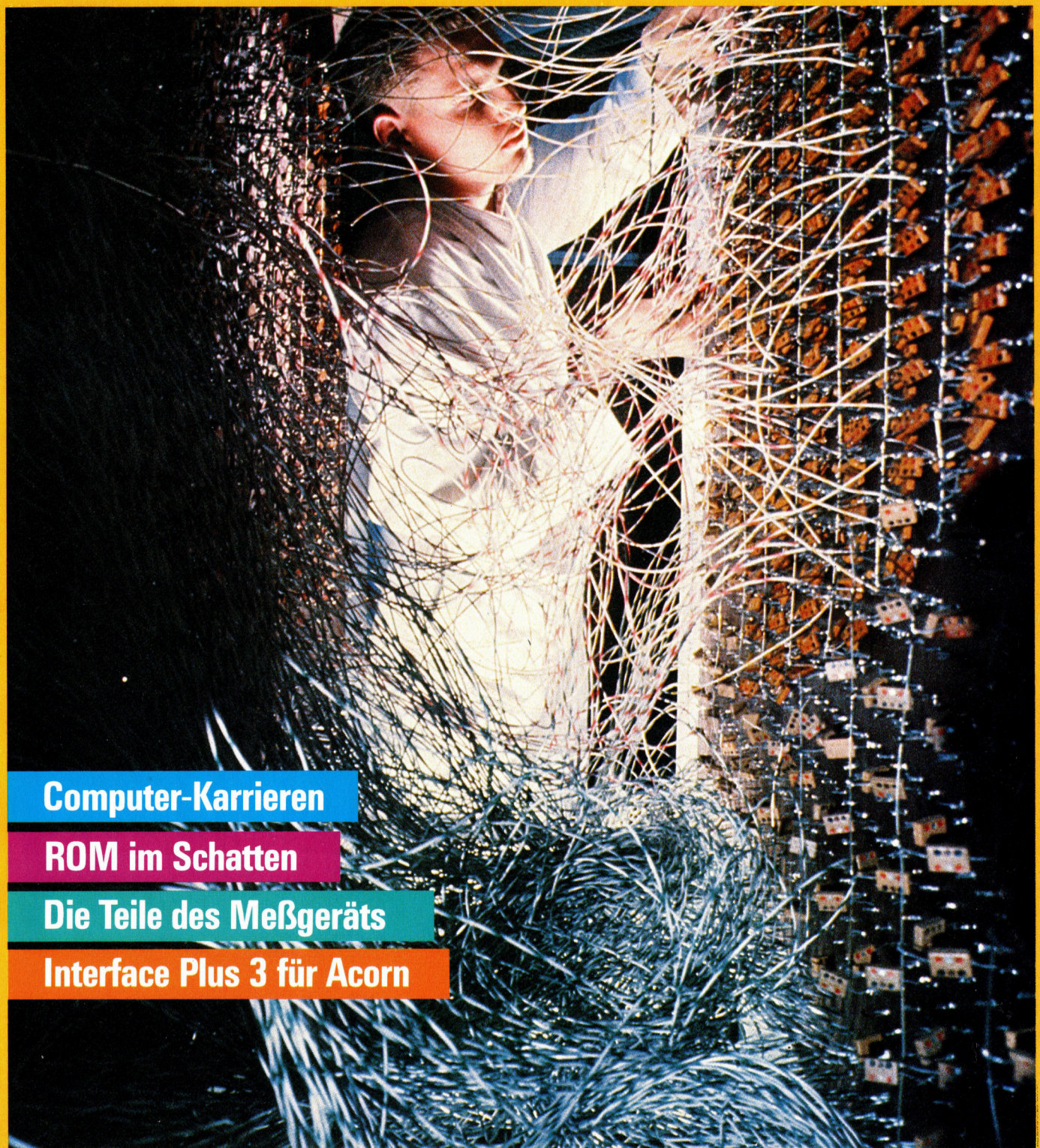
Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 73

Ein wöchentliches Sammelwerk



Computer-Karrieren

ROM im Schatten

Die Teile des Meßgeräts

Interface Plus 3 für Acorn

computer kurs

Heft 73

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.

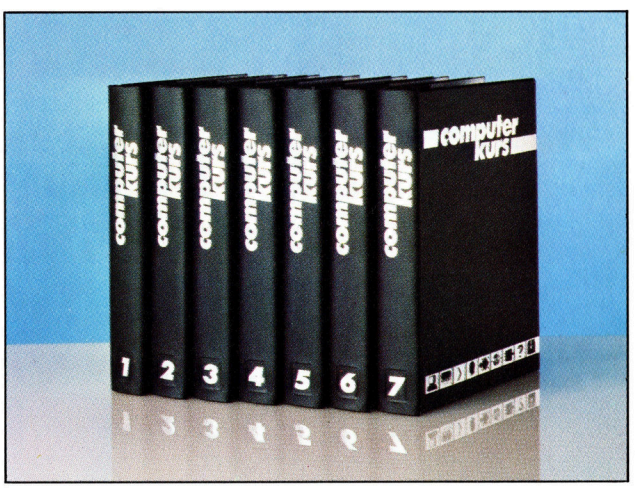
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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), Peter Aldick, Holger Neuhaus, 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

Computer-Karrieren

In diesem Bericht möchten wir Ihnen das Berufsfeld der Hard- und Softwareingenieure näherbringen – Tätigkeiten beim Entwurf neuer Systeme und Systemkomponenten. Dabei spielt auch der Bereich der Systempflege und -wartung eine bedeutende Rolle.

Der Beruf des Computer Entwicklungsingenieurs scheint heute noch ein wenig geheimnisumwittert zu sein – wir werden versuchen, den Schleier über seiner Tätigkeit etwas zu lüften und die einzelnen Bereiche dieser Aufgabe genauer darzulegen.

Am Anfang stehen die Hardware-Spezifikationen – die Entscheidung darüber, was ein neues Gerät eigentlich leisten soll. Im allgemeinen bleibt dieses Stadium dem erfahrenen Ingenieur vorbehalten, der hier eng mit der Marketing-Abteilung zusammenarbeiten muß.

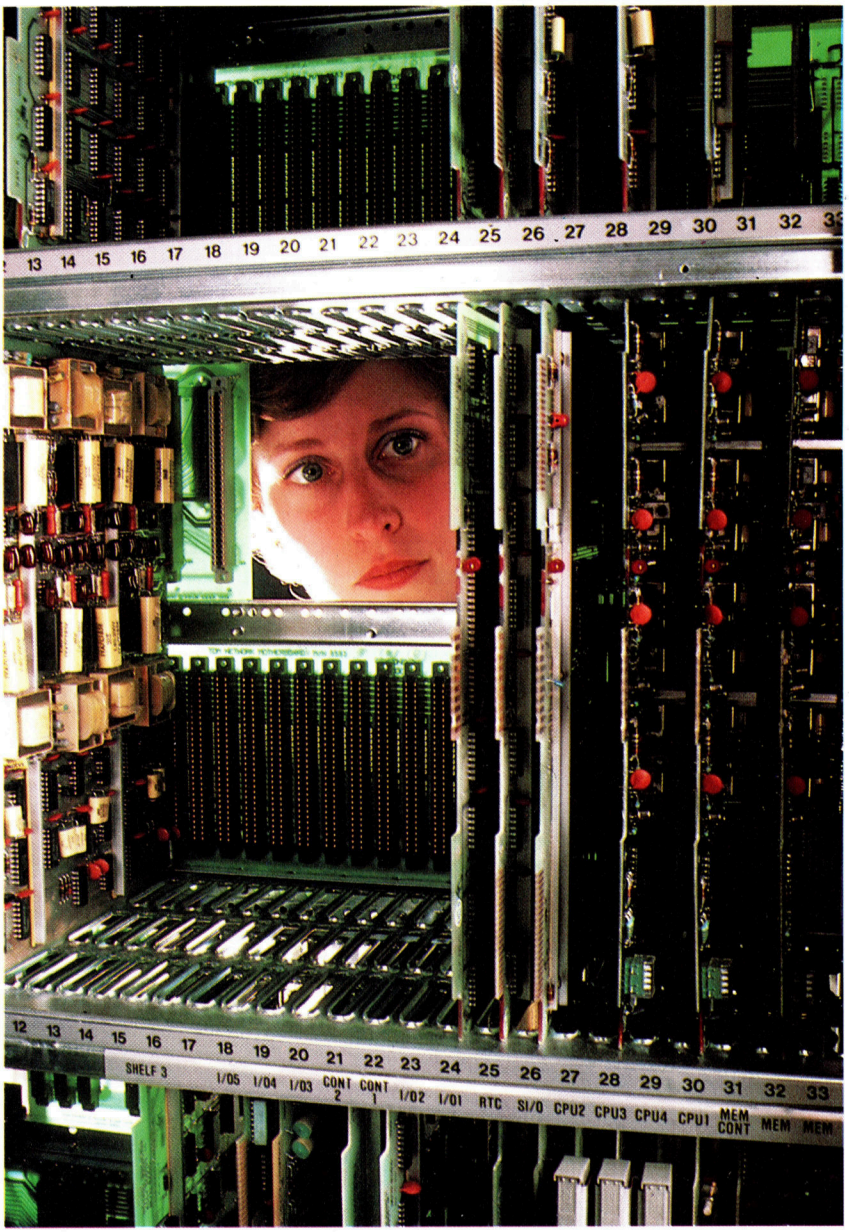
Wenn ein Ingenieur die Spezifikationen kennt, muß entschieden werden wie sie am besten umgesetzt werden können und wie lange dies dauert – Zeit ist eben wichtig bei Neuentwicklungen. Erst nach der Terminfestlegung und Kalkulation kann es ans Konstruieren gehen. Um die geeigneten Bauteile und Komponenten für ein System zu finden, muß der Ingenieur jederzeit auf dem letzten Informationsstand sein – das heißt, die Fachliteratur kennen und sein Wissen ständig aktualisieren. Vor der Produktion eines Gerätes steht meist der Test an mehreren Prototypen.

Kompromisse machen

Bei jeder Entwicklungstätigkeit müssen Kompromisse gemacht werden – wenn etwa nur sechs Monate zur Verfügung stehen, kann man nicht ein perfektes System aus dem Boden stampfen. Wenn es auf Kompatibilität mit bestehenden Geräten ankommt, ist eine Umwälzung der Systemarchitektur von vornherein unmöglich. Ist ein fester Preis vorgegeben, wirkt sich das natürlich auch auf die Bauelemente aus, die verwendet werden können. Der Ingenieur muß zwischen allen Randbedingungen den jeweils günstigsten Weg wählen.

Die Hardware-Entwicklung ist heute ein Feld der jungen Techniker. Während die „Midlife-Crisis" in den meisten industriellen Bereichen bei 40 oder 45 Jahren einsetzt, beginnt sie beim Computer-Entwickler bereits beim Übergang vom dritten ins vierte Lebensjahrzehnt. Wer mit 35 noch nicht den Sprung ins Management geschafft hat, wird oft schon für einen „Mann von gestern" gehalten, der nicht die Fähigkeiten zum Leiter in sich trägt und dessen Kenntnisse sehr bald überholt sein werden.

Der Einstieg in die Karriere des Hardware-Entwicklers vollzieht sich auf zwei verschiedenen Stufen – viele Firmen bevorzugen es, 16 bis 18-jährige Schüler mit guten Zeugnissen einzustellen und sie in der Firma selbst auszubilden. Wie in den meisten gutbezahlten Berufen gibt es aber auch hier eine zunehmende Tendenz, diplomierte Ingenieure zu engagieren. Im Hardware-Bereich haben meist nur Physiker oder Elektronikingenieure eine Chance.



Die meisten Mini- und Mainframe-Computer sind heute modular aufgebaut – die Aufgaben des Technikers werden dadurch sehr viel leichter. Für bestimmte Aufgaben gibt es jeweils einzelne Platinen. Mit besonderen Testgeräten kann jedes Modul schnell auf einwandfreie Funktion geprüft werden.

Die Unterschiede im Gehalt sind in diesem Bereich sehr augenfällig – frisch diplomierte Ingenieure oder Trainees aus dem Unternehmen selbst verdienen maximal 45.000 DM im Jahr. Die Steigerungen halten sich anfangs in Grenzen – ein zusätzliches Jahr mit neuen Erfahrungen bringt kaum mehr als einen Einkommenszuwachs von 10 %. Anders ist es beim „Senior" dieser Branche, der als Spitzenfachmann durchaus auf ein jährliches Gehalt von 100.000 DM kommen kann.

Viele Ingenieure versuchen mit 30 bis 35 Jahren, ihre Erfahrungen in der Geschäftsführung oder im Vertrieb einzusetzen. Projektleitung und Vertrieb gehören zu den hochbezahlten Aufgaben – Einkommen über 100.000 Mark sind zwar nicht die Regel, aber auch keine einmaligen „Ausreißer". Die nächsten Stufen der Karriereleiter führen den Ingenieur mehr und mehr von den technischen Aufgaben fort – im Management sind Führungsqualitäten und kaufmännische Kenntnisse Trumpf.

Die Nachfrage nach Hardware-Spezialisten ist zur Zeit groß, nicht nur in Deutschland, sondern auch im Ausland. Bemerkenswert ist, daß die Einkünfte etwa in den USA fast doppelt so hoch sind wie in der BRD, in England dagegen nur die Hälfte des hiesigen Verdienstes üblich ist. Voraussetzung sind aber in jedem Land gute englische Sprachkenntnisse – ohne Englisch als wichtigste Techniker- und Computersprache läuft hier nichts.

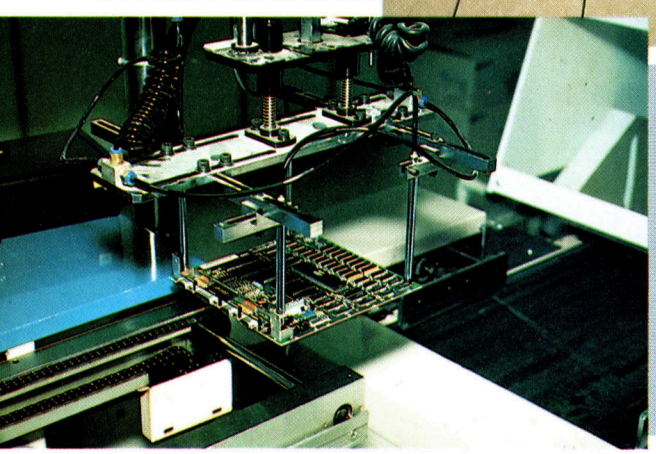
Die Entwicklung von Software steht in enger Verbindung mit der Hardware-Konstruktion – der Programmentwickler muß ja beispielsweise Betriebssysteme liefern, die sich an den Bedingungen der Hardware orientieren. Oft gehört es auch zu seinen Aufgaben, von dritter Seite gelieferte Betriebssysteme an neu entwickelte Rechnertypen anzupassen.

Software-Fachleute sind Mangelware – entsprechend ist ihre Tätigkeit besser dotiert, wobei dieses Segment des „Fachleute-Marktes" sprunghaften Schwankungen unterworfen ist. Programmentwickler kommen meist von den Universitäten und werden in den Unternehmen selbst weiter fortgebildet.

Der Karriereknick der Hardware-Entwickler jenseits der dreißig trifft auch die Programmier-Künstler, allerdings weniger ausgeprägt. Auch Software-Ingenieure finden oft den Absprung ins Management oder den Vertrieb.

Der Bereich „Wartung und Systempflege" bietet Anfängern oft noch eine Chance zum Einstieg in die Computer-Industrie, wenn man bereits Erfahrungen mit kleineren Computern gesammelt hat. Auch hier wird klar zwischen der technischen Wartung und der Entwicklung von Software unterschieden. Wer sich der Software widmet, hat im allgemeinen die besseren Verdienstmöglichkeiten.

Auch das Werkzeug für die Computer-Wartung wird zunehmend „intelligent": Unser Bild zeigt einen Servicetechniker, der den Supercomputer „Cray" mit einer Infrarotkamera untersucht. Die Kamera lokalisiert überhitzte Stellen auf der Platine, die zu einer Fehlfunktion führen könnten. Viele Rechner verfügen heute bereits über Systeme zur Selbstdiagnose – die Reparatur beschränkt sich dann auf den Austausch des als fehlerhaft gemeldeten Moduls. Fachliche Anforderungen an das Wartungspersonal werden damit in Zukunft wesentlich herabgesetzt.



Bereits im Anfangsstadium einer Planung soll der Computer-Entwickler die Eigenheiten der Produktionstechnik berücksichtigen: Bei Geräten, die sich auf dem hart umkämpften Rechnermarkt durchsetzen sollen, muß jede Rationalisierungsmöglichkeit durch Automatisierung genutzt werden. Platinen beim Apple MacIntosh werden vom Roboter bestückt.

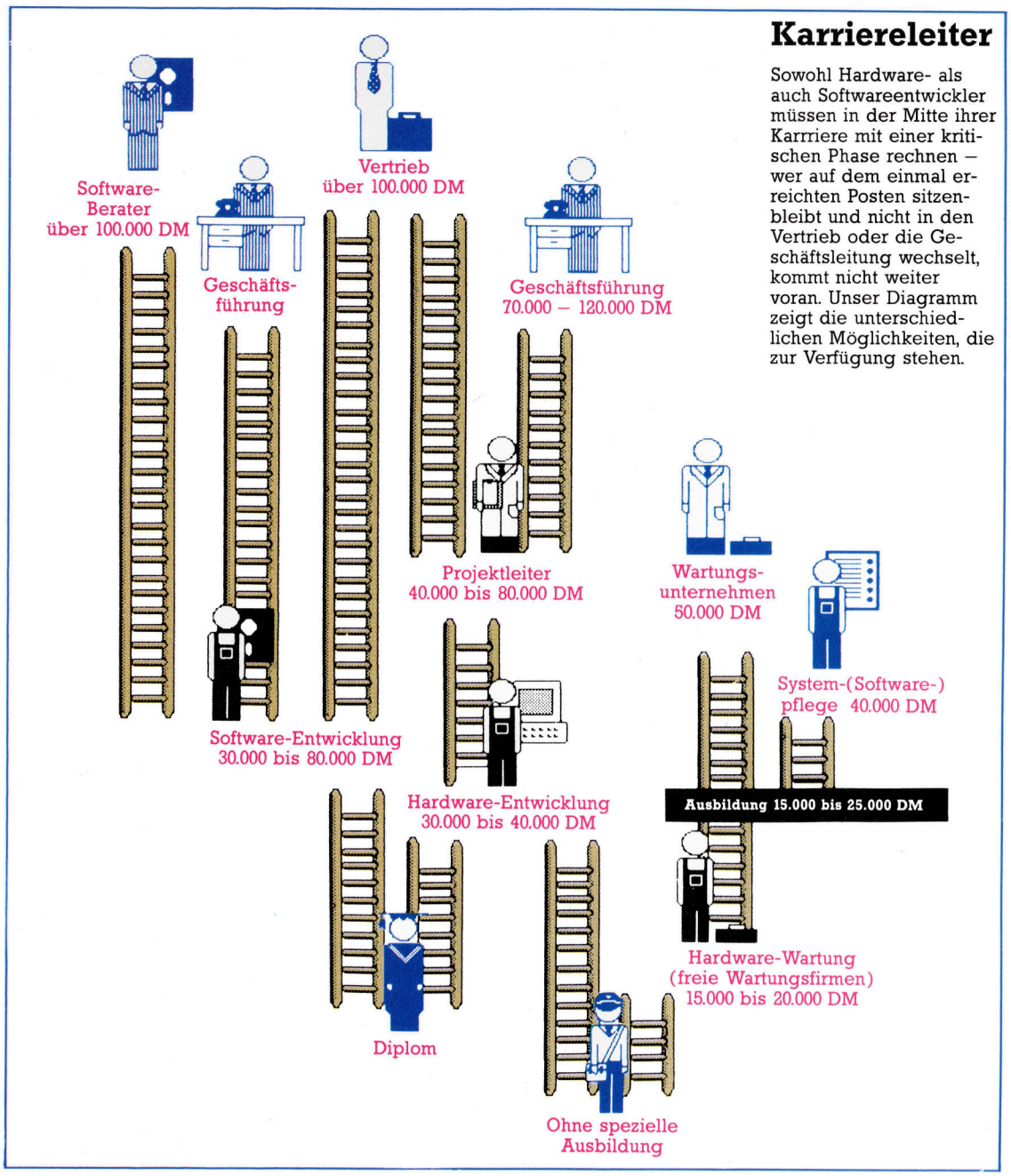
Chancen für Freaks

Zur Hardware-Wartung gehört die Installation von Systemen beim Kunden sowie die Fehlersuche und Reparatur von Anlagen. Arbeitgeber ist hier entweder der Hersteller des Gerätes oder eine Drittfirma, die sich ausschließlich um die Wartung beziehungsweise die Reparatur von Fremdrechnern kümmert.

