupts zu verwechseln. Bei jedem ACIA-Interrupt müßte der Inhalt des Empfangs-Datenregisters (Receive Data register) übernommen werden.

Unser Programm nutzt dazu die Firmware-Routinen des Schneider. Der zum Verarbeiten des ACIA-Interrupts nötige Sprungbefehl wird in den Speicherplatz &003B gepatcht. Der Interrupt-Handler selbst kommuniziert mit dem Vordergrund-Programm durch „kicken" (siehe Bits-und-Bytes-Serie) eines Ablaufs, der durch den Ereignis-Block („Event") an der mit „rxevbk" bezeichneten Adresse definiert ist.

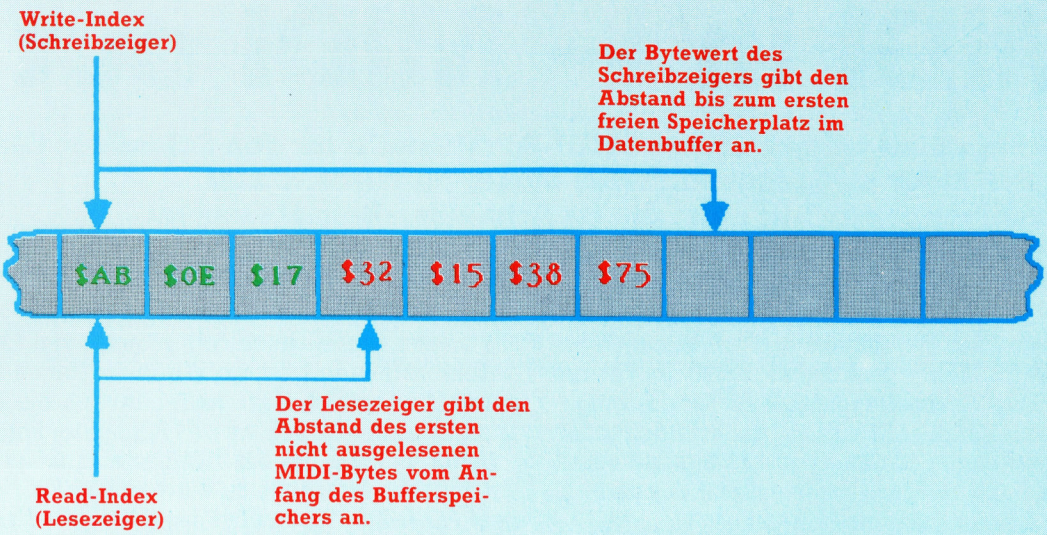
Der Buffer-Bereich umfaßt 128 Bytes im Speicher, die bei Adresse rxevbk+9 beginnen. Die



Mach' mal Pause
Das Schneider-Betriebssystem verfügt über eine Methode („event handling") zur Behandlung von Interrupts. Der ACIA-Chip unserer MIDI-Schnittstelle erzeugt immer dann einen Interrupt, wenn sein Empfangs-Datenregister voll ist. Dieser Interrupt wird vom nebenstehenden Ereignis-Block umgesetzt: Die ersten sechs Bytes haben Systemaufgaben. Byte 7 und 8 geben an, wieviele von den 128 folgenden Bytes mit MIDI-Daten belegt sind. Diese 128 Bytes dienen unserem Programm als Buffer für die empfangenen MIDI-Daten.

Zeiger für den Buffer

Die Zeiger für den Anfang und das Ende der zwischengespeicherten Warteschlange von MIDI-Bytes geben an, wo Daten hingeschrieben bzw. abgeholt werden müssen. Beim Abholen und Hineinschreiben neuer Daten durchwandern die Zeiger kontinuierlich den gesamten Buffer. Wenn der Buffer leer ist (READ- und WRITE-Zeiger haben den gleichen Wert), werden die Indizes wieder auf ihre Anfangswerte zurückgesetzt.



Routine prüft zuerst, ob noch unerledigte Aufgaben anstehen. Falls nicht, ist der Buffer leer, und die Lese- und Schreibindizes werden zurückgesetzt. Als nächstes prüft die Routine auf einen Empfängerüberlauf – das heißt, ob die ACIA ein Byte empfangen hat, bevor das letzte ausgelesen war. Der Interrupt-Befehl selbst muß ebenfalls vom Interrupt-Programm gelöscht werden.

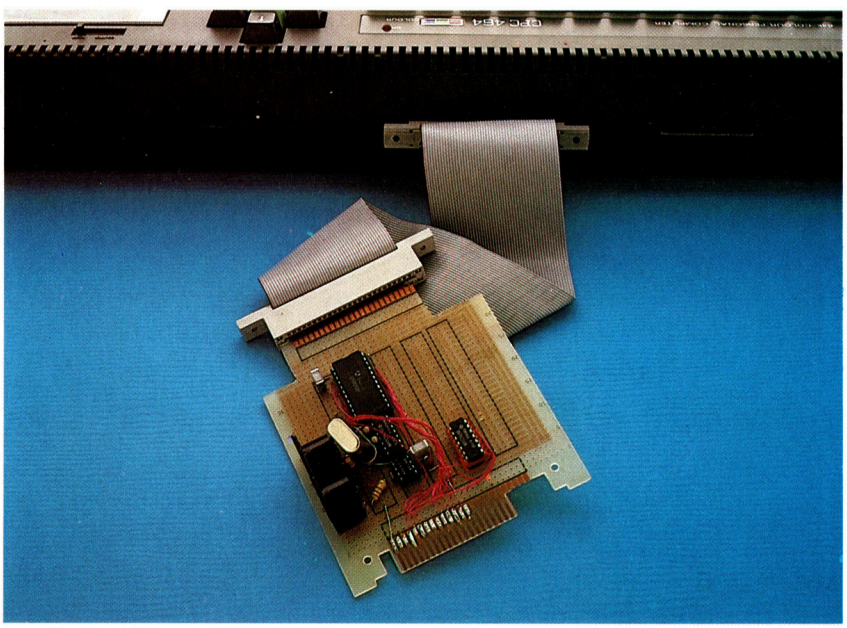
Das ACIA-Interrupt-Signal kann durch Auslesen des ACIA-Datenregisters wieder gelöscht werden. Das Programm ermittelt dann aus dem im Ereignis-Block bei rxevbk+8 gespeicherten Wert die Adresse, an der das empfangene Byte im Bufferbereich abgelegt werden muß. Bei rxevbk+8 steht einfach der Abstand zwischen dem Ende des belegten Bufferbereichs vom Bufferanfang. Wenn der Bufferspeicher nicht voll ist (Index < 128), wird das Datenbyte hineingeschrieben. Das Vordergrund-Programm initialisiert nur die ACIA-Register und den Ereignis-Block, danach springt es in eine Schleife, die Zeichen vom Anfang des Buffers holt und in der oben beschriebenen Weise weiterverarbeitet. Die Speicherplatzadresse für den Anfang des Bufferbereichs steht in Adresse rxevbk+7 des Ereignis-Blocks. Auch dieser Index wird als Abstand vom ersten Bufferbyte gespeichert.

Das Programm bricht bei jedem Tastendruck am Computer sowie bei Überfüllung des Buffers oder beim Überspringen von Daten ab.

Das vorgestellte Programm soll dazu dienen, die Technik unregelmäßiger Interrupts im Schneider-Betriebssystem zu verdeutlichen. Problemlos ist das nicht: Wenn Sie das Programm starten und von einer MIDI-Tastatur Daten schicken, bricht die Routine gelegentlich wegen eines Empfänger-Überlaufs ab: Es wurde ein zweites Datenbyte empfangen, bevor das erste aus dem Empfangsdatenregister abgeholt worden war. Das liegt an der Methode, nach der das Betriebssystem die Interrupts verarbeitet.

Der Schneider nutzt die Zweitregister des Z80 beim Auftauchen eines Interrupts, um dort vorübergehend den aktuellen Zustand der CPU zu speichern. Es gibt aber nur einen Satz von Zweitregistern – mehrere Interrupt-Ebenen sind nicht verfügbar. Während der Abarbeitung eines Interrupts dürfen also keine weiteren Unterbrechungen auftreten. Der Interrupt-Handler muß außerdem mehrmals auf die Ereignis-Struktur der Firmware zugreifen – das kann dazu führen, daß die Interrupts für länger als die bekannten 320 µs abgeschaltet bleiben, so daß ein oder mehrere Bytes verpaßt werden können.

Die beste Lösung ist es, das Interrupt-Steuerprogramm durch ein eigenes, selbstgeschriebenes zu ersetzen. Dieses Programm müßte alle periodischen Interrupts übergehen – dadurch würde allerdings die Zugriffsmöglichkeit auf Systemfunktionen fortfallen.



Die abgewandelte MIDI-Schnittstelle wird über ein 50-poliges Flachkabel mit dem Erweiterungsanschluß des Schneider CPC verbunden. Über 5-polige Standard-DIN-Buchsen lassen sich bis zu 16 MIDI-fähige Instrumente anschließen und per Programm steuern.

Hex-Ladeprogramm:

```
20 org=&408E:loadptr=0
30 WHILE flag=0
40 GOSUB 1000:IF flag=1 THEN 60:REM exit
50 POKE org+loadptr,entry:loadptr=loadptr+1
60 WEND
70 END
1000 REM **** get a hex byte s/r ****
1020 flag=0  LINE INPUT entry$
1025 IF entry$="x" THEN flag=1:RETURN
1030 IF LEN(entry$)<>2 THEN PRINT"error"
:GOTO 1020
1050 entry=VAL("&"+entry$):RETURN
```

Falls Sie keinen Assembler haben, müssen Sie das Maschinenprogramm mit dem hier abgedruckten BASIC-Hex-Ladeprogramm eingeben. Die Bytes der linken Spalte werden dazu bei laufendem Ladeprogramm einzeln eingetippt und am Schluß mit SAVE „FILENAME",B,&4000,&199 auf Diskette oder Cassette gespeichert. Jedes Byte muß mit einem Return (CR) abgeschlossen werden.