Fast alle Computer-Händler verfügen über eigenes Wartungspersonal – nicht selten wird auch dem interessierten jungen Computer-Freak eine Chance gegeben, allerdings bei recht niedrigem Salär. Das kann sich jedoch mit der Zeit ändern – ein tüchtiger Wartungsfachmann kann mit genügend Erfahrung auch die 50.000 Mark-Grenze durchbrechen.

Gerade in kleineren Firmen erwirbt das Wartungspersonal auch Software-Kenntnisse, muß Geräte vorführen und den Anwerber einweisen können. In größeren Firmen bieten sich vielfältige Entwicklungsmöglichkeiten – der Übergang zur besser bezahlten Software-Systempflege, ein Einstieg bei einem reinen Wartungsunternehmen oder der Sprung in den erfolgsorientierten Vertriebsbereich.

Es gibt reine Wartungsunternehmen mit überraschend großem Mitarbeiterstab. Für die



Weiterbildung des Personals wird meist recht gründlich gesorgt – oft drücken die Mitarbeiter mehr als vier Wochen jährlich die Schulbank und lernen den Umgang mit neuen Systemen.

Falls Sie sich selbst für eine Tätigkeit in der Wartung interessieren, sollten Sie mehrere Punkte im Auge behalten: Computer werden zunehmend verläßlicher (auch wenn man oft das Gegenteil hört . . .) und sind immer einfacher zu warten. Vielfach muß bei Störungen nur noch ein Modul ausgewechselt werden, das durch Diagnoseprogramme sogar noch spezifiziert wird. Das könnte bedeuten, daß Wartungspersonal durch den technischem Fortschritt in naher Zukunft überflüssig wird.

Zur Softwarepflege gehört es, herauszufinden, warum bestimmte Programme auf manchen Rechnern nicht laufen. Das Personal muß Programme an bestimmte Rechner-Konfigurationen anpassen und kommt auch in Berührung mit Spezialgebieten wie etwa der Daten(fern-) übertragung.

Wie im Hardware-Bereich können noch relativ unerfahrene Interessenten hier Fuß fassen. Als Einstieg bieten sich drei Möglichkeiten an: Der 16- bis 17 Jahre alte Computerfreak muß lernwillig sein und für mehrere Jahre auf gute Bezahlung verzichten. Der Anwender mit Programmiererfahrung und Kenntnissen über Personalcomputer hat ebenfalls Chancen.

Abgesehen von der ersten Bewerbergruppe ist auch die Bezahlung nicht ganz uninteressant – für den kenntnisreichen Newcomer liegt die Untergrenze bei ca. 2000 Mark im Monat, sie läßt sich aber je nach Erfahrung und Kenntniszuwachs schnell anheben.

Bewegliche Typen

Über die anwenderdefinierten Datentypen von C läßt sich diese sehr flexible Sprache noch weiter verbessern und erweitern. In dieser Folge untersuchen wir einige der Erweiterungen und sehen uns ihr Zusammenspiel in einem Bridge-Programm an.

In der auf Flexibilität ausgelegten C-Sprache lassen sich selbstverständlich auch die Standardfunktionen erweitern. Wir haben bereits gesehen, wie anwenderdefinierte Funktionen in eigenen Dateien gespeichert und mit include in ein Programm integriert werden. Auch normale Datentypen wie int, char etc. lassen sich verbessern. typdef kann die Standardtypen beispielsweise mit neuen Namen versehen. Hier einige Beispiele:

```
typdef int meter;
typdef double vektor [10];
typdef char *string;
```

Die Befehle ermöglichen, im Programm Variablen zu deklarieren.

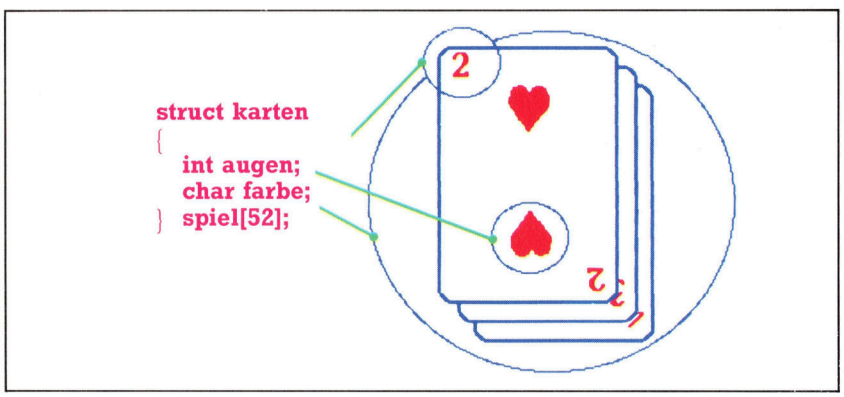
Dies scheint nicht allzuviel Sinn zu machen, hat aber zwei ganz bestimmte Gründe: Die Programme werden dadurch übersichtlicher und verursachen bei der Übertragung von einem Maschinentyp zum anderen weniger Probleme. Normalerweise ist die Zahl der Bytes einer Ganzzahl nicht standardisiert. Ein Programm mit Ganzzahlen im Vier-Byte-Format kann nur mit viel Mühe auf eine Maschine übertragen werden, die mit dem Zwei-Byte-Format arbeitet.

Ein typdef am Programmanfang läßt sich viel leichter ändern, als jedes einzelne Vorkommen dieser Definition während des gesamten Programms. typdefs stehen in einer include-Datei.

Wie PASCAL kann auch C Datenelemente unterschiedlicher Typen in Strukturen zusammenfassen. Der Befehl struct ähnelt dabei dem RECORD von PASCAL. Hier ein Beispiel:

```
struct karten
{
   int augen;
   char farbe;
}
```

Mit dem Typ struct kann der C-Programmierer unterschiedliche Datentypen zu einer Struktur zusammenstellen. Ein Kartenspiel kann beispielsweise mit Farbe und Augenzahl dargestellt und in einer Struktur zusammengefaßt werden.



Hier wird eine Struktur definiert, die aus zwei Teilen besteht – einer Ganzzahl und einer Zeichenfolge – und zusammen eine Spielkarte bezeichnet. Variablen dieses Typs können von dieser Programmposition an deklariert werden.

```
struct karten spiel [52];
```

legt ein Kartenspiel an. Die Variable läßt sich aber auch mit einer einfachen Strukturdefinition deklarieren:

```
struct karten
{
   int augen;
   char farbe;
} spiel [52];
```

Die Bezeichnung (hier karten) kann auch weggelassen werden, wenn alle Variablen an einer Stelle deklariert werden,

```
struct
{
   int augen;
   char farbe;
} spiel [52], blatt [4][13], * gespielt;
```

doch ist dies schlechter Programmierstil.

Pointervariable möglich

Strukturen lassen sich als Ganzes zuordnen, als Argumente an eine Funktion übergeben, oder wie bei einer Standardfunktion als Werte zurückgeben:

```
karten [1][5] = spiel [27];
```

Die einzelnen Elemente können aber auch mit einem Punkt als normale Variable angesprochen werden, oder mit → als Pointervariable:

```
gespielt → augen = spiel [34]. augen;
```

Die Elemente verhalten sich wie normale Variablen des entsprechenden Typs. Die Strukturen werden wie Arrays initialisiert, indem hinter den Variablennamen in Klammern eine Elementenliste aufgeführt wird.

Zur Speicherung ihrer Elemente reservieren die Strukturen aufeinanderfolgende Computerworte. „unions" verwenden jedoch den gleichen Platz für alle Elemente und machen so eine andersartige Verwendung des Arbeitsspeichers möglich.

Wenn eine Maschine beispielsweise vier Bytes für eine Ganzzahl braucht, und diese Bytes einzeln angesprochen werden sollen, kann ein union die gleichen vier Bytes für ein int oder vier chars einsetzen. Das sieht dann folgendermaßen aus:

```
union int_or_char
{  int int_teil;
   char char_teil [4];
}
```

Die Elemente von union werden wie die einer Struktur angesprochen.

Der normale Sprachgebrauch bezeichnet die Elemente einer Struktur als Felder. In C bezieht sich dieser Begriff jedoch auf einen Elemententyp, der auch einzelne Bits ansprechen kann. Wir haben bereits gesehen, daß C Bitoperatoren besitzt. Mit Feldern können Sie nun jede Bitgruppe eines Wortes mit einem oder mehr Bits über einen Namen ansprechen.

Auf einer Maschine mit 16-Bit-Worten und einem int-Typ mit 16 Bits ist folgende Definition möglich:

```
struct wort_in_bytes
{
   unsigned byte0 : 8, bytel : 8;
} struct wort_in_bits
{ unsigned bit0 : 1,bit 1 : 1,bit 2 : 1,bit 3 : 1,
   bit4 : 1,bit5 : 1, bit6 : 1,bit7 : 1,bit8 : 1,
   bit9 : 1,bit10 : 1,bit11 : 1,bit12 : 1,bit13 : 1,
   bit14 : 1,bit15 : 1;
}
union wort
{ int w;
   struct wort_in_bytes w8;
   struct wort_in_bits w1;
}
```

Sie können nun das Speicherwort als Ganzes ansprechen – als zwei Bytes oder als 16 Bits. Beachten Sie den Datentyp unsigned, der int entspricht, aber nur positive Werte aufnehmen kann. Bei einer Größenfestlegung dieser Art müssen Sie jedoch darauf achten, daß nur geeignete Werte zugeordnet werden. Ein Element mit Bitgröße kann beispielsweise nur die Werte 0 bis 1 aufnehmen.

Ein Bridgespiel

Das folgende Programmbeispiel zeigt, wie mit einigen wenigen Funktionen ein Bridgespiel aufgebaut werden kann. Das Programm mischt die Karten und teilt sie an vier Spieler aus. Die Spieler können nun über die Punktwerte der Karten entscheiden, was ein Blatt wert ist. Dabei erhält das Ass vier Punkte, der König drei, die Königin zwei und der Bube einen. Das Programm simuliert das Mischen, Austeilen und Bewerten des Blattes.

Die Kartenstruktur und das Kartenarray des Spiels werden extern definiert und damit auch gleich initialisiert.

```
struct karten
{
   int augen;
   char farbe;
}
/* jede Karte belegt den kleinstmöglichen
Platz */
/* Spiel deklarieren und initialisieren */
struct karten blatt[4][13], deck[52]=
   {{1,'C'}, {2,'C'}, {3,'C'}, {4,'C'}, {5,'C'}, {6,'C'},
   {7,'C'}, {8,'C'}, {9,'C'}, {10,'C'}, {11'C'},
   {12'C'}, {13'C'}, {1,'D'}, {2,'D'}, {3,'D'}, {4,'D'},
   {5,'D'}, {6,'D'}, {7,'D'}, {8,'D'}, {9,'D'}, {10,'D'},
   {11,'D'}, {12,'D'}, {13,'D'}, {1,'H'}, {2,'H'},
   {3,'H'}, {4,'H'}, {5,'H'}, {6,'H'}, {7,'H'}, {8,'H'},
   {9,'H'}, {10,'H'}, {11,'H'}, {12,'H'}, {'H'13'H'},
   {1,'S', {2,'S', {3,'S', {4,'S', {5,'S', {6,'S', {7,'S',
   {8,'S', {9,'S', {10,'S', {11,'S', {12,'S', {13,'S'}}
   shuffle(spiel)
   struct karten spiel[];
   /* Wir setzen voraus, dass ein Zufallszahlengenerator – random(n) – vorhanden ist, der eine Zahl zwischen 0 und n-1 liefert. Die Systemfunktion rand() erzeugt eine Ganzzahl. */
   /* Die Karten werden durch zufallsgesteuertes Austauschen ihrer Position
gemischt */
{
   int i,j;
   for (i=0; i<52;++i)
   {
      j=random(52);
      swap(&spiel[i], &spiel[j];
   }
}swap (p,q)
struct karten*p, *q;
{
   struct karten temp;
   temp=*p;
   *p=*q
   *q=temp;
}
austeilen (spiel,blatt)
struct karten spiel[], blatt[][];
{  int i,j,k=0;
   for (i=0;i<4;++i)
      for (j=0;j<13;++j)
         blatt[i][j]=spiel[k++];
}
punkte_zaehlen(a_blatt)
struct karten *a_blatt
/* a_blatt ist ein Zeiger auf die erste
Karte eines Blattes (ein Set von 13),
z. B. &blatt[2][0] */
{
   int i, punkte_zaehlen=0;
   for (i=0;i<13;++i)
   {     if (a_blatt-)augen=1)
         punkte_zaehlen+=4;
      else if (a_blatt-)augen>10)
         punkte_zaeh-
            len+=a_blatt->augen
         -10;
      ++a-blatt;
/* zeigt zur naechsten Spielkarte */
   }
 return(punkte_zaehlen);
}
```

Drangvolle Enge

Textverarbeitungspakete für Heimcomputer sind zwangsläufig durch die Leistungsgrenzen des jeweiligen Rechners geprägt, wobei vor allem die verfügbare Speicherkapazität den Spielraum sehr einengt. Wir stellen hier drei Textsysteme für einige der bekanntesten Heimcomputer vor.

Die Textverarbeitung gehört zweifellos zu den beliebtesten Einsatzfeldern für Heimcomputer; das Angebot reicht vom einfachen Texteditor bis zu komplexer und entsprechend teurer Disketten- oder ROM-Software. Keines dieser Programme kann natürlich die hardwarebedingten Beschränkungen des Rechners überspielen, und deshalb erreichen die meisten Heimcomputer-Textsysteme natürlich nicht das Leistungsniveau, mit dem die Bürorechner-Pakete aufwarten können.

Das größte Handikap für die Textverarbeitung ist bei einem 8-Bit-Rechner die geringe Speicherkapazität. Jeder zusätzliche Komfort kostet Platz für Software, der für den Text selbst benötigt würde. Das war z. B. der eigentliche Hemmschuh bei der Implementierung von „WordStar" auf den Schneider-Computern CPC 464 und 664: Zwar arbeiten beide Rechner mit CP/M, aber die Speicherkapazität reicht einfach nicht für einen sinnvollen Betrieb des Original-WordStar aus. Deshalb wurde dafür extra eine abgemagerte Taschenausgabe (Pocket WordStar) verfaßt.

Easy Script

Hinderlich bei der Textverarbeitung mit Heimcomputern ist auch das Massenspeicher-Problem. Wenn nur ein Cassettenrecorder zur Verfügung steht, ist das Laden und Speichern von Programmen und Texten eine entnervend langwierige Prozedur.

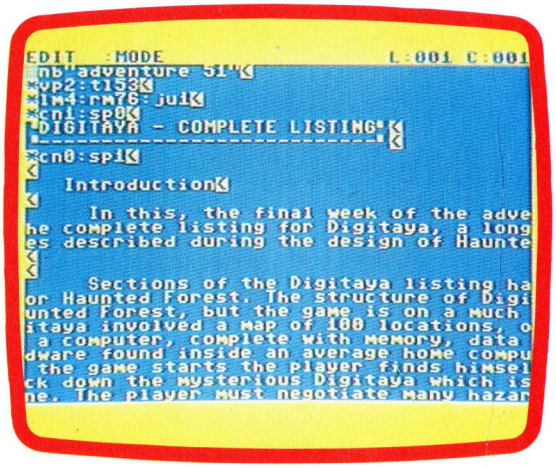
Für den Commodore 64 wurden zahlreiche Textprogramme geschrieben, von denen „Easy Script" eines der bekanntesten ist.

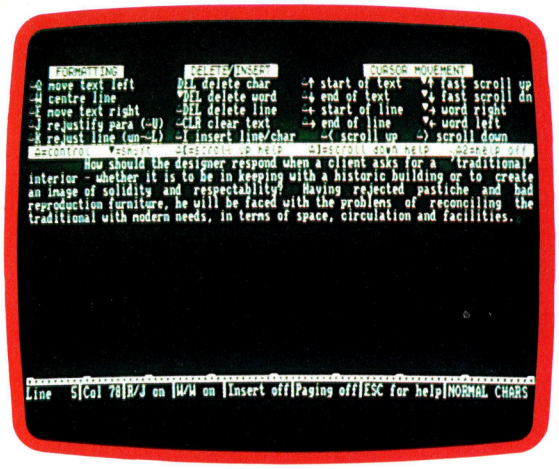
Die Hauptschwierigkeit für die Textverarbeitung beim Commodore 64 ist das Bildschirmformat – es sind nur 40 Zeichen pro Zeile bei insgesamt 25 Zeilen darstellbar. Es gibt bei Easy Script daher weder speicherintensive Bildschirmhilfen noch einen Zeilenumbruch während der Eingabe. Allerdings können Sie sich den Text vor dem Drucken im umbrochenen Zustand auf dem Schirm ansehen.

Um mit dem Problem der „kurzen" Zeilen fertigzuwerden, wird bei Easy Script der Bildschirm als „Fenster" über das Schriftstück geschoben. Zwischen linkem und rechtem Rand dürfen 240 Anschläge liegen, und wenn beim Schreiben einer Zeile die 40-Zeichen-Grenze erreicht ist, wird der Text mit jedem folgenden Anschlag um eins nach links versetzt, bis zum 240sten Zeichen. Danach springt die Anzeige abrupt auf den nächsten Zeilenanfang. Dieses Verfahren ist nicht ideal, aber man gewöhnt sich daran. Andernfalls müßte man mit 40 Zeichen pro Zeile schreiben und vor dem Drucken neu formatieren.

Die verschiedenen Bearbeitungskommandos werden über die Funktionstasten des Commodore 64 aktiviert, und zwar wird durch ihre Betätigung zunächst auf einen bestimmten Modus umgeschaltet, und erst der folgende Tastendruck ruft den gewünschten Befehl auf. Die Editierbefehle sind beispielsweise über F1 zu erreichen, wobei ein „Edit" am oberen Bildrand erscheint. Text- und Seitenformatierung wählt man über F3 – das verwandelt den Cursor in einen inversen Stern. Anschließend lassen sich die Befehle für die Randbreiteneinstellung, das Ausrichten des Textes usw. eingeben. Diese und andere Kommandos werden bei Easy Script in den Text eingebettet und ausgeführt, sobald der Schriftsatz ausgedruckt wird.

Easy Script bietet aber im übrigen fast alle Möglichkeiten, die von einem Textverarbeitungsprogramm erwartet werden können, so die ganze Blocktechnik einschließlich Kopieren, Übertragen und getrenntem Speichern von Blöcken. Allerdings funktioniert nicht alles gleich elegant; das Suchen und Ersetzen etwa geht nervtötend langsam, insbesondere bei der Anwendung auf gekettete Dateien, die einzeln geladen und durchforstet werden müssen.





Tasword II

Dieses Programm wird häufig bei Schneider-Rechnern und dem Spectrum+ eingesetzt. Nach dem Laden offenbart sich eine überraschende Ähnlichkeit mit WordStar: Oben erscheint auf dem Bildschirm ein Hilfsmenü, das verfügbare Befehle in logischer Gliederung darstellt. Darunter steht ein Feld von 16 Zeilen zu 80 Zeichen für die Texteingabe bereit.

Tasword II ist ein leistungsfähiges Textverarbeitungspaket mit umfangreichen Befehlen für die Textformatierung. Die Schrift kann nach links oder rechts verschoben und zentriert werden, bei einer maximalen Zeilenlänge von 128 Anschlägen. Wenn die Randbreiten neu gesetzt werden, läßt sich der Text wieder entsprechend ausrichten. Auch ein Zeilenumbruch ist vorgesehen, wobei nach Übernahme eines Wortes in die nächste Zeile automatisch der rechte Rand ausgeglichen wird. Ungewöhnlich dabei ist, daß die Einfügung der zusätzlichen Leerräume nur in der zweiten Zeilenhälfte erfolgt.

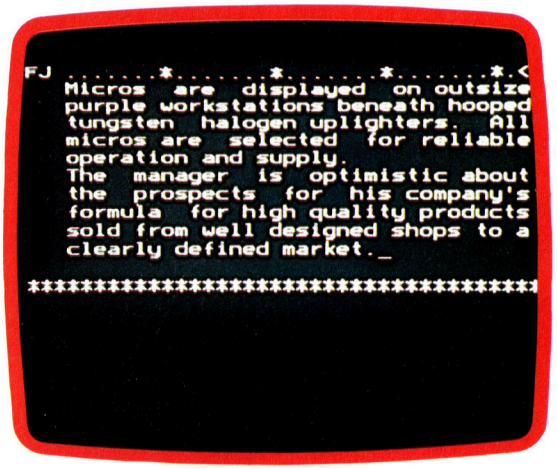
In beschränktem Umfang erlaubt das Programm auch die Bearbeitung von Textblöcken, die nach ihrer Definition verschoben, kopiert oder gelöscht werden können. Unpraktisch ist die Blocktrennung, die immer nur ganzzeilig erfolgt. Auch wenn der Benutzer den Blockbeginn mitten in eine Zeile legt, wird durch einen Transportbefehl stets die ganze Zeile mitgenommen, einschließlich der ersten Hälfte.

Ein ähnliches Problem tritt im Einfügungsmodus auf. Die meisten Textsysteme verschieben beim Einfügen auf der aktuellen Cursorposition den übrigen Zeileninhalt nach rechts. Bei Tasword kann man innerhalb der Zeile nur einen einzigen Buchstaben ergänzen. Andernfalls erzeugen Sie in jedem Fall eine neue Zeile unter der gerade bearbeiteten, auch wenn die Korrektur eigentlich in die alte Zeile gehört.

Vorteilhaft dagegen sind die vielfältigen Möglichkeiten zur Druckersteuerung. Zunächst einmal kann der Benutzer seine Tasword-Version individuell „installieren", indem er das Seiten-Layout festlegt und die Steuerzeichen für den Ausdruck nach Wunsch umdefiniert.

Tasword zeichnet sich durch eine Fülle von Drucker-Steuercodes aus, die am Epson-Standard orientiert sind. Sie werden durch gleichzeitige Betätigung der Control- und der Leertaste und einen nachfolgenden Buchstaben aktiviert. So schaltet jeweils der Großbuchstabe die betreffende Funktion ein und der Kleinbuchstabe aus. Es gibt zwanzig verschiedene Steuerzeichen, von variablem Zeilenabstand bis zu diversen Arten der deutlichen Hervorhebung von Passagen, wie etwa Unterstreichen, Fett- und Kursivdruck.

Zusätzliche Schriftarten wie „Lectura Light" und „Compacta" sind im Prinzip ebenfalls über Steuerzeichen wählbar, dazu wird aber noch das Programm „Tasprint" benötigt, das der Hersteller „Tasman" als Ergänzung vorgesehen hat. Es läßt sich für eine Vielzahl von Matrixdruckern konfigurieren und belegt einen Speicherbereich, der beim Laden von Tasword nicht überschrieben wird. Dem Benutzer stehen damit fünf weitere Schriftarten zur Verfügung, allerdings zu Lasten des verbleibenden Speicherraums für den Text. Denn Tasprint beschlagnahmt rund fünf von den 13 KByte, die sonst frei wären.



View

Etwas anders als beim Commodore 64 ist die Problematik beim Acorn B. Hier stört vor allem, daß der Bildschirm so viel von dem knappen Speicherraum beansprucht. Wie andere Hersteller liefert daher auch AcornSoft das Textsystem „View" als ROM-Version für den Einbau in einen der Acorn B-Leersockel. Damit bleibt das RAM voll verfügbar, und es besteht nicht wie bei anderen größeren Textpaketen die Notwendigkeit, Extraprogramme von der Diskette einlesen zu müssen.

Das ist ein wichtiger Gesichtspunkt. Wenn Sie mit 80 Zeichen pro Zeile arbeiten wollen, haben Sie bestenfalls noch Speicherplatz für 10 000 Textzeichen. Müßten Sie dabei noch ein Zusatzprogramm von acht KByte Länge laden, blieben nicht mehr viel für den Text übrig. Sie sind bei View aber nicht auf den 80-Zeichen-

Modus festgelegt, weil sich das Bildschirmfenster wie bei Easy Script und den meisten anderen Textsystemen über das Geschriebene hinwegschieben läßt.

View gehört zweifellos zu den anspruchsvolleren Textverarbeitungsprogrammen für Heimcomputer. Anstelle von Hilfsmenüs gibt es dazu eine Auflage für die Funktionstasten, über die fast alle Textkommandos erreichbar sind – entweder direkt oder in Verbindung mit der Shift- bzw. der Control-Taste.

Als Besonderheit bietet View noch die Möglichkeit, eigene „Makros" zu definieren. Das sind kurze Programme (bestehend aus View-Kommandos und Befehlstext), die über einen selbstdefinierten Zwei-Buchstaben-Code als Makroanweisungen aufrufbar sind.

View enthält auch Befehle für die Wortzählerfunktionen, die Formatierung und eine „kontinuierliche" Verarbeitung.

Taschenausgabe

Es ist bezeichnend, wie ernst die Rechner der Schneider-Klasse inzwischen genommen werden und daß Softwarelieferanten ihre CP/M-Bürorechnerprogramme jetzt für die Schneider-Computer umschreiben. Für die Firma Schneider (bzw. die englische Mutter Amstrad) ist es ein großer Erfolg, daß MicroPro eine CPC 464/664-Version des recht bekannten Textverarbeitungspakets „WordStar" herausgebracht hat.

Obwohl der „Pocket WordStar" sehr viel weniger Speicherplatz als die ursprüngliche Fassung belegt, hat er nicht viel an Können eingebüßt – Sie finden bei der abgemagerten Version fast alle WordStar-Funktionen wieder.

Die Lösung von MicroPro sieht so aus, daß ein Teil der Untermenüs erst bei Bedarf geladen wird. Deshalb arbeitet der „Pocket WordStar" natürlich etwas langsamer als die Originalfassung, das ist aber ein zumutbares Opfer, wenn dadurch ein so exzellentes Textprogramm für einen Rechner der unteren Preisklasse verfügbar wird.

Easy Script	Tasword II	View
ZEILENUMBRUCH Ein Zeilenumbruch findet erst statt, wenn der Schriftsatz dem Drucker übergeben wird.	**ZEILENUMBRUCH** Die Zeilen werden automatisch umgebrochen und rechtsbündig ausgerichtet.	**ZEILENUMBRUCH** Die Zeilenumbruch-Funktion wird voll unterstützt, läßt sich aber nicht abschalten.
BLOCKBEARBEITUNG Das Programm ermöglicht Übertragen, Kopieren und getrenntes Abspeichern.	**BLOCKBEARBEITUNG** Tasword unterstützt Übertragen, Kopieren und Löschen, aber nicht separates Abspeichern von Blöcken.	**BLOCKBEARBEITUNG** Bei View können Blöcke verschoben, kopiert und gelöscht werden.
BILDSCHIRMHILFE Nicht vorgesehen.	**BILDSCHIRMHILFE** In einem Fenster können alle verfügbaren Befehle im Rollmodus nacheinander gezeigt werden.	**BILDSCHIRMHILFE** Bildschirmhilfe wird nicht geboten, aber es gibt eine Auflage für die Funktionstasten.
80-ZEICHEN-BILDSCHIRM Die Bildschirmdarstellung erfolgt mit 40 Zeichen pro Zeile.	**80-ZEICHEN-BILDSCHIRM** Auf dem Bildschirm werden 80 Zeichen pro Zeile dargestellt, wobei die Zeilen insgesamt 128 Zeichen enthalten dürfen.	**80-ZEICHEN-BILDSCHIRM** Bei View sind 76 Zeichen pro Zeile darstellbar; dabei bleibt aber nur wenig Platz für die Textspeicherung übrig.
WORTZÄHLER Ein Wortzähler ist nicht vorhanden.	**WORTZÄHLER** Wird ständig angezeigt.	**WORTZÄHLER** View hat einen Wortzähler, der aber nicht ständig angezeigt wird.
SUCHEN UND ERSETZEN Die Suchfunktion verarbeitet Strings mit bis zu 32 Zeichen, benötigt aber extrem viel Zeit.	**SUCHEN UND ERSETZEN** Bei Tasword ist nur das Suchen einzelner Wörter vorgesehen.	**SUCHEN UND ERSETZEN** Es ist zwar nur das Suchen und Ersetzen einzelner Wörter vorgesehen, aber dabei sind auch Wildcards für Zeichen zulässig.
DRUCKGETREUES TEXTBILD Das Druckbild läßt sich mit Easy Script nur ausschnittweise auf dem Bildschirm wiedergeben.	**DRUCKGETREUES TEXTBILD** Etliche Formatierungsbefehle werden erst vom Drucker als Steuerzeichen erkannt und sind daher auf dem Bildschirm wirkungslos.	**DRUCKGETREUES TEXTBILD** Das Programm erlaubt die vollständige Wiedergabe des Textes, wie er im Druck erscheint, ausgenommen das Unterstreichen von Fettschrift.
SERIENBRIEFE Einfaches Serienbriefprogramm, das allerdings keine Adreßetiketten ausdrucken kann.	**SERIENBRIEFE** Nicht vorgesehen.	**SERIENBRIEFE** Nicht vorgesehen.
RECHTSCHREIBKONTROLLE Unter dem Namen „Easy Spell" ist hierfür beim Hersteller ein Zusatzprogramm erhältlich.	**RECHTSCHREIB-KONTROLLE** Nicht vorhanden.	**RECHTSCHREIBKONTROLLE** Noch nicht verfügbar, aber von AcornSoft angekündigt.
SCHRIFTARTEN Easy Script unterstützt gedehnten, komprimierten, kursiven und fetten Druck.	**SCHRIFTARTEN** Tasword II unterstützt verschiedene Schrifttypen; weitere bietet das Zusatzprogramm „Tasprint".	**SCHRIFTARTEN** Nur Fettdruck zum Hervorheben von Passagen wird unterstützt.
DATEIKETTUNG Das Programm enthält Kommandos für das Ketten von Cassetten- und Diskettendateien beim Editieren und beim Ausdrucken.	**DATEIKETTUNG** Sie können hier einer Datei höchstens eine andere vorschalten.	**DATEIKETTUNG** Durch die Makro-Befehle ist eine ganze Anzahl an Möglichkeiten zum beliebigen Ketten von Dateien gegeben.

Ein ROM im Schatten

Das Interface 1 gibt dem Sinclair Spectrum nicht nur Zugang zu den Microdrives, dem lokalen Netzwerk (LAN) und der seriellen Schnittstelle, sondern ermöglicht auch Maschinencoderoutinen. Wir sehen uns einige dieser Routinen genauer an.

Das Interface besitzt ein ROM mit acht KBytes, die den Anfangsbereich des Speichers belegen. Der als „Schatten-ROM" bezeichnete Chip enthält Routinen zur Steuerung der Zusatzgeräte (z. B. der Microdrives) und einen BASIC-Interpreter, der Befehle wie CAT und FORMAT entgegennimmt.

Es existieren mehrere Schatten-ROM Versionen, die zwischen Version 1 und den neueren Ausgaben (Version 2) wesentliche Unterschiede aufweisen. Version 1 konnte beispielsweise die Microdrives nicht optimal einsetzen.

Spätere Versionen sprechen die Microdrives effektiver an. Weiterhin wurden Fehler bereinigt und neue Fähigkeiten für serielle Drucker eingebaut.

Für den Anwender sind die Unterschiede zwischen den Versionen normalerweise „transparent" – vorausgesetzt, das Interface wird über BASIC oder mit den korrekten Maschinencodetechniken angesprochen. Wenn Sie die Routinen des Schatten-ROM direkt aufrufen, werden Sie mit Sicherheit Schwierigkeiten bekommen, da die Routinen von Version zu Version unterschiedliche Adressen benutzen. Alle Adreßangaben dieser Serie beziehen sich auf die Version 1 des Schatten-ROMs.

ROM ein, ROM aus

Wie kann nun das Schatten-ROM die ersten acht Speicherbytes belegen, wenn sich dort ebenfalls das BASIC-ROM befindet? Hier wird mit einer Technik gearbeitet, die dem System der Paged ROMs des Acorn B entspricht. Sobald der Z80 versucht, Befehle aus den Adressen &08 oder &1708 zu lesen, wird das BASIC-ROM „abgeschaltet" und das Schatten-ROM aktiviert. Das ROM kann mit einer eigenen Routine das BASIC-ROM wieder einschalten.

Das Interface 1 muß vor dem Einsatz jedoch erst initialisiert werden. Eine Initialisierung ist nötig, wenn

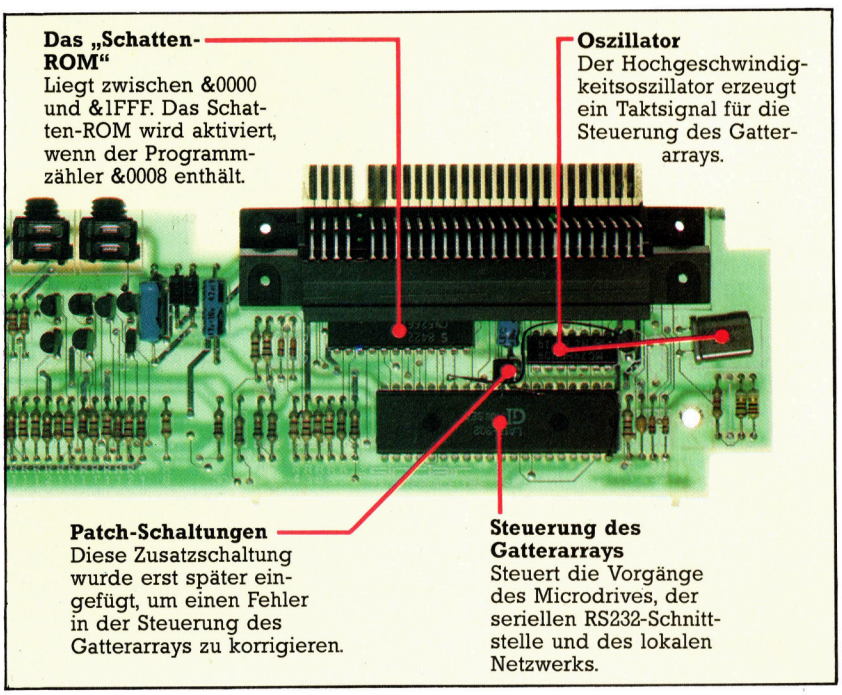
1) der Befehl NEW ausgegeben oder
2) das Schatten-ROM zum ersten Mal aktiviert wird.

Die Initialisierung legt im RAM eine neue Variablengruppe (etwa 50 Bytes) an, die vom Ende der alten Systemvariablen bis zum Anfang der Kanaldaten reicht. Da außerdem eine „Microdrive-Map" gebraucht wird und weitere Informationen über die von dem Interface 1 gesteuerten Peripheriegeräte nötig sind, verschiebt sich der Start der BASIC-Programmtexte nach oben. Aus diesem Grund sollten Sie Maschinencode auch nicht in REM-Befehlen der Zeile 1 unterbringen, da diese Zeile (bei eingebautem Interface 1) nicht mehr auf einer festen Speicherposition liegt.

Das Interface 1 enthält zusätzliche BASIC-Befehle und gibt Ihnen auch die Möglichkeit, eigene Befehle zu generieren. Auf einem Spectrum (ohne Interface 1) rufen die Befehle CAT und FORMAT Fehlermeldungen hervor, die über den Befehl RST &08 die Adresse &08 ansprechen. Mit dem Interface 1 aktiviert dieser Befehl jedoch das Schatten-ROM, das sich den „beanstandeten" BASIC-Befehl nochmals ansieht und versucht, ihn zu interpretieren. Der folgende Befehl speichert auf diese Weise das Programm „Fred" auf dem Microdrive:

```
SAVE *„m";1;„Fred"
```

Das Bild zeigt die Platine des Interface 1. Außer den Routinen zur Steuerung der Microdrives, der seriellen Schnittstelle und des LAN können Sie damit auch eigene BASIC-Befehle definieren. Wir gehen in einer späteren Folge genauer auf diesen Vorgang ein.



Der * erzeugt eine Fehlerbedienung, die die Steuerung an die Adresse &08 und damit an das Schatten-ROM übergibt. Dort erfolgt dann die korrekte Interpretation des Befehls.

Sehen wir uns nun die Maschinencoderoutinen an, die das Interface 1 zu bieten hat. Wir untersuchen zunächst die „Allzweckroutinen" – das heißt die Module des Schatten-ROMs, die nicht zur Steuerung der Microdrives, der seriellen Schnittstelle und des LAN eingesetzt und benötigt werden.

Routinen – aber wie?

Zunächst stellt sich die Frage, wie die Routinen überhaupt aufgerufen werden, da das Schatten-ROM doch nur nach Ansprechen einder der beiden zuvor erwähnten Adressen aktiv ist. Die Lösung ist ein Sprung auf die Adresse &08 mit dem Ablauf

```
RST &08
DEFB nn
```

wobei nn einen Wert zwischen &1B und &32 angibt (Werte außerhalb dieses Bereichs erzeugen eine Fehlermeldung). Die Werte von nn heißen „Hakencodes" (englisch: Hook Codes) und rufen unterschiedliche Routinen des Schatten-ROM auf. Die Auswirkungen der Hakencodes wurden in allen Versionen des Schatten-ROM unverändert beibehalten. Bevor wir uns einige dieser Aufrufe ansehen, hier noch ein paar Hinweise:

1) Da Vorgänge mit Hakencodes die Tendenz haben, Registerinhalte zu zerstören, müssen Sie vor dem Ablauf alle Register sichern, die Sie später noch einsetzen wollen. Auch das Registerpaar HL sollte gesichert werden, da es für die Rückkehr ins BASIC unbedingt nötig ist.

2) Einige Hakencodes schalten die Interrupts ab. Im Zweifelsfall sollten Sie sie wieder anschalten.

3) Beim Aufruf eines Hakencodes sollte der Wert &5C3A im Registerpaar IY stehen.

4) Vor dem Einsatz eines Hakencodes sollten Sie sicherstellen, daß die Systemvariablen des Interface aktiv sind. Diese Aufgabe kann der zuerst beschriebene Hakencode übernehmen.