MIDI-Programm für Schneider CPC

```
                org  #4000
klev:   defs 2
rxevbk: defs 140          ;event block + read/write
cont:   equ  #f8e0
txreg:  equ  #f9e0
stat:   equ  #fae0
rxreg:  equ  #fbe0
;os call addresses
kmrdch: equ  #bb09
kmarbr: equ  #bb45        ;arm breaks
txtout: equ  #bb5a
klinev: equ  #bcef
kleven: equ  #bcf2
kldisa: equ  #bd0a        ;disarm async event
klpoll: equ  #b921        ;carry returns event
klnext: equ  #bcfb        ;get next sync event
kldosy: equ  #bcfe        ;do sync event
kldone: equ  #bd01        ;done sync event
klsres: equ  #bcf5        ;clear event queue
start:                    ;patch interrupt routine

F3              di
113B00          ld   de,#003b
211441          ld   hl,jpinst
010300          ld   bc,3
EDB0            ldir
FB              ei
        ;initialise receive event
210240          ld   hl,rxevbk
115941          ld   de,rxevrt    ;event routine address
0601            ld   b,1
CDEFBC          call klinev
3600            ld   (hl),0       ;initialise read/write
23              inc  hl
3600            ld   (hl),0
        ;save kl event routine
        addr
ED5BF3BC        ld   de,(kleven+1)
CBBA            res  7,d
CBB2            res  6,d          ;remove rom select bits
ED530040        ld   (klev),de
        ;initialise 6850
01E0F8          ld   bc,cont
3E03            ld   a,3
ED79            out  (c),a
3E96            ld   a,#96
ED79            out  (c),a
        wait:
CDFBBC          call klnext
DC0941          call c,dosync     ;do event if waiting
210A40          ld   hl,rxevbk+8
CB7E            bit  7,(hl)
2005            jr   nz,wait1     ;exit if buffer full
CD09BB          call kmrdch
30EE            jr   nc,wait      ;jump if no key pressed
        wait1:
CB76            bit  6,(hl)
280C            jr   z,wait2      ;skip message if not over
21F340          ld   hl,string
0616            ld   b,strlen
7E      nexc:   ld   a,(hl)
CD5ABB          call txtout
23              inc  hl
10F9            djnz nexc
        wait2:
CDF5BC          call klsres

F3              di
3EC9            ld   a,#c9        ;ret instruction
323B00          ld   (#003b),a
3E16            ld   a,#16
ED79            out  (c),a        ;disable 6850 interrupts
FB              ei
C9              ret               ;exit to editor
0A0D    string: defb #0a,#0d
41434941        defm "ACIA
                overrun error"
0A0D            defb #0a,#0d
        strlen: equ  $-string
        dosync:                   ;perform sync event
E5              push hl           ;save event address
F5              push af           ;save previous priority
CDFEBC          call kldosy       ;perform event
F1              pop  af
E1              pop  hl
CD01BD          call kldone
C9              ret
C31741  jpinst: jp   rcvint
        rcvint:                   ;rcv interrupt routine
3A0440          ld   a,(rxevbk+2)
B7              or   a            ;test for count = zero
2006            jr   nz,rcv1
320940          ld   (rxevbk+7),a ;initialise indices
320A40          ld   (rxevbk+8),a
C5      rcv1:   push bc           ;save old rom state
210A40          ld   hl,rxevbk+8  ;write index address
CB91            res  2,c          ;enable lower rom
ED49            out  (c),c
01E0FA          ld   bc,stat
ED78            in   a,(c)
CB6F            bit  5,a
280A            jr   z,rcv15
36FF            ld   (hl),#ff     ;set overrun flag
3E16            ld   a,#16
05              dec  b
05              dec  b
ED79            out  (c),a
1817            jr   rcv2
04      rcv15:  inc  b
ED78            in   a,(c)
34              inc  (hl)         ;increment index
5E              ld   e,(hl)
CB7B            bit  7,e          ;check for buffer full
200E            jr   nz,rcv2      ;exit if full
1600            ld   d,0
19              add  hl,de        ;get write address
77              ld   (hl),a
210240          ld   hl,rxevbk
ED5B0040        ld   de,(klev)
CD1600          call #0016
C1      rcv2:   pop  bc
ED49            out  (c),c
C9              ret
        rxevrt:                   ;rx event routine hl=even
210940          ld   hl,rxevbk+7
34              inc  (hl)         ;increment index
5E              ld   e,(hl)
1C              inc  e
1600            ld   d,0
19              add  hl,de        ;point to next read data
7E              ld   a,(hl)
FEF8            cp   #f8
D0              ret  nc
B7              or   a
F27641          jp   p,evrt1
47              ld   b,a
3E0A            ld   a,#0a
CD5ABB          call txtout
3E0D            ld   a,#0d
CD5ABB          call txtout
78              ld   a,b
        evrt1:
CD7A41          call hexpr        ;print char
C9              ret
        hexpr:                    ;hexadecimal print
0602            ld   b,2
        hexpr1:
3E00            ld   a,0
ED6F            rld               ;get ms digit
CD9141          call digasc       ;convert to ascii
CD5ABB          call txtout       ;print
10F4            djnz hexpr1
3E20            ld   a," "
CD5ABB          call txtout
CD5ABB          call txtout
C9              ret
        digasc:                   ;hex  digit to ascii
C630            add  a,"0"
FE3A            cp   ":"
D8              ret  c
C607            add  a,"A"-":"
C9              ret
```

Einen Speicherblock in die Datei schreiben

Der folgende Ablauf schreibt einen Speicherblock auf Cassette oder Diskette:

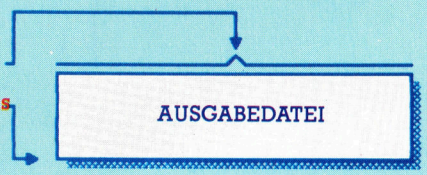
1) Ausgabedatei eröffnen

B = Länge des Dateinamens

HL = Adresse des Dateinamens

DE = Adresse des 2K-Buffers

CALL CAS OUT OPEN



2) Daten schreiben

HL = Adresse der Ausgabedaten

DE = Länge der Daten

BC = Eintragsadresse des Kopfsatzes

A = Dateityp für den Kopfsatz

CALL CAS OUT DIRECT

3) Ausgabedatei schließen

CALL CAS OUT CLOSE

Format des Kopfsatzes

Bei der Eröffnung einer Datei mit CAS IN OPEN wird der Kopfsatz der Datei ins RAM geschrieben und seine Anfangsadresse an HL zurückgegeben. Weiterhin enthält DE die ursprüngliche Speicheradresse der Daten (wie im Kopfsatz eingetragen). In BC ist die logische Dateilänge gespeichert und in A der Dateityp. Die Informationen des Dateityp-Bytes haben bitweise Bedeutung (siehe unten).

Der Dateityp wird von mehreren Feldern definiert:

Bit 0	Bit 1–3	Bit 4–7
Dateischutz	Dateiinhalt	Version
0 = Geschützt 1 = Nicht geschützt	0 = BASIC 2 = Binär 4 = Bildschirm 6 = ASCII 8 = nicht zugeteilt	16 = ASCII 32 nicht zugeteilt

Gut und sicher

Mit dem integrierten Cassettendeck des Schneider CPC 464 und der Diskettenstation des CPC 664 lassen sich Dateien leicht zwischen diesen beiden Maschinen austauschen.

Der Erfolg des Schneider CPC 464 wurde wesentlich von dem integrierten Cassettendeck mitbestimmt, das Daten selbst bei hoher Schreibgeschwindigkeit sicher speichern kann. Der CPC 664 stellte dann statt des Recorders eine Diskettenstation mit noch höherer Speichergeschwindigkeit zur Verfügung. Da aber auch die beste Hardware ohne entsprechende Software wertlos ist, untersuchen wir, welche Firmware-Routinen dem Programmierer dafür zur Verfügung stehen.

Die Firmware-Routinen lassen sich in drei Bereiche unterteilen: die allgemeine Dateiübertragung, die Steuerung des Bandsystems und schließlich die Diskettenroutinen.

Jede Datei enthält zwei separate Informationsbereiche: den Kopfsatz und die Daten. Im Kopfsatz sind alle Informationen gespeichert, die die Firmware zur Bearbeitung der Datei braucht – zum Beispiel der Dateityp und die Dateilänge. Auf den Kopfsatz folgt der eigentliche Dateiinhalt – die Daten.