- **Hakencode 49 (&31)** veranlaßt die Anlage der Systemvariablen für das Interface 1. Wenn Sie nicht genau wissen, ob die Variablen bereits angelegt sind, sollten Sie vor dem Einsatz anderer Hakencodes folgenden Aufruf durchführen:

```
RST 8
DEFB 49
```

Nach Ausführung eines Hakencodes durch das Schatten-ROM geht die Steuerung wieder an das Haupt-ROM zurück.

Das Schatten-ROM bietet aber auch Routinen, die (in komplizierterer Form) im BASIC ROM enthalten sind und die Tastatur-E/A und die Bildschirmausgabe steuern.

- **Hakencode 27 (&1B)** wartet auf die Eingabe eines Zeichens per Tastatur und übergibt den Tastencode an das Register A (das Zeichen erscheint nicht auf dem Bildschirm).
- **Hakencode 32 (&20)** prüft unmittelbar nach seinem Aufruf die Tastatur und zeigt über den Status des C-Flag an, ob eine Taste gedrückt wurde. Die Routine wartet jedoch nicht auf einen Tastendruck. Wurde eine Taste betätigt, dann steht das C-Flag auf Eins; wenn nicht, steht es auf Null.

Vor dem Aufruf der Hakencodes 27 und 32 müssen die Interrupts aktiviert werden, da der Spectrum das Prüfen der Tastatur über Interrupts steuert.

- **Hakencode 28 (&1C)** setzt das (beim Aufruf der Routine als ASCII-Code im A-Register gespeicherte) Zeichen in den Strom 2, der normalerweise dem oberen Teil des Bildschirms zugeordnet ist. Das folgende Programmodul zeigt den Einsatz dieser Routine und des Hakencodes 27. Beachten Sie, daß hier eine Endlosschleife vorliegt, die nur durch Abschalten der Maschine beendet werden kann.
- **Hakencode 31 (&1F)** ähnelt dem Hakencode 28, setzt das Zeichen (den ASCII-Code im A-Register) aber in Strom 3 (normalerweise dem ZX Drucker zugeordnet).
- **Hakencode 50 (&32)** spricht eine Allzweckroutine an. Der Code aktiviert vom Haupt-ROM aus Routinen des Schatten-ROMs über ihre direkten Adressen und nicht über den Hakencode. Die Routine verursacht jedoch einen Systemabbruch, wenn sie auf einer anderen ROM-Version eingesetzt wird. Hier ihr Einsatz:

```
      ;call routine in Shadow Rom - use with care!
210000      ld   hl,ADDRESS          ;insert address of routine
22ED5C      ld   (23789),hl          ;put address in system variable
CF          rst  #08
32          defb #32                 ;do it
```

Zwei Systemadressen des Interface 1 (23789 und 23790) müssen die Adresse enthalten, die die gewünschte Routine des Schatten-ROMs angibt. Danach wird der Hakencode folgendermaßen aufgerufen:

```
      ;wait for char, print to stream 2
      ;Hook Codes #1B and #1C
CF    start: rst  #08                ;set up Interface 1...
31           defb #31                ;...system variables
FB    loop:  ei                      ;ei - just in case
CF           rst  #08                ;wait for a char...
1B           defb #1B                ;...from the keyboard
CF           rst  #08                ;char now in A reg...
1C           defb #1C                ;...so print it
18F9         jr   loop               ;round again ad infinitum
```

Dieser Aufruf ist nur dann sinnvoll, wenn Sie wissen, mit welcher Version Sie arbeiten und welche Aufgaben die Routinen des Schatten-ROMs ausführen. Wir werden genauer darauf eingehen, wenn wir mit dem Interface 1 das BASIC um neue Befehle erweitern. In der nächsten Folge beschäftigen wir uns mit den Hakencodes, die die Microdrives steuern.

Stück für Stück

Im letzten Abschnitt haben wir uns mit den Prinzipien der Analog/Digital-(A/D) bzw. der Digital/Analogwandlung (D/A) befaßt. In dieser Folge geben wir einen Gesamtüberblick – alle Komponenten unseres Meßinstruments werden einzeln vorgestellt und erklärt.

Die Schaltung des Digitalvoltmeters besteht aus sieben Abschnitten bzw. Blocks (siehe Zeichnung). Wir wollen diese Komponenten im einzelnen genauer beschreiben.

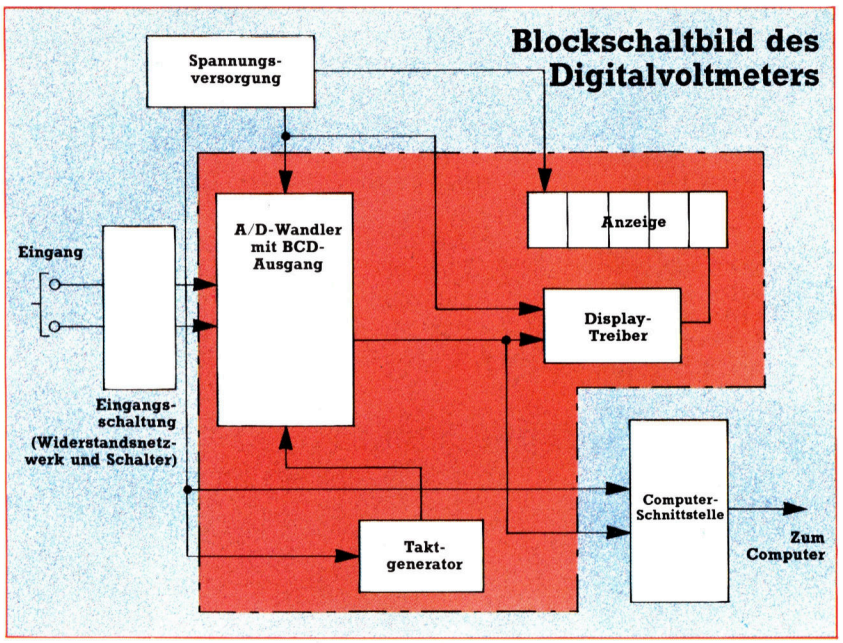
● **Eingangsabschwächer und Schalter:** Das Eingangssignal wird über Meßleitungen einem Widerstands- und Schalternetzwerk zugeleitet, bevor es zur eigentlichen Meßschaltung gelangt. Der Umfang dieser vorgeschalteten Einheit hängt davon ab, wie groß der Meßbereich des Instruments sein soll (etwa von Microvolt bis Kilovolt). Außerdem ist dafür entscheidend, welche elektrischen Größen gemessen werden sollen (Gleichspannung, Wechselspannung, Gleichströme, Widerstände usw.). Der Aufbau dieser Eingangsschaltung wird später noch genauer dargelegt.

● **Die Spannungsversorgung:** Dazu ist wenig zu sagen – es sind nur die Spannungen +5 Volt und –5 Volt nötig, wobei die Belastung recht klein ist. Wir können für diesen Zweck zei Festspannungsregler verwenden und entweder die Netzspannung oder Batterien einsetzen. Bis auf die Eingangsschaltung, die nur passive Bauelemente enthält, sind alle Systemkomponenten mit der Spannungsversorgung verbunden. Beim Aufbau muß sehr sorgfältig verdrahtet werden, um unerwünschte Masseschleifen zu vermeiden.

● **Die Taktgenerator-Schaltung:** Der A/D-Wandler 7135 braucht – wie fast alle anderen Wandler auch – externe Takt-Signale. Das Taktsignal kann sehr einfach mit rückgekoppelten TTL-Bausteinen (TTL=Transistor-Transistor-Logik) erzeugt werden. Wir haben uns für das vielseitige Timer-IC 555 entschieden. Der interne Aufbau ist recht kompliziert, zur Außenbeschaltung sind aber nur zwei Widerstände und ein Kondensator nötig.

● **Die Computer-Schnittstelle:** Diese Komponente ist ein Extra, auf das auch verzichtet werden kann. Unsere Schaltung ist als unabhängiges Gerät konzipiert und soll eine Alternative zu den teuren fertigen Digitalvoltmetern sein. Falls Ihr Computer aber Meßwerte aus seiner Umgebung verarbeiten soll – beispielsweise Spannungen, Widerstände, Temperaturen – können Sie die Schnittstelle nachrüsten.

● **Der A/D-Wandler:** Der Wandler ist „Herz" des Digitalvoltmeters. Er ist aus einem einzigen IC aufgebaut, das einen Standard-Meßbereich von zwei Volt hat (genauer gesagt 1,9999 Volt).



Das Diagramm verdeutlicht das Zusammenwirken der einzelnen Komponenten unseres Digital-Multimeters. Der rote Bereich des Blockdiagramms entspricht dem auf der nächsten Seite detailliert dargestellten Teil-Schaltplan.

Höhere Spannungen müssen von der Eingangsschaltung erst reduziert werden – am Wandler darf keine Spannung über zwei Volt liegen.

Vorteile beim Chip

Der verwendete Chip (IC 7135) ist anderen A/D-Konvertern in mehrfacher Hinsicht überlegen. Er kann die Eingangsspannung im BCD-Code (Binär Codierte Dezimalwerte) angeben und ist dadurch sehr viel leichter mit einem Computer zu verbinden als Wandlertypen, die für die Ansteuerung von Sieben-Segment-Anzeigen vorgesehen sind. Aber der Chip kann noch mehr: Das IC 7135 arbeitet sehr genau und liefert im Zwei-Volt-Bereich vier Dezimalstellen. Ein automatischer Nullabgleich sorgt dafür, daß ohne angeschlossene Spannung wirklich Null in der Anzeige steht.

Das IC 7135 verfügt über einen Ausweg für die Meldung einer Meßbereichs-Unterschreitung. In unserer einfachen Schaltung können wir ihn zur Ansteuerung einer LED gebrauchen; sie macht uns darauf aufmerksam, daß wir in einen kleineren Meßbereich umschalten können. In hochentwickelten Digitalvoltmetern werden die Meldeleitungen der Meßbereichs-über- und Unterschreitung für die selbsttätige Bereichswahl genutzt.

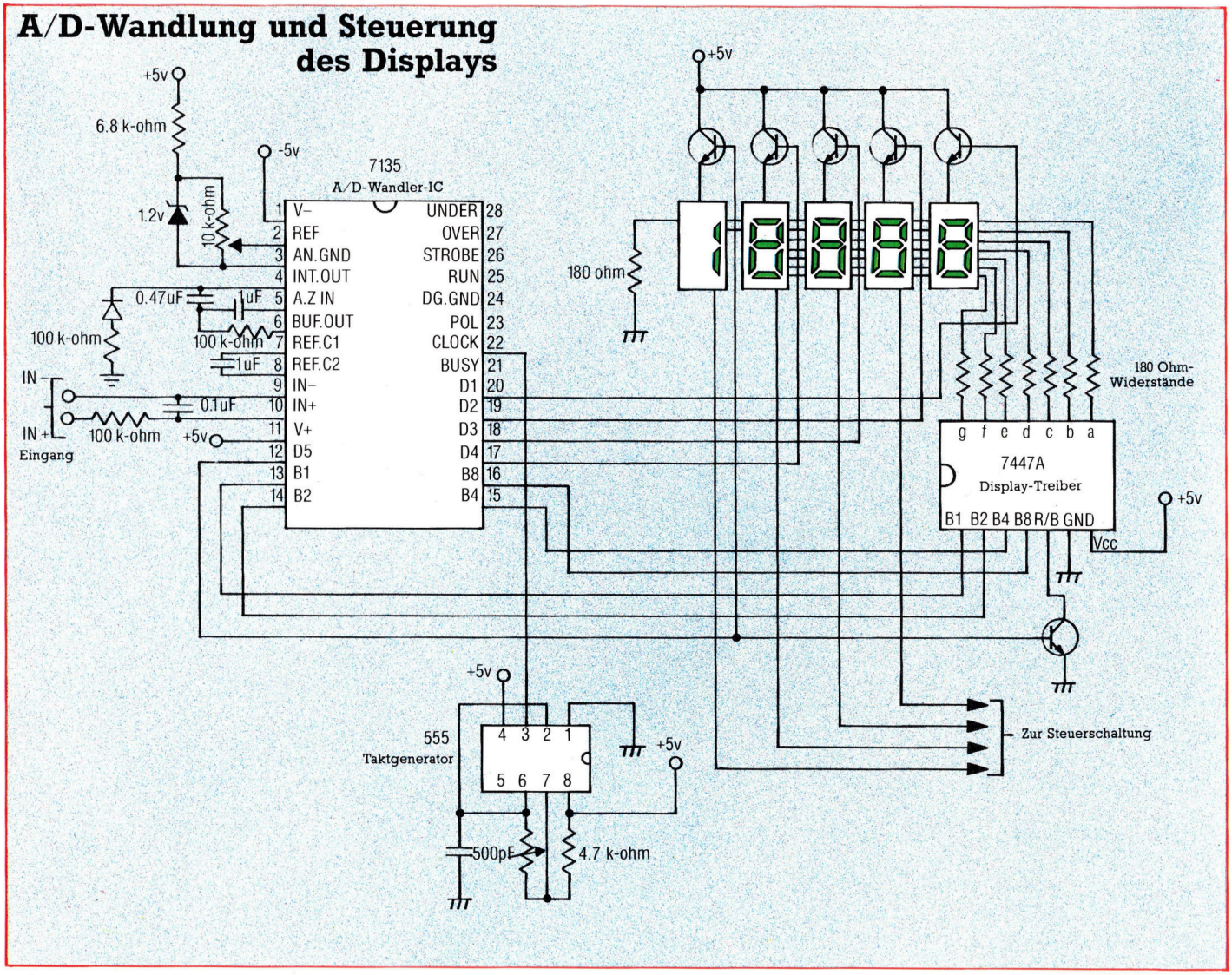
Die Grundlage unseres Digitalvoltmeters ist das A/D-Wandler-IC 7135. Das Bauteil wird extern von einem 555-Chip mit Taktsignalen versorgt. Dieser Baustein ist unten im Schaltbild dargestellt. Der 7135 gibt Signale im BCD-Code aus, die von einem 7447A-IC für die Darstellung über Sieben-Segment-Anzeigen aufbereitet werden. Durch fünf Schalttransistoren können alle fünf Anzeigefelder im Multiplex-Verfahren durch ein einziges Treiber-IC gesteuert werden. Da die Schaltung in dieser Form noch nicht vollständig ist, sollten Sie mit dem Bau aber noch nicht beginnen.

Die Ausgangsspannungen des ICs sind direkt TTL-kompatibel – das erleichtert die Verbindung zum Computer. Dazu liefert die Schaltung auch einige Signale, mit denen der Anschluß am Rechner einfacher wird. Dazu zählt eine STROBE-Leitung, die den Datentransfer zu externen Latches erleichtert (Latch=Zwischenspeicher für Daten, der ohne ständige Eingabe auskommt). Durch eine RUN/HOLD-Eingabeleitung kann vom „Einfrieren" des aktuellen Meßwertes auf freilaufenden (unabhängigen) Betrieb des Multimeters umgeschaltet werden. Die BUSY-Leitung meldet dem Rechner, ob Daten gültig sind oder nicht: Der Zustand der BUSY-Leitung gibt Auskunft darüber, ob die Daten auf den BCD-Leitungen ausgewertet werden können oder sich gerade im nicht definierten Zustand befinden.

● **Display-Treiber:** Der 7135-Wandler liefert am Ausgang ein BCD-Signal, das für die direkte Steuerung einer Anzeige nicht geeignet ist. Zum Glück gibt es einen preiswerten IC-Baustein, der ohne zusätzliche Beschaltung den BCD-Code für die Displaysteuerung umsetzt: Das IC 7447A liefert Signale, mit denen eine einzelne Sieben-Segment-Anzeige versorgt werden kann. Bei einer fünfstelligen Anzeige müssen wir im Multiplexverfahren arbeiten. Die vier BCD-Signale werden allen Anzeigen zugeleitet, sind aber jeweils nur für eine der Sieben-Segment-Anzeigen gültig. Welche das ist, wird durch Signalleitungen vom IC 7135 angegeben. Die Signale für die jeweils gültige Dezimalstelle werden verwandt, um über Transistoren nur je eins der Anzeigefelder zu aktivieren. Da die Sieben-Segment-Anzeigen sehr schnell hintereinander ein- und ausgeschaltet werden, nimmt das leicht zu trügende Auge ständig erleuchtete Ziffern wahr.

● **Das Anzeigefeld:** Zur Zeit sind LCDs (Flüssigkristallanzeigen) sehr aktuell, speziell für batteriebetriebene Geräte. Wir haben uns trotzdem für Leuchtdioden entschieden – sie haben einige Vorteile gegenüber ihren Konkurrenten: LEDs sind leichter anzuschließen, besonders was die Steuerung des Dezimalpunktes angeht. Flüssigkristall-Anzeigen benötigen außerdem ein Wechselstrom-Signal und zusätzliche Logik-Schaltungen für die Positionierung des Dezimalpunkts.

A/D-Wandlung und Steuerung des Displays



Attraktiver Ausbau

Der Mangel an Schnittstellen war ein „Geburtsfehler“ beim Acorn Electron. Die Interfacebox „Plus 1“ erlaubte dann zwar schon einen gewissen Ausbau, aber Diskettenbetrieb ist erst mit dem hier vorgestellten Erweiterungsmodul „Plus 3“ möglich.

An das Disketten-Erweiterungsmodul „Plus 3“ für den Acorn Electron läßt sich auch noch die Interfacebox „Plus 1“ anschließen. In dieser Konfiguration kommt das System ausstattungsmäßig dem Acorn B nahe.

Die karge Schnittstellenausstattung war beim Acorn Electron oft Gegenstand massiver Kritik. Zwar verfügt der Electron über das ausgezeichnete BBC-BASIC, aber er bietet außer Monitorbuchsen (RGB/FBAS/Antenne) und einem Cassettenrecorder-Interface keinerlei Peripherie-Anschlüsse, nicht einmal einen Joystick-Port. So war der Ausbau des Rechners anfangs sehr problematisch. Acorn kündigte wiederholt Erweiterungsmodule an, die das Schnittstellenangebot auf das Niveau des Acorn B bringen sollten, aber ihre Einführung erfolgte mit großen Verzögerungen.

Als erste Ergänzung kam das Interface „Plus 1“ heraus, das Kritiker schon erheblich besänftigen konnte. Trotz der damit verfügbaren Schnittstellen fehlte aber immer noch die Möglichkeit des Diskettenbetriebs, also auch ein brauchbarer Massenspeicher.

Dieser Mißstand wurde erst durch das Interface „Plus 3“ mit integriertem Diskettenlaufwerk beseitigt. Das Modul ist L-förmig, wobei die Steuerung mit dem Diskettenbetriebssystem im längeren Schenkel hinter dem Rechner untergebracht ist und das Laufwerk rechts daneben, so daß es die Rechnerbuchse für den Netzteilanschluß verdeckt. Deshalb gehört zum „Plus 3“ eine eigene Stromversorgung, die Rechner und Zusatzmodul gleichzeitig speist.