Wir werden uns zunächst mit dem „Cassettenmanager" beschäftigen. Der Begriff ist etwas irreführend, da die gleichen Routinen auch für Diskettendateien eingesetzt werden, auf die wir später eingehen.

Der Cassettenmanager

Der Cassettenmanager arbeitet mit zwei „Strömen" (für Ein- und Ausgabe), die beide gleichzeitig aktiv sein können. Bei einem Bandgerät läßt sich diese Fähigkeit kaum einsetzen, bei einer Diskettenstation ist sie jedoch außerordentlich praktisch. Vor ihrem Einsatz müssen die Ströme initialisiert werden. Dabei wird dem entsprechenden Strom ein Name und ein Arbeitsbereich (Buffer) zugewiesen. Da nur jeweils eine Ein- und eine Ausgabedatei aktiv sein kann, läßt sich der Ausgabestrom nicht als zweiter Eingabestrom verwenden. Die Datenübertragungsroutinen sind in einer speziellen Tabelle eingetragen.

Daten lassen sich auf zwei Arten zu einer Datei schicken oder von dort empfangen: per Block oder per Zeichen. Bei der Zeichenüber-

tragung benutzt die Firmware den (dem Strom zugeordneten) Buffer als Zwischenspeicher. Das bedeutet, daß die Daten blockweise zum Band (oder zur Diskette) gesandt oder von dort empfangen werden, während das Programm sie byteweise bearbeitet. Natürlich ist ein Programm, das den Speicher zeichenweise anspricht, sehr langsam.

Die beiden Übertragungsarten schließen sich gegenseitig aus. Wenn eine Datei einmal auf eine bestimmte Art angesprochen wurde, kann nicht mehr auf die andere Art umgeschaltet werden.

Bei der Blockübertragung (CAS IN DIRECT und CAS OUT DIRECT) ist der Buffer nicht nötig, da die Daten direkt aus dem Speicher gelesen bzw. dort abgelegt werden.

Die Routinen CAS IN CLOSE und CAS OUT CLOSE sind besonders wichtig, da sie die Firmware informieren, daß der Strom geschlossen wurde und für eine neue Datei zur Verfügung steht. Wenn nach einer Zeichenausgabe kein CAS OUT CLOSE folgt, kann es vorkommen, daß der Buffer-Inhalt nicht auf das Speichermedium übertragen wird. CAS OUT CLOSE zwingt die Firmware, auch teilweise gefüllte Datenblöcke zu senden, und stellt so sicher, daß alle Daten geschrieben werden.

Unser erstes Bild zeigt, wie Datenblöcke in den Speicher geschrieben werden. Zunächst eröffnet CAS OUT OPEN die Datei und reserviert einen Namen und einen Ausgabestrom dafür. Danach werden die Daten mit CAS OUT DIRECT ausgegeben, und die Datei wird mit CAS OUT CLOSE wieder geschlossen.

In den Kopfsätzen speichert der Cassettenmanager alle wichtigen Dateiinformationen. Ein Lesevorgang holt diese Information aus dem Dateianfang, während ein Schreibvorgang den Kopfsatz in den Buffer schreibt und dann auf den Dateianfang überträgt.

Auf der Firmware-Ebene hat der Parameter **Dateityp** keine Auswirkung – jede Datei kann unabhängig von ihrem Typ gelesen oder geschrieben werden. Wichtig wird der Typ erst bei einem Programm auf höherer Ebene.

Die **Datenadresse** ist als die Speicherstelle definiert, von der aus die Datei ursprünglich geholt wurde. Dies hat jedoch nur dann Bedeutung, wenn die Datei direkt aus dem Speicher

Routinen zur Datenspeicherung

Cassetten-System:

$BC65 CAS INITIALISE
Stellt sicher, daß alle Standardeinstellungen gesetzt sind (Ein- und Ausgabeströme geschlossen, die Meldungsroutine eingeschaltet).

$BC6B CAS NOISY
Schaltet vom Cassettenmanager gelieferte Meldungen ein/aus.

$BC77 CAS IN OPEN
Initialisiert eine Datei zum Lesen

$BC80 CAS IN CHAR
Liest einzelnes Zeichen aus Eingabedatei

$BC83 CAS IN DIRECT
Liest ganze Datei in den Speicher

$BC89 CAS TEST EOF
Prüft auf das Ende einer Eingabedatei

$BC7A CAS IN CLOSE
Schließt die aktuelle Eingabedatei

$BC7D CAS IN ABANDON
Schließt aktuelle Eingabedatei, selbst wenn sie noch nicht vollständig geladen wurde.

$BC8C CAS OUT OPEN
Initialisiert eine Datei zum Schreiben

$BC95 CAS OUT CHAR
Sendet einzelnes Zeichen zur Ausgabedatei

$BC98 CAS OUT DIRECT
Schreibt Speicherblock in die Ausgabedatei

$BC8F CAS OUT CLOSE
Schließt Datei nach Ende des Schreibvorganges

$BC92 CAS OUT ABANDON
Schließt die Datei und ignoriert alle nicht geschriebenen Daten

$BC9B CAS CATALOG
Legt einen Dateikatalog an

Disketten – System:

$BE80 SET MESSAGE
Meldungen der Diskettenroutine ein- und ausschalten

$BE83 SETUP DISK
Initialisiert die Zeitparameter des Laufwerks (Schrittgeschwindigkeit, Verzögerung von Load/unload und Timeout Ein/Aus)

$BE86 SELECT FORMAT
Diskettenformat wählen

$BE89 READ SECTOR
Liest einen Diskettensektor

$BE8C WRITE SECTOR
Schreibt einen Diskettensektor

$BE8F FORMAT TRACK
Formatiert eine Spur

$BE92 MOVE TRACK
Stellt Schreib-Lesekopf auf angegebene Spur

$BE95 GET DR STATUS
Liefert ein Byte, das den Status des aktuellen Laufwerks anzeigt

$BE98 SET RETRY COUNT
Setzt die Zahl der Wiederholungsversuche bei Fehlern.