Zweites Laufwerk möglich

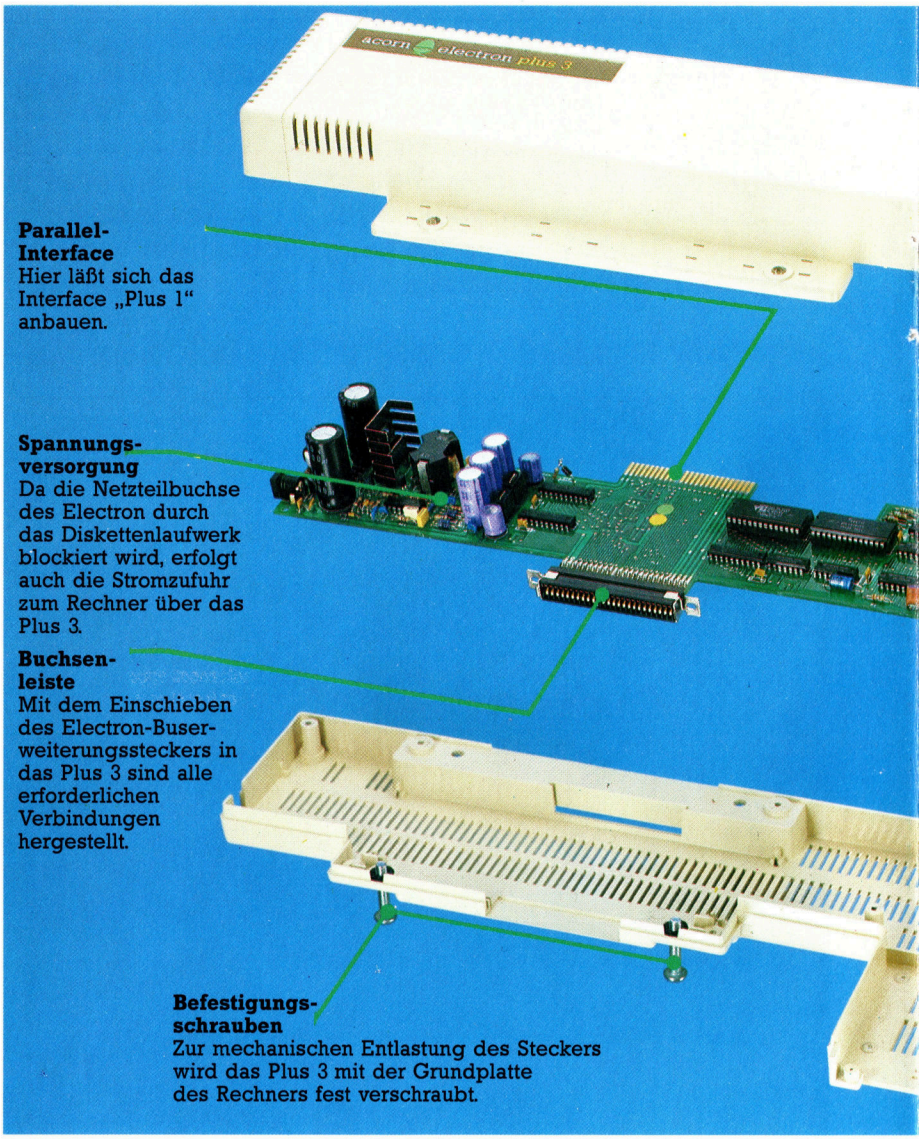
Wie das Interface „Plus 1“ wird auch das „Plus 3“ auf den rückwärtigen Platinenstecker des Electron aufgesetzt. Damit nichts wackelt oder gar bricht, ist der Zusatz mit zwei kräftigen Schrauben an der Grundplatte des Rechners zu befestigen. Im Unterschied zum „Plus 1“ hat das „Plus 3“ selbst hinten wieder einen Erweiterungsstecker, der den Anschluß der Interfacebox „Plus 1“ erlaubt.

Vielleicht überrascht es, daß Acorn sich beim „Plus 3“ für die 3 1/2-Zoll-Sony-Disketten und nicht für das übliche 5 1/4-Zoll-Format entschieden hat. An der Laufwerk-Rückseite gibt es allerdings noch eine weitere Steckleiste für

ein zweites Laufwerk. Für den Anschluß des 5 1/4-Zoll-Laufwerks des Acorn B ist als Zubehör ein spezielles Interface lieferbar.

Das Innenleben des „Plus 3" besteht aus vier Einheiten: Ganz links liegt die Spannungsversorgung, die an den großen Kondensatoren zu erkennen ist; daneben der Electron-Busanschluß, von dem aus einige Leiterbahnen zur Laufwerksteuerung führen, während andere für die Verbindung zum „Plus 1" sorgen. Danach folgt das Disketten-Betriebssystem mit den ROM-gespeicherten Routinen für das Ansprechen des Laufwerks. Zum Schluß kommt ganz rechts das Laufwerk selbst in einem Metallgehäuse, das als thermischer Schutz und auch als Abschirmung gegen Streufelder dient.

Die Sony-Disketten haben eine Kunststoffhülle mit einer verschiebbaren Abdeckung über dem Schreib/Lese-Fenster. Beim Formatieren werden 80 Spuren mit 16 Sektoren angelegt. Da jeder Sektor 256 Byte an Information enthält, ergibt sich eine Gesamtkapazität von 320 KByte.



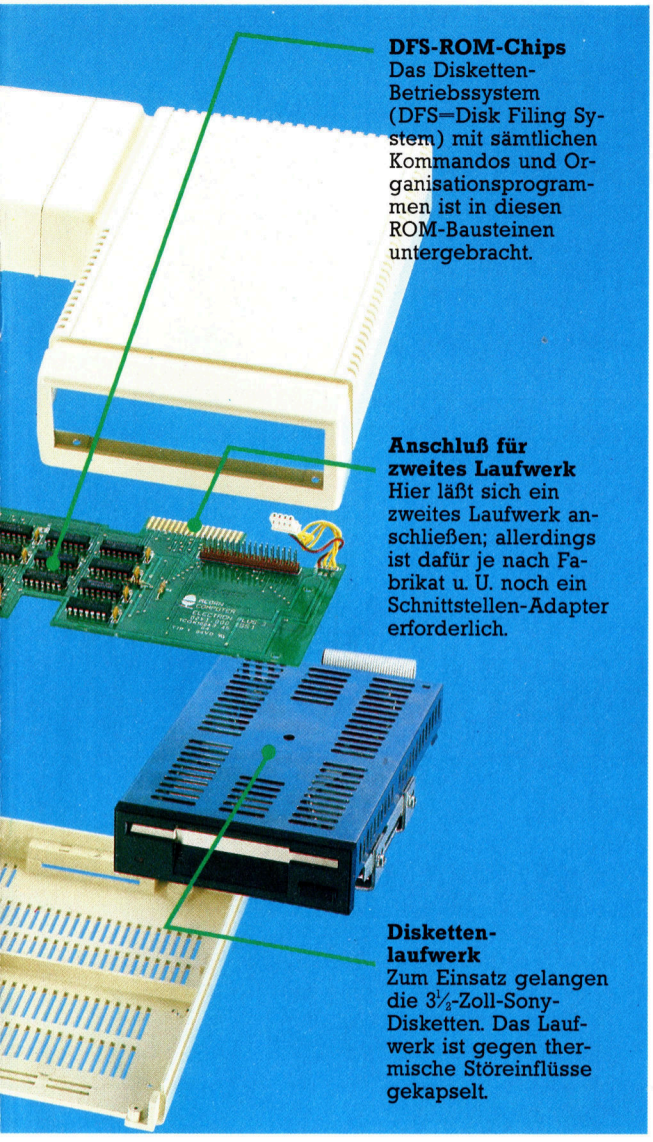
Kleine Unterschiede vorhanden

Nach dem Anschluß des „Plus 3" und dem Einschalten des Rechners erscheint auf dem Bildschirm außer der normalen Kopfzeile die Mitteilung ACORN ADFS (Advanced Disk Filing System = Fortschrittliches Diskettensystem). Obwohl das Electron-DFS in vieler Hinsicht eng mit dem des Acorn B verwandt ist und die meisten Kommandos identisch sind, gibt es beim Betrieb kleine Unterschiede. Mit dem „Plus 3" wird eine „Welcome"-Diskette mit Demonstrationsprogrammen geliefert, ähnlich der Begrüßungskassette beim Electron. Die Diskette enthält weiterhin eine Anzahl von Systemprogrammen, die für die Anwendung des DFS äußerst nützlich sind. Dazu gehören BACKUP für das Duplizieren einer ganzen Diskette, DIRCOPY für das Kopieren einzelner Dateien von einer Diskette zur anderen, BUILD für den Aufbau von BOOT-Routinen auf Diskette und DUMP zum Ausdrucken von Hex- oder ASCII-Dateien.

Wie beim Acorn B ist auch hier den Diskettenbefehlen ein „*" voranzustellen; sie haben also die vertraute Form *LOAD, *RUND, *DELETE usw. Das über *CATalogue abrufbare Dateiverzeichnis wird mit Diskettenname, Laufwerknummer, Optionen, aktuellen Directory- und Library(Unterverzeichnis)-Namen und dann auch mit der eigentlichen Datei-Namensliste ausgegeben.

Der wesentliche Unterschied zum Acorn B besteht in der Art der Initialisierung einer Diskette nach dem Einschieben. Beim „Plus 3" muß der Benutzer zunächst den Befehl *MOUNT (Vorbereiten) eintippen und damit die aktuelle Directory (CSD=Currently Selected Directory) und das aktuelle Unterverzeichnis (CSL=Curr. Sel. Library) in den Arbeitsspeicher laden. Vor einem Diskettenwechsel ist dann *DISMOUNT einzugeben, um alle noch offenen Dateien abzuschließen und CSD sowie CSL zu löschen. Nach dem Wechsel geht es wieder mit *MOUNT los. Abgesehen von diesen Zusatzkommandos funktioniert das Betriebssystem aber genau wie beim Acorn B.

Das ADFS arbeitet mit einem untergliederten Dateiverzeichnis, so daß einzelne Files schnell und einfach über Untertitel erreichbar sind. Ganz oben in der Hierarchie steht der Name der aktuellen Directory (CSD), wofür nach dem Einschalten automatisch „$" gesetzt wird. Um von der CSD zu einem andern Dateiverzeichnis auf der eingeschobenen Diskette überzugehen, braucht der Benutzer nur *DIR und den Namen der gewünschten Directory einzugeben. Unter dem CSD-Titel können nun entweder unmittelbar die Dateien angelegt werden, oder es werden Zwischentitel (Sub-Directories) eingeschoben, die dann jeweils für eine Datei-„Bibliothek" (Library) zuständig sind. Da eine Directory nicht mehr als 47 Dateien aufnehmen kann, ist eine Aufteilung in „Libraries" zumindest dann erforderlich, wenn Sie sehr viele Files auf einer Diskette unterzubringen haben – die Un-



DFS-ROM-Chips
Das Disketten-Betriebssystem (DFS=Disk Filing System) mit sämtlichen Kommandos und Organisationsprogrammen ist in diesen ROM-Bausteinen untergebracht.

Anschluß für zweites Laufwerk
Hier läßt sich ein zweites Laufwerk anschließen; allerdings ist dafür je nach Fabrikat u. U. noch ein Schnittstellen-Adapter erforderlich.

Diskettenlaufwerk
Zum Einsatz gelangen die 3½-Zoll-Sony-Disketten. Das Laufwerk ist gegen thermische Störeinflüsse gekapselt.

Acorn-ADFS-Mitteilungen
Das Electron-Plus 3-Betriebssystem „ADFS" entspricht weitgehend dem „DFS" für den Acorn B. Oben sehen Sie, wie das ADFS auf den *CAT-Befehl reagiert. Unten wurde über das Kommando *INFO die Ausgabe von Dateinamen, Zugriffsbeschränkungen usw. angefordert.

Acorn Electron Plus 3

ABMESSUNGEN:
ca. 465 × 200 ×50 mm

KAPAZITÄT:
320 KByte pro Diskette (einseitig)

SCHNITTSTELLEN:
Buchsenleiste für Rechneranschluß; Platinenstecker für Interface Plus 1 und zweites Laufwerk

DOKUMENTATION:
Das mitgelieferte ausführliche Handbuch erlaubt dem Anfänger ein schrittweises Kennenlernen der Möglichkeiten des Plus 3.

STÄRKEN:
Wer seinen Acorn-Electron ausbauen will, ist mit der Diskettenstation „Plus 3" einschließlich Betriebssystem gut bedient.

SCHWÄCHEN:
Bisher gibt es wenig Software für das System. Außerdem ist für die volle Nutzung des Diskettenlaufwerks ein Drucker fast unentbehrlich, und für dessen Betrieb ist zusätzlich die Anschaffung der Interfacebox „Plus 1" erforderlich.

terverzeichnisse können insgesamt wesentlich mehr Dateinamen als die CSD beinhalten. Um bei dieser Struktur etwa die Datei „Meier" auf einer Diskette zu finden, wäre dann $.library1.meier. einzugeben. Das erscheint ziemlich umständlich, hat aber den Vorteil, daß Sie in jedem Unterverzeichnis eine Datei unter dem Namen „Meier" führen können. Wenn Sie dann beispielsweise Dateien mit SAVE zurückspeichern und in der gleichen Library geblieben sind, werden die anderen „Meier"-Files auf der Diskette nicht überschrieben.

Vielzahl von Befehlen

Wie das DFS beim Acorn B bietet auch das Electron-ADFS eine Reihe nützlicher Dienstprogramme. So gibt es Befehle zum Benennen und Umbenennen von Disketten, Directories oder Dateien und außerdem mehrere Kommandos für die Einschränkung des Datei-Zugriffs. Sie können beispielsweise durch Eingabe von *ACCESS file1 RL eine Datei gegen versehentliches Löschen oder Überschreiben sperren, wobei das Lesen jederzeit möglich bleibt. Über *ACCESS file1 WR läßt sich die Datei wieder entsperren. Um zu verhindern, daß jemand ein Maschinencode-File von der Diskette kopiert, brauchen Sie der Datei nur ein „E" anzuhängen – dann wird für sie nur noch das Kommando *RUN akzeptiert, und auch das Löschen ist nun nicht mehr ohne weiters möglich.

Das Handbuch zum Plus 3 ist sehr ausführlich und enthält eine auch für Anfänger gut geeignete, umfangreiche Anleitung. Die Acorn-Dokumentationen haben ja durchweg einen guten Ruf am Markt.

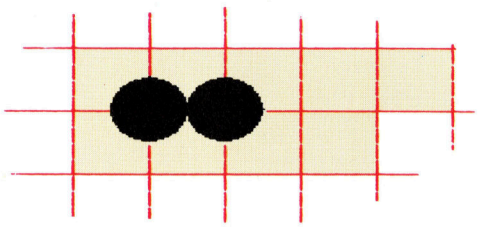
Seit der Markteinführung ist der Electron in der Acorn-Familie immer nur als zu kurz gekommener kleiner Bruder des Acorn B betrachtet worden. Wegen der fehlenden notwendigen Schnittstellen, insbesondere für ein Diskettenlaufwerk, erschien der Electron-Rechner eigentlich nur für Spiele geeignet. Mit der Preissenkung für den Rechner selbst und der Verfügbarkeit des Plus 1 und Plus 3 ist der Acorn Electron aber sehr viel attraktiver geworden und kann als ein durchaus ernstzunehmender Rechner gelten, mit dem man arbeiten und auch Spaß haben kann.

Verbindungen

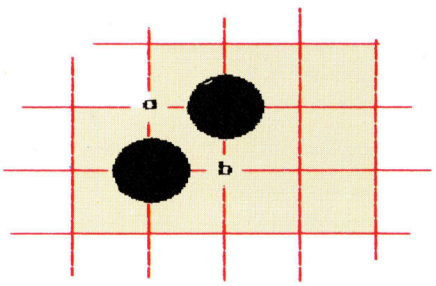
Wir befassen uns jetzt mit dem vielleicht wichtigsten Punkt des Go-Spieles – der Verknüpfung von Steingruppen. Dazu müssen alle hierfür notwendigen Routinen verbunden werden.

Bei Spielende gilt ein Gebiet nur dann als eingeschlossen, wenn es durch eine ununterbrochene Steinreihe eingekreist ist. Dies können gerade (siehe Bild 1) oder diagonale Verbindungen (siehe Bild 2) sein, wobei, wenn Weiß auf „a" setzt, Schwarz durch Setzen auf „b" zwei ununterbrochene Verbindungen bilden kann und umgekehrt. Normalerweise entstehen solche festen Verbindungen erst gegen Spielende.

1

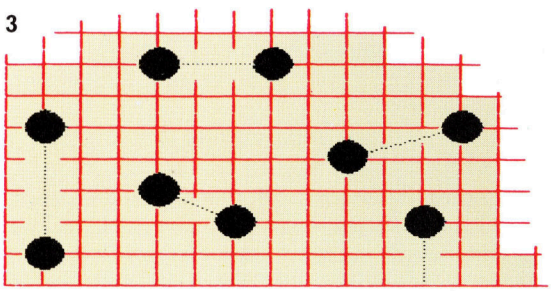


2



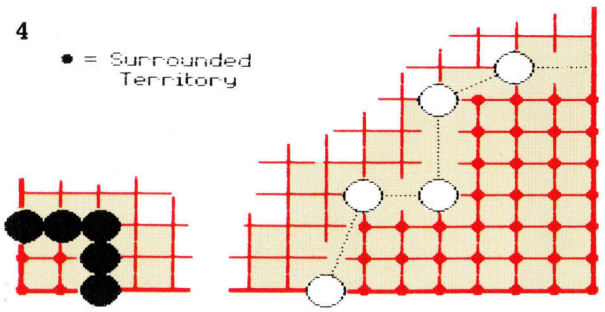
Meistens werden während des Spieles die Steine durch ein oder mehrere Leerfelder getrennt sein, wobei imaginäre Verbindungen mit anderen Steinen oder dem Brettrand bestehen (Bild 3). In diesen Fällen formt man feste Verbindungen nur, wenn die imaginären durch den Gegner unterbrochen werden könnten.

3



Wenn wir wollen, daß unser Go-Programm vernünftig spielt, müssen wir diese flexiblen Verbindungen berücksichtigen. Bringen wir dem Programm nur bei, daß feste Verbindungen mit aneinandergrenzenden Steinen zur Gebietseinkreisung verwendet werden können, wird es sicherlich Probleme geben. In Bild 4 etwa hat Weiß, während das Programm (Schwarz) vier Punkte eingeschlossen hat, mit derselben Steinanzahl ein Gebiet mit 37 Punkten umkreist!

4



Um dem Computer diese Verbindungen „beizubringen", müssen verschiedene Punkte berücksichtigt werden. Insgesamt gibt es vier Verbindungsmöglichkeiten, die wir behandeln wollen. Dies sind:

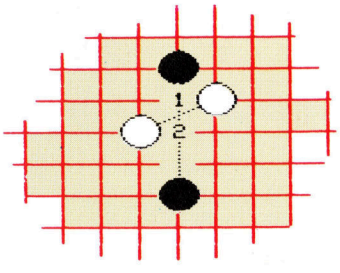
– Verteidigung einer Verbindung zwischen zwei Gruppen, wenn der Gegner auf eine mögliche Angriffsposition setzt.

– Angriff einer gegnerischen Verbindung durch Setzen eines Steines auf die imaginäre Verbindung.

– Spiel eines „start-attack"-Steines. Dies ist entweder eine Vorbereitung zu einem Angriff durch Setzen eines Zentralsteines oder Rückkendeckung für einen zuvor bereits gesetzten Zentralstein.

– Starten einer Verbindung durch Setzen eines Steines in gewissem Abstand von einem zuvor gesetzten Stein. Durch Verwendung des Bewertungssystems, das wir im letzten Artikel entwikkelt haben, wird automatisch eine vorteilhafte Position gefunden.