Cassetten- und Diskettenabläufe werden mit Firmware-Routinen ausgeführt. Ein- und Ausgabedateien können gleichzeitig eröffnet sein. Die Verwaltungsbefehle von Cassetten- und Diskettendateien sind nahezu identisch. Dateien lassen sich direkt aus dem RAM abrufen und dort speichern (CAS IN/OUT DIRECT) oder zeichenweise über einen 2K-Buffer (CAS IN/OUT CHAR). Nach Eröffnung einer Datei kann die Ein- und Ausgabeart nicht mehr verändert werden.

genommen wird – bei der Zeichenausgabe steht hier die Adresse des Ausgabespeichers. **Logische Länge** ist die Gesamtzahl der Datenbytes einer Datei, ohne Kopfsatz oder Zeichen, die zum Füllen des letzten Datenblocks eingesetzt wurden.

Einsprungadresse ist eine Speicherstelle, von der aus Maschinencodeprogramme vom BASIC aus automatisch gestartet werden. Die anderen Felder beziehen sich auf die Bearbeitung von Cassettendateien.

Die Firmware führt automatisch eine Fehlerprüfung durch, die mit einem CRC-Code arbeitet. Dank der umsichtigen Systemprogrammierung sind Fehler jedoch selten. Die Fehler werden von der Firmware angezeigt.

Um Lesefehler leichter abfangen zu können, werden Diskettendateien nicht als Ganzes geschrieben, sondern in 2K-Blöcken. Da jeder Block als separate Einheit behandelt wird, lassen sich fehlerhafte Blöcke leicht nochmals laden, ohne daß fehlerlose Blöcke wiederholt werden müssen. Die Blockstruktur verlangsamt zwar die Übertragung, macht die Speicherung aber sehr sicher. Die Schreibgeschwindigkeit auf Cassette läßt sich per Software über CAS SET SPEED (bei $BC68) einstellen, die Lesegeschwindigkeit wird von der Firmware automatisch festgesetzt.

Transparente Befehle

Die transparent angelegten Diskettenbefehle sind eine der praktischsten Einrichtungen des Betriebssystems. Da dadurch alle allgemeinen Dateiroutinen mit den gleichen Parametern arbeiten, laufen für Cassette geschriebene Programme ohne Änderung auch mit Disketten. Dies beruht auf der Technik des „Jumpblock-Patchens", die wir bereits beschrieben haben. Hier einige Hinweise, die das Verständnis der Patch-Technik erleichtern.

Speziell der Einsatz des Speichers will genau überlegt sein. Das Diskettensystem wird zwar von einem Zusatz-ROM gesteuert, braucht aber auch einen gewissen Arbeitsbereich im RAM. Dieser Platz liegt normalerweise oben im Speicher, kann aber auch anderen RAM-Bereichen zugeordnet werden. Bei Programmen, die mit Disketten arbeiten, ist darauf zu achten, daß dieser RAM-Bereich ($504H unter HIMEM) nicht verfügbar ist.

Weiterhin dürfen Dateinamen wegen des CP/M-Formats nicht länger als zwölf Zeichen sein. Wenn Sie sich auf acht Zeichen beschränken, vermeiden Sie Probleme, die bei der Übertragung von Cassette auf Diskette entstehen können.

Die Kopfsätze der Diskettendateien werden anders behandelt als die der Cassettendateien: Die Blockinformationen haben hier keine Bedeutung und werden ignoriert. Da der Kopfsatz mit einer Prüfsumme versehen ist, kann die Firmware außerdem leicht zwischen CP/M- und AMSDOS-Dateien unterscheiden.

Unser Beispielprogramm zeigt, wie Informationen aus dem Kopfsatz einer Datei herausgezogen werden. Die Datei wird zunächst mit dem Aufruf von CAS IN OPEN eröffnet, der automatisch im RAM einen Kopfsatz anlegt. Nachdem die entsprechende Information im Speicher ist, wird die Datei vor dem Lesen mit CAS IN ABANDON abgeschaltet. Das BASIC-Programm dient als Anwenderschnittstelle für den Maschinencode.

```
10 'File Header Reader
20 'Uses CAS IN OPEN at BC77H
30 routine=&8800: address=&8000: ON ERROR GOTO 530
40 MODE 1: MEMORY &5FFF
50 WHILE -1
60 CLS
70 LOCATE 10,10: PRINT "Tape or disc file? ";CHR$(18)
80 a$=""
90 WHILE a$<>"T" AND a$<>"D"
100 a$=UPPER$(INKEY$)
110 WEND
120 IF a$="T" THEN ITAPE ELSE IF a$="D" THEN IDISC
130 LOCATE 10,12: INPUT "Filename";file$
140 IF LEN(file$)=0 AND a$="D" THEN 130
150 GOSUB 420
160 CALL routine
170 buffer=PEEK(address)+256*PEEK(address+1)
180 LOCATE 20,12: ff=0
190 FOR name=0 TO 7-8*(t$="T")
200 char=PEEK(buffer+name)
210 IF (char<32 OR char>126) AND char>0  THEN  ff=1: GO
TO 230
220 PRINT CHR$(char);
230 NEXT
240 IF ff=1 THEN PRINT CHR$(8);"No filename";
250 PRINT
260 'test filetype
270 LOCATE 5,16: PRINT "Filetype: ": LOCATE 20,16
280 type=PEEK(buffer+18)
290 IF (type AND 1)=1 THEN PRINT "Protected ";
300 file = type AND &FE
310 IF file=0 THEN PRINT "Basic"
320 IF file=2 THEN PRINT "Binary"
330 IF file=6 THEN PRINT "ASCII"
340 IF file>6 THEN PRINT "Unknown"
350 length=PEEK(buffer+24)+256*PEEK(buffer+25)
360 LOCATE 5,18: PRINT "Data length:":LOCATE 20,18: PRI
NT HEX$(length,4)
370 entry=PEEK(buffer+26)+256*PEEK(buffer+27)
380 LOCATE 5,20: PRINT "Entry address:": LOCATE 20,20:
PRINT HEX$(entry,4)
390 LOCATE 9,25:PRINT "Press a key to continue"
400 WHILE INKEY$="":WEND
410 WEND
420 RESTORE 520: READ byte: off=0
430 WHILE byte<>999
440 POKE routine+off,byte
450 off=off+1:READ byte
460 WEND
470 POKE routine+1,LEN(file$)
480 FOR char=1 TO LEN(file$)
490 POKE routine+&100+char-1,ASC(MID$(file$,char,1))
500 NEXT
510 RETURN
520 DATA &06,&00,&21,&00,&89,&11,&00,&60,&cd,&77,&bc,&7
d,&32,&0,&80,&7c,&32,&01,&80,&cd,&7d,&bc,&c9,999
530 'error trap
540 IF ERL=80 THEN a$="":RESTORE
550 STOP
```

Funktionsfähig

Wir zeigen, wie MS-DOS-Funktionen von der Assemblersprache des 8088 und von Hochsprachen wie PASCAL oder FORTRAN aus angesprochen werden.

Die Funktionsaufrufe von MS-DOS lassen sich in sechs Gruppen unterteilen:

- Die Ein- und Ausgabe von Zeichen an angeschlossene Standardgeräte
- Dateizugriff (darunter auch die Verwaltung des Inhaltsverzeichnisses)
- Speicherverwaltung
- Ablaufsteuerung
- Vermischte Systemfunktionen
- Spezielle Aufrufe für Microsoft-Netzwerke

Diese Systemroutinen können von jeder Anwendung aus aufgerufen werden und bieten eine einheitliche und umfassende Schnittstelle, die die unterschiedlichen Betriebssystemversionen untereinander kompatibel macht. Aufrufe, die mit absoluten Adressen arbeiten oder Hardwarekomponenten direkt ansprechen, lassen sich nicht immer auf andere MS-DOS-Systeme übertragen und laufen auch nur höchst selten mit verschiedenen MS-DOS-Versionen.

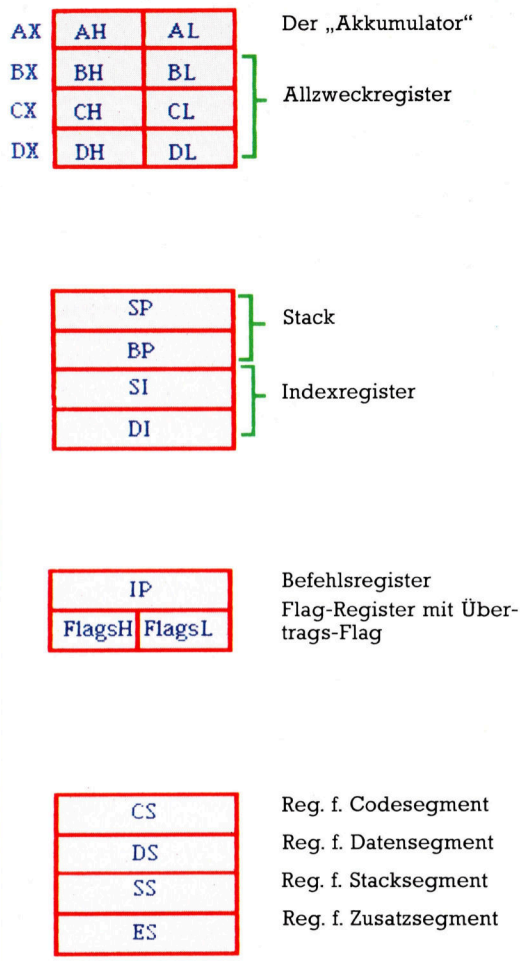
Der Begriff „Funktion", mit der in der MS-DOS-Terminologie Systemaufrufe bezeichnet werden, läßt sich über Unix auf BCPL und C zurückverfolgen. Damit sind jedoch keine Routinen gemeint, die Daten bearbeitet zurückgeben, sondern Vorgänge, die etwa Dateien eröffnen. Derartige Vorgänge sind zwar Prozeduren, werden bei MS-DOS aber mißverständlich Funktionen genannt.

Die größte Bandbreite haben die vermischten Systemfunktionen. Sie arbeiten mit Softwareinterrupts und durchlaufen der Reihenfolge nach folgende Stadien:

1) Alle benötigten Daten in die entsprechenden Register laden (die Laufwerksnummer wird beispielsweise in DL gebraucht).
2) Die Funktionsnummer in AH (das höherwertige Byte von AX) laden.
3) Den Softwareinterrupt 21H auslösen.

Alle von der Funktion zurückgelieferten Daten befinden sich nach der Rückkehr in den Registern des 8086. Die DOS-Funktion 19H (die Funktionsnummern haben Hexadezimalformat) stellt beispielsweise den Code des aktiven Laufwerks in das Register AL. Die Funktion benötigt keinen zusätzlichen Code und läßt sich daher leicht aufrufen:

```
mov ah,19H ;aktuelles Laufwerk feststellen
int 21H    ;Funktion aufrufen
```



Register im Aufruf

Der 8088/8086-Prozessor kann bis zu einem Megabyte adressieren. Er benötigt dafür 20-Bit-Adressen und – da der 8088 nur mit 16-Bit-Adressen arbeitet – zusätzliche Bits zum Aufbau der Adressen. Die Konstrukteure des 8088/8086 umgingen dieses Problem, indem sie das Segment „Offset Format" wählten. Dabei adressiert das Segmentregister einen 64 KByte großen Speicherblock, während das Offsetregister das Byte angibt.

Die wichtigsten Register sind A, B, C, D und das Flag-Register. Sie werden entweder mit dem Zusatz ‚X' als vollständige 16-Bit-Worte angesprochen (z. B. AX) oder als nieder- und höherwertige Bytes mit den Zusätzen ‚L' und ‚H'.

Das Ergebnis steht nun im Register AL. Es stellt das Laufwerk durch einen Zahlenwert dar (A=1, B=2 etc.). Wenn Sie daran den ASCII-Code des Zeichens „A" anhängen und die DOS-Funktion 05H aufrufen (den Code von DL als ASCII-Zeichen darstellen), wird das aktuelle Laufwerk als Großbuchstabe dargestellt:

```
add al,'A'  ;0 in 'A', 1 in 'B' etc. wandeln
mov dl,al   ;in das Register DL stellen
mov ak,05H  ;Zeichen darstellen
int 21H     ;Funktion aufrufen
```

Die im vorigen Beispiel eingesetzten mnemotischen Kürzel entsprechen dem Microsoft-Assembler, der von den OEM-Firmen aber nicht immer mitgeliefert wird. Glücklicherweise ist ein Assembler nicht unbedingt nötig, da MS-DOS sich auch von einer Hochsprache aus leicht aufrufen läßt. Wenn Sie viel in Assembler programmieren, sollten Sie beim Kauf sicherstellen, daß Ihr Assembler verschiebbare Maschinencodedateien im Standard Intel Format (.OBJ) erzeugt. Dieses Format gilt als Industriestandard für Maschinen der 8086-Familie. Mit den meisten Linkern der bekannteren Hersteller (für PASCAL, FORTRAN, C etc.) können Sie die in diesen Sprachen erstellten Module (unter Verwendung bereits assemblierter Standardroutinen im Maschinencode) zu Programmen zusammenstellen.