5

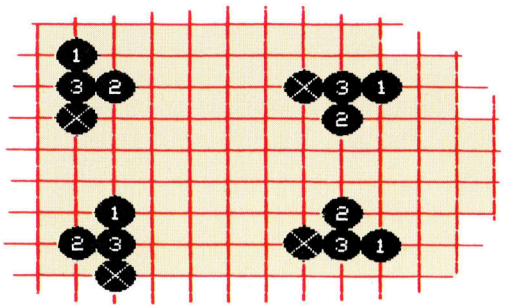


Diese Operationen wurden unter Verwendung einfacher Vergleichs-Techniken programmiert. Im Bild 5 will Weiß durch Spiel auf Position „2" die Verbindung der schwarzen Steine unterbrechen, und Schwarz will dies durch Setzen auf

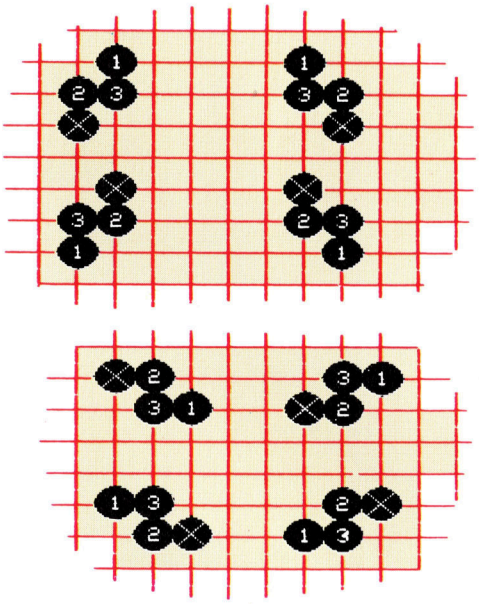
dieselbe Position verhindern. Gleichzeitig will Schwarz die weiße Verbindung angreifen, während Weiß ihre Solidität durch Spiel auf diese Punkte verteidigt.

Es ist möglich, alle vier Verbindungs-Operationen unter Verwendung eines Mustersatzes zu implementieren. Diese Muster werden durch die Routine bei Zeile 790 initialisiert, wobei die in den Bildern 6 bis 9 gezeigten Muster definiert werden. In allen Fällen repräsentiert der angekreuzte Stein den aktuellen Stein.

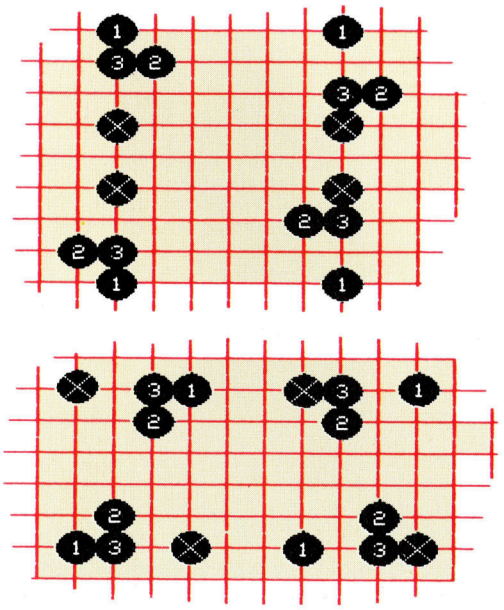
6



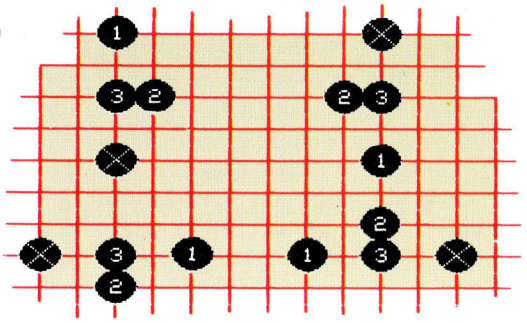
7



8



9



Da theoretisch jeder Stein der aktuelle sein könnte, ist es nicht notwendig, alle Richtungen für ein Muster zu berücksichtigen. In Bild 9 beispielsweise verfügen wir nicht über ein Muster mit dem aktuellen Stein ganz unten (0), dem ersten Stein oben (+64), und Stein 2 links von Stein 3 (+31 und +32). Zu einem späteren Zeitpunkt der Suche wird jedoch der momentan mit 1 markierte Stein als aktueller Stein angesehen. Das Muster oben rechts in Bild 9 ist dann für die eben beschriebene Richtung zuständig.

Sie werden bemerken, daß jede zweite DATA-Anweisung eine Umkehrung der vorherigen Zeile ist, so daß wir bei mangelndem Speicherplatz die Anzahl der Muster noch reduzieren können, speziell durch den Versuch der Umkehrung aller gespeicherten Muster.

Die Muster-Vergleichsroutinen berechnen die Brett-Positionen dieser Vorgaben und finden die Steinfarbe an den relativen Positionen. Mit diversen Regeln wird dann entschieden, ob ein Stein gesetzt wird oder nicht. Soll ein Stein gespielt werden, wird der Position eine Bewertung zugeordnet. Am Ende dieser Routine wird der Stein mit der höchsten Bewertung gesetzt.

Die Routinen werden dann der Reihe nach überprüft.

Angenommen, Schwarz wäre am Zug, lauteten die Regeln zum Setzen eines Steines wie folgt:

```
defend_connection:
   IF     angekreuzter Stein  = SCHWARZ
   AND    Stein eins          = SCHWARZ
   AND    Stein zwei          = WEISS
   THEN   versuche Zug bei Stein drei.
attack_connection:
   IF     angekreuzter Stein  = WEISS
   AND    Stein eins          = WEISS
   AND    Stein zwei          = SCHWARZ
   THEN   versuche Zug bei Stein drei.
start_attack:
   IF     angekreuzter Stein  = WEISS
   AND    Stein eins          = WEISS
   AND    Stein drei          = LEER
   THEN   versuche Zug bei Stein zwei.
start_connection:
   IF     angekreuzter Stein  = SCHWARZ
   AND    Stein drei          = LEER
   THEN   versuche Zug bei Stein eins.
```

Die vier hier gezeigten Verbindungsroutinen werden von den Zeilen 2580 bis 2610 (blackmove Routine) aufgerufen.

Somit können jetzt Verbindungen zwischen Steinen auf dem Brett gehandhabt werden. Die Bilder 3 und 4 zeigen, daß es auch Verbindungen zwischen Steinen und den Bretträndern gibt, die ebenso wichtig sind. Für diese Situationen gibt es die PROCboundary-Routine. Wie Sie sich erinnern, ist das Brett mit DIM board%256 auf ein Brett mit 15 × 15 Feldern definiert. Dies ergibt 225 Bytes für das Brett und einen aus einzelnen Feldern bestehenden Rand um das Brett (insgesamt +256 Bytes). PROCboundary füllt diesen Rand mit dem als Parameter über V% übergebenen Wert. Da die Vergleichsroutinen das ganze Brett untersuchen, einschließlich Rand (von 1 bis 255), werden auch Verbindungen mit den Ecken des Brettes überprüft. Denken Sie aber immer daran, bei Verlassen der Routine den Rand wieder auf Null zu setzen, indem Sie PROCboundary mit dem Wert 0 aufrufen.

Wegen der Verarbeitungsgeschwindigkeit wurde die Anzahl der Muster minimal gehalten. Trotzdem können Sie die Daten ab Zeile 860 beliebig ändern. Vergessen Sie dann aber nicht, auch die Schleifenparameter in den Zeilen 2940, 3110, 3260 und 3410 anzupassen.

Fünftes Modul

Acorn B:

```
  280 PROCread_patterns
  780 :
  790 DEF PROCread_patterns
  800 LOCAL L%
  810 DIM pat% 71
  820 RESTORE 860
  830 FOR L% = 0 TO 71
  840   READ pat%?L%
  850   NEXT
  860 DATA 32,17,16,2,-15,1
  870 DATA -32,-17,-16,-2,15,-1
  880 DATA 33,16,17,31,16,15
  890 DATA -33,-16,-17,-31,-16,-15
  900 DATA -14,1,-15,18,1,17
  910 DATA 14,-1,15,-18,-1,-17
  920 DATA 48,33,32,48,17,16
  930 DATA -48,-33,-32,-48,-17,-16
  940 DATA 3,-14,2,3,-15,1
  950 DATA -3,14,-2,-3,15,-1
  960 DATA 64,33,32,4,-14,2
  970 DATA -64,-33,-32,-4,14,-2
  980 ENDPROC
  990 :
 1000 REM*******************************
 2580 IF location%=0 THEN PROCdefend_con
nection:T$="DEF"
 2590 IF location%=0 THEN PROCattack_con
nection:T$="ATT"
 2600 IF location%=0 THEN PROCstart_atta
ck:T$="SAT"
 2610 IF location%=0 THEN PROCstart_conn
ection:T$="SCN"
 2890 :
 2900 DEF PROCdefend_connection
 2910 LOCAL A%,B%,C%,D%,P%,S%,hi,score
 2920 PROCboundary(black%)
 2930 FOR A%=1 TO 255 : S%=board%?A% : I
F S%<>black% THEN 3010
 2940   FOR P%=pat% TO pat%+70 STEP 3
 2950     B%=board%?((A%+?P%) AND 255):C
%=board%?((A%+?(P%+1)) AND 255)
 2960     IF B%<>black% OR C%<>white% TH
EN 3000
 2970     D%=(A%+?(P%+2)) AND 255:score=
RND(1)+weight%?D%
 2980     IF (D% AND 240)=0 OR (D% AND 1
5)=0 OR score<=hi THEN 3000
 2990     IF FNlegality(D%,black%)=0 AND
 clib%>2 THEN hi=score:location%=D%
 3000     NEXT
 3010   NEXT
 3020 PROCboundary(0)
 3030 ENDPROC
 3040 :
 3050 REM*******************************
 3060 :
 3070 DEF PROCattack_connection
 3080 LOCAL A%,B%,C%,D%,P%,S%,hi,score
 3090 PROCboundary(white%)
 3100 FOR A%=1 TO 255 : S%=board%?A% : I
F S%<>white% THEN 3180
 3110   FOR P%=pat% TO pat%+70 STEP 3
 3120     B%=board%?((A%+?P%) AND 255):C
%=board%?((A%+?(P%+1)) AND 255)
 3130     IF B%<>white% OR C%<>black% TH
EN 3170
 3140     D%=(A%+?(P%+2)) AND 255:score=
RND(1)+weight%?D%
 3150     IF (D% AND 240)=0 OR (D% AND 1
5)=0 OR score<=hi THEN 3170
 3160     IF FNlegality(D%,black%)=0 AND
 clib%>2 THEN hi=score:location%=D%
 3170     NEXT
 3180   NEXT:PROCboundary(0)
 3190 ENDPROC
 3200 :
 3210 REM*******************************
 3220 :
 3230 DEF PROCstart_attack
 3240 LOCAL A%,B%,C%,D%,P%,S%,hi,score
 3250 FOR A%=1 TO 255 : S%=board%?A% : I
F S%<>white% THEN 3330
 3260   FOR P%=pat% TO pat%+70 STEP 3
 3270     B%=board%?((A%+?P%) AND 255):D
%=board%?((A%+?(P%+2)) AND 255)
 3280     IF B%<>white% OR D%<>0 THEN 33
20
 3290     C%=(A%+?(P%+1)) AND 255:score=
RND(1)+weight%?C%
 3300     IF (C% AND 240)=0 OR (C% AND 1
5)=0 OR score<=hi THEN 3320
 3310     IF FNlegality(C%,black%)=0 AND
clib%>2 THEN hi=score:location%=C%
 3320     NEXT
 3330   NEXT
 3340 ENDPROC
 3350 :
 3360 REM*******************************
 3370 :
 3380 DEF PROCstart_connection
 3390 LOCAL A%,B%,C%,D%,P%,S%,hi,score
 3400 FOR A%=1 TO 255 : S%=board%?A% : I
F S%<>black% THEN 3470
 3410   FOR P%=pat% TO pat%+70 STEP 3
 3420     C%=board%?((A%+?(P%+2)) AND 25
5):IF C%>0 THEN 3460
 3430     B%=(A%+?P%) AND 255:score=RND(
1)+weight%?B%
 3440     IF (B% AND 240)=0 OR (B% AND 1
5)=0 OR score<=hi THEN 3460
 3450     IF FNlegality(B%,black%)=0 AND
 clib%>2 THEN hi=score:location%=B%
 3460     NEXT
 3470   NEXT
 3480 ENDPROC
 3490 :
 3500 REM*******************************
 4410 :
 4420 DEF PROCboundary(V%)
 4430 LOCAL X%,Y%
 4440 FOR X%=0 TO 15
 4450   Y%=16*X%
 4460   board%?X%=V%
 4470   board%?Y%=V%
 4480   NEXT
 4490 ENDPROC
 4500 :
 4510 REM*******************************
 4520 REM*******END OF PROGRAM***********
```

Der Standard

MS-DOS von Microsoft hat sich zum Standard für 16-Bit-Maschinen entwickelt und bildet die Grundlage für das vielkopierte IBM PC-DOS.

In der Anfangstagen der Acht-Bit-Micros waren der Z80 von Zilog in Verbindung mit dem Betriebssystem CP/M Marktführer. Gary Kildall – der Gründer von Digital Research – hatte aber das Betriebssystem CP/M ursprünglich für den Intel 8080 entwickelt.

Als sich IBM dem Microcomputer widmete, neigte sich die Ära der Acht-Bit-Systeme zwar ihrem Ende zu, doch gab es noch keine Kombination von CPU und Betriebssystem, die die Nachfolge von Z80 und CP/M angetreten hatte. Für IBM bestand daher die Möglichkeit, mit ihren 16-Bit-Maschinen eine völlig neue Hard- und Softwarekombination einzuführen. Mehrere Gründe sprachen jedoch dagegen:

1) Trotz beherrschender Stellung in der Groß-EDV besaß IBM keine Erfahrung in der völlig andersartigen Welt der Microcomputer.

2) Der Erfolg von DEC (Digital Equipment Corporation) im Bereich der Minicomputer hatte IBM gezeigt, daß Anwender nicht mehr bereit waren, IBM-Produkte blind zu kaufen.

3) Ein völlig neues System hätte viel Zeit für die Entwicklung einer Softwarebasis gebraucht. Ohne Unterstützung vieler unabhängiger Softwarehäuser hätte sich IBM in der sich schnell verändernden Microcomputerwelt nur schwer behaupten können.

4) Hätte IBM Hard- und Software selbst entwikkelt, wäre das Endprodukt für Anwender nicht mehr erschwinglich gewesen.

IBM entschied sich daher, das „Beste" der bekannteren Betriebssysteme zu nehmen und es an seine 16-Bit-Umgebung anzupassen.

Nichts Neues im Vergleich

Das Ergebnis dieses konservativen Konzeptes war ein recht langsamer IBM PC, der im Vergleich mit Maschinen der gleichen Generation nichts Neues bot. Selbst die Wahl des Hauptprozessors Intel 8088 und die damit verbundene Entscheidung für die Chipfamilie des 8086 (8086/186/286/386) hatte rein praktische Gründe. Der 8088 arbeitete intern mit einer 16-Bit Struktur, besaß aber einen Acht-Bit externen Datenbus und konnte so die bewährten (und billigen) Peripheriechips im Acht-Bit-Format einsetzen. Die Anlage lief langsam, da jedes 16-Bit „Wort" in zwei Teilen geladen werden mußte.

Inkompatibel

Hauptursache erster Kompatibilitätsprobleme des IBM PC war die Verlegung der Sprungtabelle um zwei Speicherseiten nach „unten". Die Tabelle befand sich nun bei der ROM Adresse 400H (die MS-DOS Tabelle beginnt bei 600H). Dieser Teil des BIOS (Basic Input/Output System) wurde von IBM urheberrechtlich geschützt und stellte Firmen vor Probleme, die hundertprozentig kompatible Nachbauten des IBM PC herstellen wollten. Dennoch lief Software, die die Hardware über die korrekten Systemaufrufe des MS-DOS ansprach, auf allen MS-DOS Geräten. Diese Kompatibilität hat jedoch ihren Preis. Die Systemaufrufe verlangsamen die Ausführgeschwindigkeit und verhindern den Einsatz späterer Hardwareverbesserungen. Aus diesen Gründen entwickelten etliche Softwarehäuser hard- (oder firmware-) abhängige Programme, die nur in einer dem IBM PC sehr ähnlichen Hardwareumgebung liefen. Einige Softwarehersteller machten ihre Produkte sogar absichtlich inkompatibel. So ändert Digital Research beispielsweise bei GEM-Anwendungen einige wenige Codebytes, die für jeden Maschinentyp den Kauf einer anderen Version nötig machen – selbst bei hundertprozentig kompatiblen Nachbauten ist dies notwendig.

„Quick and Dirty"

Für Betriebssystem und Software kontaktete IBM die Firma Microsoft, die hauptsächlich für ihren leistungsfähigen BASIC-Dialekt bekannt war. Microsoft hatte zu diesem Zeitpunkt Kontakt zu „Seattle Computer Products", die auf der Basis von Intel-Chips 16-Bit-Systeme entwikkelte. Der Programmierer Tim Patterson hatte für die 8086-Familie ein Maschinencode DOS (SCP-DOS) geschrieben, das sich eng an CP/M lehnte. Sein System wurde auch QDOS (nicht zu verwechseln mit dem Sinclair OS für den QL) genannt, wobei die Abkürzung „Quick and Dirty Operating System" – „schnell und unsauber" geschriebenes OS – bedeutete. Die Beschreibung traf zu: Pattersons OS funktionierte zwar, war aber nicht durchkonstruiert und hatte Fehler. Microsoft kaufte nun die Rechte für dieses System und gab es in Lizenz für Original Equipment Manufacturers (OEM) – Hersteller von Geräten, die Bauteile anderer Hersteller enthalten – heraus.

Aus der MS-DOS Version (Version 1) entwikkelte IBM für den PC ein eigenes System: PC-DOS. Bei jeder wichtigen Weiterentwicklung von MS-DOS erschien eine entsprechende PC-

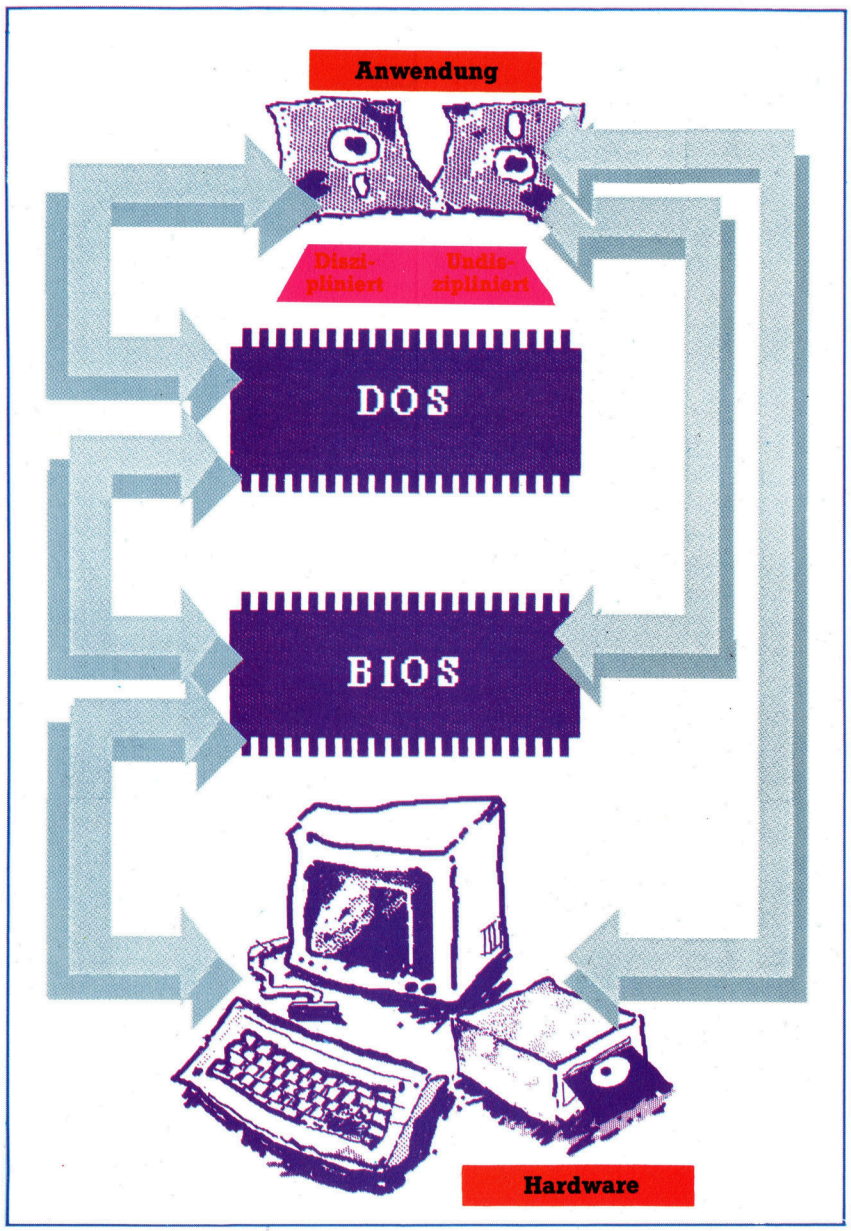
Die Vormachtstellung des IBM PC und seine MS-DOS Variante (mit Namen PC-DOS) sicherte MS-DOS den ersten Platz unter den 16-Bit-Betriebssystemen. Das System hatte seinen Anfang in einer recht groben CP/M-Anpassung eines früheren Angestellten von Digital Research.