Assemblieren von Hand

Statt eines Assemblers können Sie auch „handassemblierten" Code verwenden, den Sie mit dem BASIC-Befehl POKE in einen freien Speicherblock setzen. Dieser Ablauf eignet sich jedoch nur für einfache Aufgaben (wie die eben beschriebenen). Sie brauchen dazu eine Liste der Intel Opcodes, eine ausführliche Memory-Map der Maschine, die Information, welche Speicherbereiche zur Verfügung stehen – und viel Geduld. Da der so entstandene Code nur schwer lesbar ist, sollten Sie diese Methode aber möglichst vermeiden. Praktischerweise enthalten die meisten compilierten Sprachsysteme für MS-DOS gleich alle nötigen Erweiterungen für Systemroutinen, so daß nicht erst externer Maschinencode geschrieben werden muß.

Das von Microsoft entwickelte PASCAL-86 kann mit der Prozedur DOSXQQ direkt auf Funktionen zugreifen, und auch MT+86 von Digital Research besitzt eine ähnliche Routine. Vielleicht die beste (mit Sicherheit aber die einfachste) Lösung bietet Prospero Software in dem weit verbreiteten Compiler im ISO Standard (Pro PASCAL) und dem preiswerten Turbo PASCAL. Beide Systeme sind eine ausgezeichnete Grundlage für die DOS-Programmierung ohne Assemblersprache und helfen dem Einsteiger weiter.

Pro PASCAL und Turbo PASCAL besitzen eine TYPE Definition, die alle wichtigen Register des 8086 zu einem RECORD zusammenstellt. Dessen Felder werden vor dem Aufruf initialisiert und enthalten danach die von den DOS-Funktionen gelieferten Daten. Die Syntax ist bei jedem Aufruf gleich (die Handbücher sollten zu Rate gezogen werden, denn sie enthalten eine genaue Beschreibung).

Prospero bietet weiterhin eine Prozedur namens system, die mit einem Variablenparameter dieser Art (SysReg) den Funktionsaufruf durchführt. Die Prozedur zur Anzeige des aktuellen Laufwerks lautet daher in PASCAL:

```
WITH register DO
  BEGIN {aktuelles Laufwerk anzeigen}
    AH:=25; {Funktionscode 19H}
    system ( register ); {aufrufen }
    WriteLn ( 'Das aktuelle Laufwerk ist ',
    chr ( AL + ord ( 'A') ),';')
  END
```

Eine vollständige Funktion

Vor dem Speichern von Daten sollten Sie feststellen, wieviel Platz auf der Diskette noch zur Verfügung steht, da das Speichern auf eine fast volle Diskette unangenehme Folgen haben kann. Über den Aufruf der DOS-Funktion 36H (54 dezimal) prüfen Sie die Diskette vor dem Schreiben und können dann entweder die Datei rechtzeitig schließen oder eine andere Diskette einlegen.

Für den Systemaufruf wird die Funktionsnummer wie gewohnt in AH gestellt und die Laufwerksnummer in DL (Laufwerk A ist 1, B ist 2 etc.). Code 0 gibt das Standardlaufwerk an. Sie müssen nun nur noch das DL-Feld mit dem entsprechenden Wert initialisieren, 36H in AH speichern und die Systemfunktion aufrufen. Nach dem Rücksprung enthält AX entweder den Fehlercode 0FFFFH (beispielsweise) oder folgende Register die gültigen Daten:

AX Die Anzahl der Sektoren pro „Cluster"
BX Die Zahl der verfügbaren „Cluster"
CX Die Anzahl der Bytes pro Sektor
DX Die Gesamtzahl der „Cluster"
Die Bytezahl der freien Diskettenkapazität errechnet sich aus folgenden Werten:
Cluster * Sektoren pro Cluster * Bytes pro Sektor
Cluster sind Speicherbereiche, die eine Reihe vollständiger Sektoren umfassen.

```
FUNCTION DiskSpace ( Laufwerk : integer )
  : integer;
  { liefert die Zahl der freien Diskettenbytes }
  VAR
    register : SysReg; { siehe Handbuch }
  BEGIN
    WITH register DO
      BEGIN
        DL := Laufwerk; { 0 = Standard,
                A = 1 etc. }
        AH := 36H; { Funktionsnummer }
        system ( register ); { aufrufen }

        IF AX = OFFFFH
          THEN { ein Fehler ist aufgetreten }
            BEGIN
            WriteLn ( 'DiskSpace : ERROR
            ( ungültiger Code ? )');
            DiskSpace := 0 { mehr wissen
              wir nicht }
          END
        ELSE
            DiskSpace := AX * BX * CX
    END { Sektoren * Cluster * Bytes }
  END; { DiskSpace }
```

Fachwörter von A bis Z

Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen, damit Sie es leichter haben, das von Ihnen Gesuchte zu finden.

Query Language = Datenbanksprache
Als „Query Language" wird die Kommandosprache eines Datenbank-Betriebssystems bezeichnet. Den eigentlichen Zugriff auf die Datenbasis organisiert das System anhand der vorgegebenen Suchparameter selbst. Häufig läßt die Query Language als „interaktive" Datenbanksprache auch einen Dialog mit dem Datenbanksystem zu.

Das Vokabular derartiger Sprachen besteht aus einer begrenzten Anzahl von Befehlen, die vom Rechner interpretiert und ausgeführt werden. Dem Benutzer bleibt dabei verborgen, wie die Datenbasis im einzelnen abgefragt wird. Bei anspruchsvolleren Datenbanksprachen ist meist nicht nur das Aufspüren einer einzigen Information und ihre Wiedergabe auf dem Bildschirm vorgesehen, sondern über bedingte Anweisungen läßt sich die Recherche beliebig ausdehnen.