Diszipliniert
Software, die mit standardmäßigen MS-DOS Aufrufen arbeitet, wird „diszipliniert" genannt. Das DOS spricht die BIOS-Routinen dabei indirekt über eine Sprungtabelle an. Unser Bild zeigt, wie diese Art der Programmierung aufgebaut ist. Programme, die mit absoluten BIOS-Adressen arbeiten oder MS-DOS völlig umgehen und die Hardware direkt ansprechen, heißen „undiszipliniert". Auf dem IBM PC liegt das BIOS in einem urheberrechtlich geschützten ROM. Die Sprungtabelle fängt bei der Adresse 400H an und liegt damit 512 Bytes (zwei Hexadezimalseiten) unter dem Anfang.

DOS Version, so daß jede Aussage dieser Serie über MS-DOS auch auf PC-DOS zutrifft. Unterschiede zwischen beiden Systemen fallen für Anwender kaum ins Gewicht. Intern gibt es einige Abweichungen, und auch die Numerierung der Disketten variiert etwas. Theoretisch wären dies alle Unterschiede, doch leider adressieren viele Anwendungsprogramme des IBM PC die Hardware direkt und nicht über die entsprechenden Systemroutinen. So findet sich hier die Hauptursache für Kompatibilitätsprobleme (und nicht im Unterschied zwischen PC-DOS und MS-DOS). Programme, die das DOS korrekt ansprechen, sollten ohne Anpassung unter jedem MS-DOS oder PC-DOS der gleichen (oder einer späteren) Version laufen. Selbstverständlich funktionieren DOS-Verbesserungen, die erst kürzlich (z. B. in Version 3.1) herausgegeben wurden, nicht mit früheren Versionen ohne diese Fähigkeiten.

Die erste MS-DOS-Version war eigentlich ein an die 16-Bit-Familie von Intel angepaßtes CP/M-80, das in vielen Befehlen und Abläufen noch CP/M ähnelte. Tim Patterson hatte jedoch einige Befehle bedienerfreundlicher gestaltet.

Für Verbesserungen des Systems orientierte sich Microsoft zunächst an Xenix, ihrer eigenen Version von Unix (das Mehrplatz-Betriebssystem von Bell Labs) und versuchte, über gemeinsame Systemaufrufe einen gewissen Grad an Kompatibilität herzustellen. Für den Anwender hatten jedoch in DOS 2 die E/A-Richtungsumkehrung, die „Pipelines" und das hierarchische Inhaltsverzeichnis weit größere Bedeutung. All diese Konzepte gingen auf das wohlbekannte Unix zurück.



Mehrplatzanwendungen

DOS 3 – die neueste MS-DOS-Version – enthält alle Eigenschaften des DOS 2, unterstützt darüberhinaus noch Mehrplatzanwendungen. OEMs für Netzwerksysteme (z. B. Apricot oder Research Machines Limited) arbeiten mit Version 3.1 – einer von beiden Hauptversionen dieser Generation – und dem darin eingebauten Microsoft Netzwerk.

Da jedoch das Betriebssystem dadurch sehr umfangreich wird, steht für andere Umgebungen das von allen Netzwerkfähigkeiten des 3.1 befreite DOS 3.0 zur Verfügung, das aber auch den Mehrfachzugriff auf Dateien unterstützt. Diese Softwarelösung läßt sich gut für Mehrplatzanwendungen einsetzen, die mit gleichzeitigem Zugriff auf gemeinsame Datenbanken arbeiten.

Der Intel 80286 des IBM PC-AT hat einen Adreßraum von drei MBytes und Hardwareschaltungen zur Sperrung von Dateibereichen. Auf dem 8086 und 80186 ist es unter DOS 3.0 zwar möglich, den Zugang zu einem Bereich mit zusätzlichen Systemaufrufen zu sperren, doch geht dies zu Lasten der Verarbeitungsgeschwindigkeit. Durch weitere Verbesserungen sind inzwischen fast alle Fehler des alten „QDOS" ausgemerzt, und MS-DOS stellt ein ausgereiftes Produkt dar.

Wichtige Verbesserungen

Eine der wichtigsten Verbesserungen ist Microsoft Windows. Wie GEM von DR bietet MS-Windows eine WIMP-ähnliche Umgebung mit Multitasking und einfachem Datenaustausch zwischen den Anwendungen. Das System wird als Erweiterung von MS-DOS/PC-DOS eingesetzt und soll auf den herkömmlichen Computern eine neue Ära der Bedienerfreundlichkeit einleiten. In der nächsten Folge untersuchen wir die Grundbefehle von MS-DOS. Vermutlich am häufigsten wird wohl dabei auf allen Systemen der Befehl „Directory" eingesetzt. Er zeigt nämlich alle Dateinamen einer Diskette, oder – von DOS 2 an aufwärts – ein „Directory", einem Teil der Diskette.

Die Klassiker

Klassische Sprachen wie FORTRAN und COBOL haben noch immer viel Einfluß auf die modernen Programmtheorien. Wir beschäftigen uns mit ihrer Entstehung und den ersten wichtigen Konzepten.

Die meisten Menschen empfinden die Programmierung eines Computers als einen ganz normalen Vorgang. Selbst wenn sie damit nur wenig Erfahrung haben, wissen sie zumindest, worum es geht. Um 1950, als das Konzept der Programmierung noch kaum entwickelt war, sah es jedoch völlig anders aus. Dennoch stammt aus dieser frühen Periode eine Reihe von Programmiersprachen, die noch heute im Einsatz sind.

Wir untersuchen, wie sich diese Sprachen zu ihrer jetzigen Form entwickelten, verfolgen ihren Einfluß auf modernere Sprachsysteme und sehen uns die Wirkung auf die Microcomputer der heutigen Zeit an. Doch zunächst: Wie konnte in den frühen Tagen des Computers die Idee einer „Computersprache" überhaupt entstehen?

Programme aus Zahlen

Wenn wir die ersten Entwicklungsstadien beiseitelassen, in denen eine „Programmierung" durch Umlöten der Kontakte geschah, dann bestanden die ersten echten Programme nur aus Zahlen. Geschrieben wurden die Zahlenfolgen üblicherweise in Oktal (Basis Acht). Die Nummern mußten mühsam über Schalter an der Vorderseite des Gerätes und später über Lochkarten oder Lochstreifen eingegeben werden. Jedes neue Programm wurde auf niedrigster Ebene von Grund auf neu entwickelt, und oft mußten die Programmierer die gleichen Routinen wieder und wieder schreiben. Nach und nach entstand das Konzept einer Bibliothek für Subroutinen – den Begriff „Subroutine" gab es damals allerdings noch nicht. Diese „Bibliotheken" fanden sich damals in den Notizbüchern der Programmierer und wurden je nach Bedarf in neue Programme abgeschrieben. Die Notizbücher waren sorgsam gehütete Geheimnisse, zu denen andere Programmierer nur selten Zugang erhielten.

In Manchester geschah ein großer Schritt nach vorn, als 1951 das EDSAC-System gebaut wurde und Wheeler, Wilkes und Gill einen aufeinander abgestimmten Satz von Subroutinen entwickelten, der zum Lieferumfang der Maschine gehörte. Da nun alle mit den gleichen Subroutinen arbeiteten, war auch die Programmentwicklung einfacher, und es zeigten sich die ersten strukturellen Ähnlichkeiten mit modernen Abläufen. Die Codeblocks führten bestimmte Aufgaben durch den Aufruf von Subroutinen aus, die Standardfunktionen wie Ein-/Ausgabe und Berechnungen steuerten.

Mit wachsenden Speicherkapazitäten ließen sich die Subroutinen auch intern – oder zumindest im direkten Zugriff – speichern. Von hier war es nur noch ein kleiner Schritt zur Entwicklung eines Codes, in dem der Programmierer die gewünschten Subroutinen mit alphabetischen Zeichen oder mathematischen Formeln angeben konnte. Das Programm „sah" sich diesen Code Zeile für Zeile an, rief die erforderlichen Subroutinen auf und bearbeitete dann die nächste Zeile. Mit dieser Technik arbeiten auch heute noch die modernen BASIC-Interpreter. Eines der ersten Systeme dieser Art war der „Short Code", den John Maunchly 1949 auf dem BINAC-Computer und später dem UNIVAC entwickelte.

In einer weiteren Verbesserung dieser frühen Technik wurden die Subroutinen vor ihrer Ausführung erst zu einem Ausgabegerät geschickt und so das Programm gespeichert. In diesem Stadium entstanden dann auch die ersten Compiler („compilieren" – zusammentra-

Meilensteine

Eine einfache Chronologie der frühen Programmiersprachen:

1951 Das erste Programm wird digital in dem EDSAC-Computer (Manchester) gespeichert

1952 Der „Short Code" von John Maunchly

1953 „Speedcoding" von Laning und Zierly

1954 Der A-2 Compiler
Die ersten FORTRAN-Beschreibungen

1955 Die ersten echten Compiler für FORTRAN und FLOW-MATIC

1956 Allgemeine Veröffentlichung der ersten FORTRAN- und FLOW-MATIC-Compiler
Beschreibung für FORTRAN II

1957 Definition und Veröffentlichung des ersten ALGOL-Compilers

1958 Veröffentlichung von FORTRAN II

1959 Erste Beschreibung von COBOL
Erster LISP-Compiler

1960 Veröffentlichung des ersten COBOL-Compilers und des überarbeiteten ALGOL 60

Einfache Reservierungssysteme

FORTRAN IV:

```
C       THEATER-PLATZ-RESERVIERUNG
C       DAS PROGRAMM NIMMT EINE SITZ-
C       NUMMER ENTGEGEN, PRUEFT, OB ER
C       BEREITS GEBUCHT IST
C       BUCHT IHN, WENN ER FREI IST, ODER
C       GIBT EINE MELDUNG AUS.
C
C       VARIABLEN DEKLARIEREN
C
        INTEGER SEATNO, SEAT (500)
C
C       ALLE FREIEN SITZE MARKIEREN
C
        DO 100 I = 1,500
100     SEAT (I) = 0
C
C       SITZNUMMER LESEN UND VERFUEG-
C       BARKEIT PRUEFEN
C
101     READ (1,10) SEATNO
        IF (SEAT (SEATNO). NE.0) GOTO 102
C
C       SITZ IST FREI
C
        SEAT (SEATNO) = 1
        WRITE (1,20)
        GOTO 103
C
C       SITZ IST NICHT FREI
C
102     WRITE (1,30)
C
C       NAECHSTE SITZNUMMER
C
103     GOTO 100
C
C       AUSGABE FORMATIEREN
C
10      FORMAT (I4)
20      FORMAT (1H, 13HSITZ IST FREI)
30      FORMAT (1H, 24HSITZ IST BEREITS GE-
        BUCHT)
        END
```

ALGOL 60:

```
comment, THEATER-PLATZ-RESERVIERUNG
DAS PROGRAMM NIMMT EINE SITZNUMMER
ENTGEGEN, PRÜFT, OB ER BEREITS GEBUCHT
IST
BUCHT IHN, WENN ER FREI IST, ODER GIBT EINE
MELDUNG AUS.
BEACHTEN SIE, DASS ALGOL KEINE EIN- ODER
AUSGABEBEFEHLE ENTHÄLT:
begin:
  integer SEATNO, I;
  integer array SEAT[1:500];
  comment ALLE FREIEN SITZE MARKIEREN
  for I:= 1 step 1 until 500 do
    SEAT [1]:=0
  comment SITZNUMMER LESEN UND VER-
  FÜGBARKEIT PRÜFEN;
NEWSEAT: ((read SEATNO));
  if SEAT [SEATNO] = 0 then begin
    SEAT [SEATNO]: = 1;
    ((print 'SITZ IST FREI'));
  end
  else
    ((print 'SITZ IST BEREITS GEBUCHT'));
    goto NEWSEAT;
  end
```

COBOL 74
(Nur Daten- und Verarbeitungsteil):

```
* THEATER-PLATZ-RESERVATIONEN.
* DAS PROGRAMM NIMMT EINE SITZNUMMER
ENTGEGEN, PRÜFT, OB ER BEREITS GEBUCHT
IST,
*BUCHT IHN, WENN ER FREI IST, ODER GIBT
EINE MELDUNG AUS.

DATA DIVISION.
WORKING STORAGE SECTION.
01 SEAT-AVAILABILITY.
  02 SEAT PIC 9 OCCURS 500 TIMES.
77 SEAT-NUMBER PIC 999.
77 LOOP-COUNTER PIC 999.
77 SEAT-AVAILABLE PIC X (13) VALUE
  „SITZ
IST FREI".
77 SEAT-BOOKED PIC X (24) VALUE
  „SITZ IST BEREITS GEBUCHT".
PROCEDURE DIVISION.
MAIN PARAGRAPH.
* ALLE FREIEN SITZE MARKIEREN
  PERFORM MARK-SEAT-AVAILABLE-
    PARAGRAPH
    VARIYING LOOP-COUNTER FROM 1 BY 1
    UNTIL I › 500.
* SITZNUMMER LESEN UND VERFÜGBARKEIT
PRÜFEN
  PERFOM GET-SEAT-NUMBER PARAGRAPH.
  STOP RUN.
MARK-SEAT-AVAILABLE-PARAGRAPH.
   MOVE ZERO TO SEAT (LOOP-COUNTER).
GET-SEAT-NUMBER-PARAGRAPH.
  ACCEPT SEAT-NUMBER.
  IF SEAT (SEAT-NUMBER) IS EQUAL TO 0
    MOVE 1 TO SEAT (SEAT-NUMBER)
  DISPLAY SEAT-AVAILABLE
  ELSE
  DISPLAY SEAT-BOOKED.
  GOTO GET-SEAT-NUMBER-PARAGRAPH.
```

Die folgenden drei Listings in FORTRAN, COBOL und ALGOL zeigen die Programmstruktur und einige vergleichbare Eigenschaften. Jedes Programm nimmt eine Sitznummer entgegen, prüft, ob die Nummer zuvor schon eingegeben (gebucht) wurde. Falls nicht, wird der Sitz als gebucht markiert.

gen – bedeutete ursprünglich, daß Programme ähnlich wie Artikelsammlungen aus vielen Einzelteilen zusammengetragen wurden).

Einer der frühesten erfolgreichen Compiler war der A-2, den 1955 ein von Grace Hopper geleitetes Team bei Remington Rand entwickelte. Der A-2 arbeitete mit „Dreieradressen". Dabei erhielt jeder Vorgang einen mnemotischen Namen, gefolgt von drei Adressen – zwei für die Quellenangabe und eine für die Bestimmung. In einigen Bereichen ähnelte er einem Assembler, war jedoch auf keine bestimmte Maschine angepaßt. Später entstand daraus ARITH-MATIC, gefolgt von AT3 oder MATH-MATIC – eine ähnliche, vom Remington Rand entwickelte Sprache.

Ende 1951 wurde klar, daß auf den Computern Programme liefen, die vom Blickpunkt des Anwenders aus nicht in der Maschinensprache des Gerätes geschrieben waren. Die Tatsache, daß der Computer das Programm erst in seinen Maschinencode übersetzen mußte, war dabei nicht von Bedeutung.

In diesem Fall konnte man aber auch eine völlig neue Sprache entwickeln, die dem Programmierer die Programmentwicklung erleichterte, statt die Übersetzung noch weiter zu beschleunigen. Die neue Hardware wurde sowieso immer schneller und leistungsfähiger, so daß die Programmierer kaum mithielten.

Das nächste Problem trat auf, als sich zeigte, daß kein wie auch immer gearteter Standard existierte. Jede Maschine arbeitete mit ihrer eigenen Sprache, die auf ihren Aufgabenbereich ausgerichtet war. Ohne eine hardwareunabhängige Sprache, die sich für einen weiteren Aufgabenkreis eignete, konnte es nicht weitergehen. Die Anfänge von 1954 führten schließlich zu FORTRAN (das FORmula TRANslation System von IBM), der ersten echten Programmiersprache.

Sprache war gefordert

FORTRAN eignete sich ausgezeichnet für die Zahlenverarbeitung, aus denen die Computerarbeit damals hauptsächlich bestand. Als die Geschäftswelt sich jedoch dafür interessierte, ihre großen Datenmengen von Computern bearbeiten zu lassen, mußte eine Sprache her, die dem Geschäftsenglisch entsprach.

Interessanterweise gab es zu dieser Zeit eine ganze Reihe von Programmierern, die sich nicht vorstellen konnten, daß Computer je Wörter statt Zahlen „verstehen" könnten. Diese Meinung erwies sich jedoch schon bald als falsch, als Grace Hopper (und andere) eine Sprache namens FLOW-MATIC entwickelten, die von Management und Programmierern gleichermaßen verstanden wurde. 1956 wurde daraus COBOL (COmmon Business Oriented Language). Das Sprachsystem war exakt definiert.

Zu diesem Zeitpunkt war die Zahl der Computer und Programmierer bereits stark gewachsen. Einige Forscher beschäftigten sich mit dem theoretischen Aspekt der Programmiersprachen und suchten bessere und elegantere Wege für die Umsetzung von Algorithmen. Dies führte 1958 schließlich zu der Entwicklung von ALGOL (ALGOrithmic Language). ALGOL fand nie die Verbreitung von FORTRAN oder COBOL, nimmt in der Sprachentwicklung aber einen bedeutenden Platz ein. ALGOL förderte als erste den systematischen Programmaufbau, der zu einem der wichtigsten Bestandteile aller neueren Sprachen wurde.

In den frühen Tagen entstand eine Reihe weiterer Sprachen, die teilweise heute noch in Gebrauch sind. Eins der besten Beispiele ist die zwischen 1956 und 1958 entwickelte LISP (LISt Processing Language). Die Sprache gewinnt inzwischen im Feld der Künstlichen Intelligenz eine ständig wachsende Bedeutung. Die Hauptsprachen der letzten zwanzig Jahren bleiben jedoch FORTRAN, COBOL und ALGOL. Sie wurden im Laufe der Zeit mehrmals überarbeitet und erhielten neue Fähigkeiten, die den modernen Trends der Programmentwicklung Rechnung tragen. Die neuesten Versionen sind FORTRAN 77 (die Sprachdefinition von 1977), COBOL 74 und ALGOL 68.

1964 erschien am Dartmouth College (USA) eine vereinfachte (und von ALGOL beeinflußte) FORTRAN-Version, die das Lernen der Sprache erleichtern sollte. Die Version funktionierte außerdem auf den neuen Mehrplatzanlagen, die im Timesharing-Verfahren arbeiteten. Diese Version wurde unter dem Namen BASIC (Beginners All-purpose Symbolic Instruction Code) allgemein bekannt.