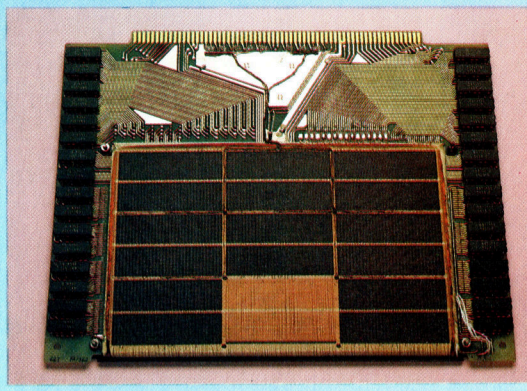
Queue = Warteschlange
Das Prinzip der Warteschlange ist etwa beim FIFO-Speicher realisiert, bei dem die zuerst eingegebenen Daten auch als erste wieder auftauchen, im Gegensatz zum „Stack" oder LIFO-Speicher.

FIFO-Speicher werden häufig als Buffer eingesetzt, vor allem bei Druckern. Da die Übertragungsrate des Rechners sehr viel höher ist als die maximale Druckgeschwindigkeit, werden die Zeichen zunächst in eine Warteschlange gestellt, die dann von vorn abgearbeitet wird.

Quicksort = Quicksort
Der „Quicksort" ist ein schneller Sortieralgorithmus. Dabei wird zunächst ein Mittelelement der ungeordneten Liste zur Vergleichsgröße erklärt und das Feld dann so in zwei Untermengen geteilt, daß alle Elemente der einen größer und alle der andern kleiner als das Vergleichselement sind. Für jede Teilmenge wird dann wieder eine Vergleichsgröße aus der Mitte genommen, abermals sortiert usw., bis die Teillisten nur noch ein Element enthalten – ihre Kettung ergibt das geordnete Feld.

Radix = Basis
In jedem Zahlensystem wird eine definierte Anzahl verschiedener Ziffernwerte benötigt – im Dezimalsystem sind es (mit der Null) zehn, im Hexadezimalsystem 16. Diese Zahl ist die „Basis" oder auch „Wurzel" des jeweiligen Systems. Bei mehrstelligen Zahlen bezeichnen die einzelnen Ziffern Vielfache von Potenzen der Basis; man spricht dabei auch von „Radix-Schreibweise". Im Zehnersystem gilt z. B.
$12{,}5 = 1*10^1+2*10^0+5*10^{-1} = 1*10+2*1+5/10$
Das Komma (oder der Punkt, engl. „Radix Point") trennt den ganzzahligen Teil vom gebrochenen, und zwar nicht nur im Dezimalsystem: Hexadezimal wäre z. B.
$23.FF = 2*16^1+3*16^0+15/16+15/16^2$
woraus sich dezimal 35,996 errechnet, und im Dualsystem entspräche 1000101.01 dezimal dem Wert 69,25.

Unter „Radix-Sorting" versteht man ein stellenweises Sortieren: Die Elemente einer numerischen Liste werden zuerst nach dem Ziffernwert der letzten Stelle geordnet, dann nach dem der nächsten usw. bis zur höchsten Stelle. Der „Radix Exchange" (stellenweiser Tausch) verläuft umgekehrt: Die Listenelemente werden nach der höchstwertigen Stelle in Teillisten einsortiert, innerhalb derer nun nach der nächstniedrigen Stelle geordnet wird usw. bis zur letzten Stelle.



Die Fortschritte in der Halbleitertechnologie haben zu einer drastischen Reduktion der Kosten und des Platzbedarfs für das einzelne RAM-Byte geführt. Die abgebildete Platine aus einem 1976 gebauten Minicomputer der Firma Computer Automation Inc. beherbergt einen Ferritkernspeicher mit etwa 2 K Kapazität – auf der gleichen Leiterplatte ließen sich heute 64 RAM-Chips zu je 256 K unterbringen.

RAM = RAM
Das „Random-Access Memory" (Direktzugriffs-Speicher) ist fast die wichtigste Komponente eines Rechnersystems. RAM-Bausteine enthalten in Gestalt von Halbleiterschaltungen zahllose Speicherzellen, die jeweils eine Null oder Eins aufnehmen können. Die Zellen sind i. a. in Matrixform angeordnet und werden von der CPU in Achtergruppen als Bytes adressiert; ihr Inhalt läßt sich auslesen, löschen oder überschreiben.

Es gibt im wesentlichen zwei RAM-Typen: Bei den „dynamischen" RAMs wird die Information auf winzigen Kondensatoren als elektrische Ladung gespeichert, die wegen der Leckströme alle paar Millisekunden „aufgefrischt" werden muß. Bei den „statischen" RAMs dienen als Speicherzellen dagegen Flipflops, die den logischen Zustand Eins oder Null ohne Auffrischung beliebig lange festhalten und außerdem einen schnelleren Zugriff gestatten.

Bildnachweise
2129, 2141, 2142,
U3: Marcus Wilson-Smith
2130, 2132, 2138, 2139, 2146, 2147, 2149, 2152, 2153, 2155: Caroline Clayton
2134: Liz Heany
2136, 2137, 2142: Kevin Jones

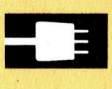
+ Vorschau +++ Vorschau +++ Vorschau +++ Vorschau +++ Vorschau +

computer kurs

Ein einziges Chaos im Adreßbuch? Alle Mitgliedsausweise und Clubkarten durcheinandergeraten, oder macht die Cassettenkartei Ärger? Im Sonderteil bringen wir ein Programm, das gründlich Ordnung schafft.

Motorolas Meisterstück

Der Adreßbereich des 68000-Microprozessors reicht an Groß-EDV heran. Wir sehen uns den Programmierteil an.



Die lange Leitung

In den USA und in England sind Modems das große Geschäft. Nun halten sie Einzug im privaten Bereich.

Zellaktivierung

Die Programmierung der Bildschirmdarstellung für den Acorn B, Schneider CPC und Sinclair Spectrum.