Als 1968 der neue ALGOL-Standard vorbereitet wurde, war Niklaus Wirth mit der wachsenden Komplexität der Sprache nicht einverstanden und versuchte, ein einfacheres und eleganteres Konzept durchzusetzen. Inzwischen wird das sehr komplexe ALGOL 68 nur noch in akademischen Bereichen eingesetzt, während die weit einfachere Version von Wirth – PASCAL – weite Verbreitung fand.

Short Code

In dem Short Code von John Maunchly sieht die vertraute BASIC-Anweisung

10 A = B + C

so aus:

10 S0 03 S1 07 S2

Dabei ist 10 die Zeilennummer, und S0, S1 und S2 stellen die Symbole für A, B und C dar. 03 und 07 sind die Verarbeitungscodes für Zuordnung und Addition.
In A-2 lautet der Befehl:

ADD B C A

Bits im Wandel

Nach Erläuterung der Grundlagen einer D/A- oder A/D-Wandlung und dem Entwurf unseres Digitalmultimeters untersuchen wir jetzt die am weitesten verbreiteten Wandlertechniken. Den integrierten Wandler 7135 wollen wir etwas detaillierter beschreiben – er soll in unserer Konstruktion verwendet werden.

Während die Zeit für die Phasen I und II festgelegt ist, ist die Dauer der dritten Phase proportional zur Eingangsspannung V in. Während dieser Phase wird die vorher integrierte Eingangsspannung gleichmäßig auf Null reduziert: Das dauert um so länger, je größer die Spannung anfangs war. Man kann deshalb unser Gerät sehr einfach mit einem Computer koppeln. Der BUSY-Ausgang des 7135 geht zu Beginn der Signal-Integration auf „High" und bleibt dort bis einen Puls nach dem Nulldurchgang des Integrators. Ein logisches AND zwischen den Taktpulsen und dem BUSY-Signal ergibt die Anzahl gültiger Pulse. Diese Pulse können über ein serielles Interface an den Rechner weitergegeben werden.
Zieht man hiervon die 10 001 Pulse der Signal-Integration und den einen am Ende der Referenz-Integration ab, ist die verbleibende Zahl der Eingangsspannung direkt proportional.

Die meisten D/A-Wandler arbeiten entweder nach dem Wägeverfahren oder mit integrierender Wandlung. Das gilt sowohl für auf ICs zusammengefaßte Schaltungen als auch für einen „diskreten" Aufbau, also ein Gerät aus Einzelbauteilen. Das IC 7135 ist ein integrierter Wandler. Bevor wir seine Funktion genauer untersuchen, wollen wir kurz die Grundlagen beider Methoden beschreiben.

Beim Wägeverfahren benutzt man einen D/A-Wandler in einem Rückkopplungskreis zusammen mit einem Komparator (Vergleicher). Die Ausgangsspannung des D/A-Wandlers wird Bit für Bit verändert, wobei man mit dem höchstwertigen Bit (MSB) anfängt und sich zum niederwertigen (LSB) vorarbeitet. Zu Beginn werden alle Bits auf Eins gesetzt. Solange die Eingangsspannung noch größer als die Ausgangsspannung des D/A-Wandlers ist, bleibt das untersuchte Bit auf Eins, andernfalls wird es zurückgesetzt. Dann geht es mit der nächsten Stelle weiter. Sind alle Bits getestet (meist 8, 12 oder 16), erzeugen die auf 1 stehenden (gesetzten) Bits am D/A-Wandler eine Spannung, die der am Analogeingang anliegenden exakt gleicht.

Dazu braucht eine Acht-Bit-Wandlung acht Vergleichsoperationen, eine 16-Bit-Wandlung 16 usw. Das Wägeverfahren arbeitet sehr schnell, 100 000 Wandlungen in der Sekunde oder mehr sind keine Seltenheit. Damit ist es ideal für Anwendungen in der Audio- Video- oder Radartechnik, wo eine Vielzahl analoger Daten in kürzester Zeit auf Digitalform gebracht werden müssen. Nachteilig ist, daß Wägekonverter sehr teuer sind und eine überaus komplizierte Beschaltung benötigen.

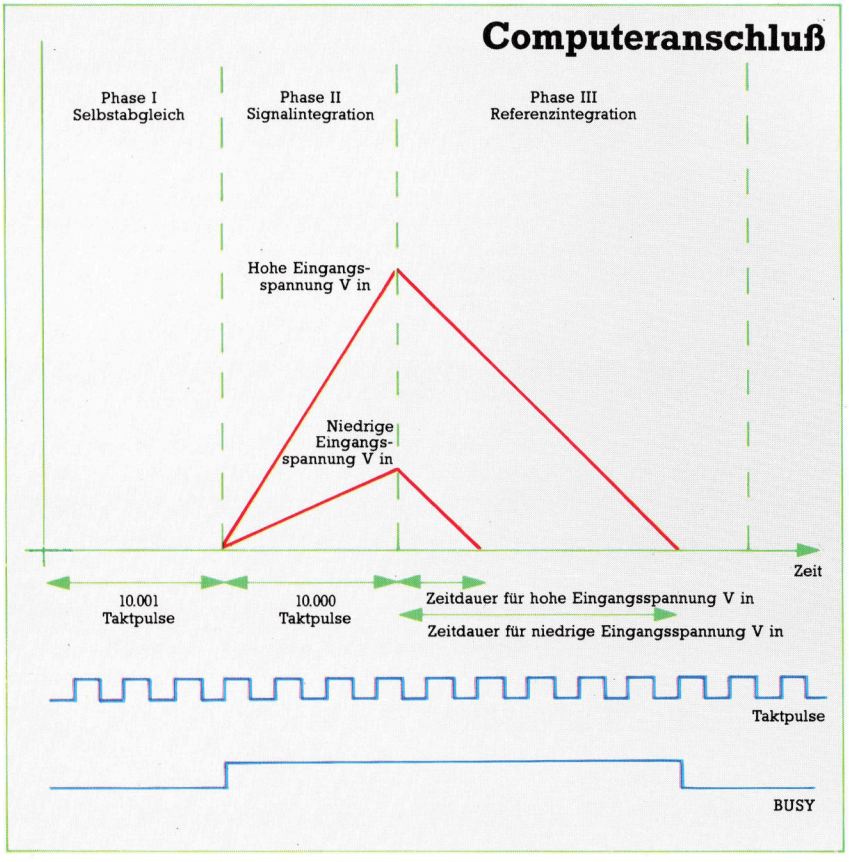
Sample and hold

Wenn, wie etwa bei Digitalmultimetern, nicht die Geschwindigkeit im Vordergrund steht, entscheidet man sich üblicherweise für einen integrierten Wandler. Damit können Wandlungsraten von einigen Messungen pro Sekunde erreicht werden. Das reicht fast immer – vorausgesetzt, man möchte nicht gerade hochfrequente Spannungsverläufe untersuchen.

Der Ausgang eines integrierten Wandlers gibt den Mittelwert der analogen Eingangsspannung innerhalb einer fest gewählten Zeit an. Anders als beim Wägeverfahren muß die analoge Eingangsspannung nicht zwischengespeichert werden (Sample and hold). Der zeitliche Ablauf der Wandlung wird durch die Pulse eines Taktgenerators gesteuert.

Die Vorzüge der in unseren Multimeter eingesetzten integrierten Wandlung liegt in der hohen Genauigkeit, dem sehr guten Rauschverhalten und einem Aufbau, der – abgesehen von der Schaltung zur Erzeugung einer Referenzspannung – ohne „kritische" Bauteile auskommt. Außerdem fällt die „Sample and hold"-Komponente der oben beschriebenen Wandlungsmethode fort, was sich auch durch geringere Kosten angenehm bemerkbar macht.

Bei einem typischen Wandler dieses Typs wird die Konversion in drei Phasen vorgenommen: Auf den Selbstabgleich folgt die Signal-Integration und Referenz-Integration. Durch diese Technik wird eine relativ große Unempfindlichkeit gegenüber hochfrequenten Störun-



gen und Eingangsrauschen erreicht.

Neben einem stabilen Taktgenerator braucht man dafür nur eine genaue Referenzspannung, die aber mit handelsüblichen Zener-Dioden recht einfach erzeugt werden kann. Diese Dioden begrenzen die höchstzulässige Eingangsspannung.

Die meisten integrierten Wandler arbeiten nach dem sogenannten „Dual-Slope"-Verfahren, dessen Ablauf so aussieht:

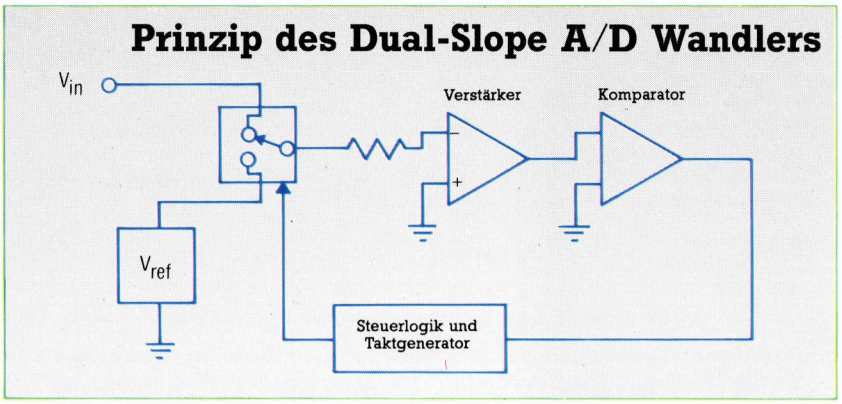
Phase 1/Selbstabgleich: Der Eingang wird auf Masse gelegt, die Fehlerinformation über die analogen Eingangskomponenten wird in einem Kondensator gespeichert.

Phase 2/Signal-Integration: Das Eingangssignal wird über eine bestimmte Anzahl von Taktpulsen integriert – für eine Wandlung mit 4 1/2 Stellen sind 10 000 üblich. Nach der Integration ist die erzeugte Spannung dem Eingangssignal direkt proportional.

Phase 3/Referenz-Integration: Zu Beginn dieser Phase wird der Eingang des Integrators an die Referenzspannung gelegt. Die Zahl der Pulse, die zwischen dem Anlegen der Referenzspannung und dem Nulldurchgang des Integrators gezählt wird, ist dann der Eingangsspannung direkt proportional.

Dual-Slope A/D-Wandler sind sehr exakt, weil das Meßergebnis nur von der Genauigkeit der Referenzspannung und der Stabilität des Taktgebers abhängt. Die verwendete Zenerdiode sollte deshalb sorgfältig ausgewählt werden, die Frequenz des Taktgenerators ist dagegen relativ beliebig, auch das Tastverhältnis ist nicht wichtig. Wir haben uns daher für das Timer IC 555 entschieden, dessen Frequenzstabilität für diese Anwendung ausreicht.

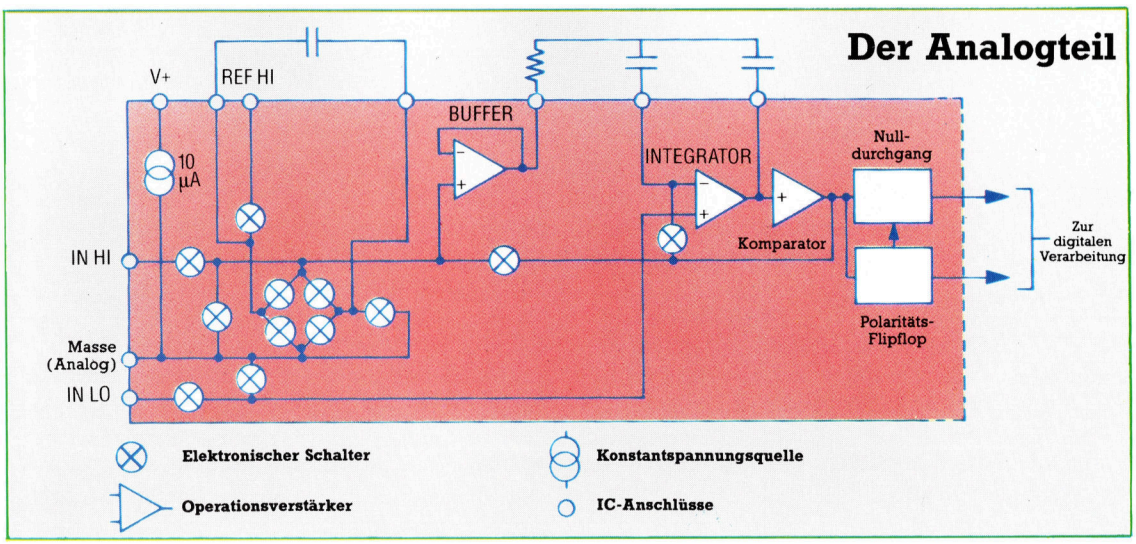
Die Stabilität anderer Komponenten, etwa des Integrationskondensators, ist ohne Bedeutung, solange sich sein Wert nicht innerhalb einer Wandlung ändert. Beim Wägeverfahren hängt die Exaktheit des Ergebnisses dagegen sehr stark von der Genauigkeit des Widerstands-Netzwerks ab. Da die Genauigkeit der Uhr leicht auf Eins zu eine Million gebracht werden kann, sind Dual-Slope-Wandler dem Wägeverfahren deutlich überlegen – jedenfalls, solange auf hohe Wandlungsgeschwindigkeit kein Wert gelegt wird.



Der Schaltplan zeigt den analogen Teil des 7135-ICs. Die meisten Symbole kennen Sie bereits, der Doppelkreis und die durchkreuzten Kreise sind Ihnen aber vielleicht noch neu. Der Doppelkreis steht für eine Konstantspannungsquelle, die durchkreuzten Kreise stellen elektronische Schalter dar.
Ein integrierender Dual-Slope A/D-Wandler arbeitet in drei Phasen: Selbstabgleich, Signal-Integration und Referenz-Integration. Während der Signal-Integration wird das Eingangssignal -Vin- und während der Referenz-Integration die Referenzspannung -Vref- eingeschaltet. Der Ablauf wird durch einen externen Taktgeber gesteuert.

Es ist wichtig, das Zeitverhalten in den drei "hasen der Wandlung – Selbstabgleich, Signal-Integration und Referenz-Integration – etwas genauer zu betrachten. Die Anzahl der Taktpulse in den ersten beiden Phasen ist festgelegt, nämlich 10001 Pulse für den Selbstabgleich und 10000 Pulse für die Signal-Integration. Die Anzahl der Pulse in der dritten Phase dagegen ist variabel. Es werden bis zu einer Obergrenze von 20001 so viele Pulse gezählt, bis der Ausgang des Integrators einen Nulldurchgang macht. Daher können die Ergebnisse mit relativ unkomplizierter Technik über ein serielles Interface an einen Computer weitergegeben werden.

Die Frequenz des Taktgenerators spielt innerhalb weiter Grenzen keine Rolle. Sie kann bei 5 kHz liegen (die Wandlung dauert dann etwa zehn Sekunden), oder bei 1 MHz. Bei zu niedrigen Frequenzen treten allerdings Fehler durch die Leckströme des Integrationskondensators auf, bei sehr hohen Frequenzen muß der Kondensator dagegen durch ein Widerstandsnetzwerk kompensiert werden. Am wenigsten Probleme gibt es bei Frequenzen zwischen 100 und 160 kHz.



Das Diagramm zeigt den analogen Flußplan des 7135 A/D-Wandlers. Bauteile im schattierten Gebiet befinden sich auf dem Chip, die anderen Komponenten müssen extern dazugeschaltet werden. Der Analogteil des 7135 versorgt den digitalen Teil mit Informationen über den Nulldurchgang und die Polarität. Zusammen mit den Signalen des externen Taktgebers kann der Digitalteil des ICs daraus die gewünschten Daten berechnen und über den BCD-Anschluß des Chips ausgeben.

Gesteuerte Treiber

In der letzten Folge haben wir uns mit den „Hakencodes" des Interface 1 beschäftigt. Diesmal sehen wir uns die Routinen an, mit denen die Microdrives des Spectrum gesteuert werden. Viele dieser Codes lassen sich in der Maschinenprogrammierung einsetzen.

Die Microdrives werden entweder indirekt über Hakencodes oder direkt mit den ROM-Routinen des Interface 1 angesprochen. Wir gehen hier jedoch nur auf die Codes ein, die in Verbindung mit dem Microdrive eingesetzt werden.

Zuvor ein kurzer Blick auf die Funktionsweise der Microdrives. Eine formatierte Cartridge enthält 255 Sektoren, in denen die Datenblöcke physisch gespeichert werden. Jeder Sektor besteht aus einem Kopf (mit dem Cartridgenamen und anderen Informationen), einem Datenblock (für den Namen der Datei, die später hier gespeichert wird), weiteren Daten und 512 Datenbytes. Der Formatiervorgang markiert außerdem alle Cartridgesektoren, die nicht beschreibbar sind.

Die Datenübertragung zwischen Microdrives und Spectrum läuft über einen Kanal, dem 595 Bytes zur Verfügung stehen. Ihm ist eine „Microdrive-Map" mit 32 Bytes zugeordnet, die den Status der Sektoren anzeigt. Die Microdrive-Map gibt an, welche Sektoren bereits belegt oder nicht einsetzbar sind.

Beim Anlegen eines neuen Kanals wird der Speicher zwischen dem Ende des Kanalbereichs für Microdrives und STKEND nach oben verschoben, um Platz zu schaffen. Nach dem Schließen eines Kanals wird der Speicher wieder entsprechend nach unten verlegt.

Microdrivebefehle

Das System hat zwei Möglichkeiten, Microdrivekanäle anzulegen: direkt – beim Eröffnen einer Datei mit dem Befehl OPEN und indirekt – bei einem Systemablauf, der einen Kanal automatisch öffnet. So eröffnet der Microdrive-Befehl SAVE auf indirekte Weise einen Kanal, der später möglicherweise nicht direkt eingesetzt wird. Direkt eröffnete Kanäle müssen duch einen weiteren Befehl wieder geschlossen werden, indirekt eröffnete Kanäle erledigen dies automatisch.

Das nebenstehende Bild zeigt die Struktur eines Microdrivekanals. Die Anfangsbytes dieses Kanals stimmen mit den Anfangsbytes der Kanäle überein, die Daten an den Bildschirm senden, oder von der Tastatur empfangen. Die meisten dieser Bereiche werden jedoch nur selten direkt angesprochen.

Da im Kanal auch die Adresse der Microdrive-Map gespeichert ist, läßt sich leicht feststellen, wieviel Platz auf der Cartridge noch zur Verfügung steht. Auch der BASIC-Befehl CAT arbeitet mit der Microdrive-Map, die Sie sich mit folgender Routine anzeigen lassen können (geben Sie zuerst NEW ein):

```
 10 OPEN #5;"M";1;"testprog"
 20 start=PEEK(23870)+256*PEEK(23871)
 30 FOR I=start TO start+31
 40 LET sector=PEEK(I)
 50 FOR J=1 TO 8
 60 IF sector/2=INT(sector/2) THEN PRINT "0";
 70 IF sector/2<>INT(sector/2) THEN PRINT "1";
 80 LET sector=sector/2:LET sector=INT(sector)
 90 NEXT J
100 PRINT
110 NEXT I
120 CLOSE #5
```

Beim Programmablauf zeigt eine 1 einen nicht einsatzbereiten oder bereits beschriebenen Sektor an und eine 0 einen freien. Über eine kleine Programmänderung kann man sich auch den gesamten auf der Cartridge zur Verfügung stehenden Platz anzeigen lassen.

Das OS füllt die 512 Bytes des Datenbuffers mit Informationen, die auf Band geschrieben werden sollen. Bei einem gefüllten Buffer oder bei Ausgabe des Befehls CLOSE (oder des entsprechenden Hakencodes) läuft automatisch der Schreibvorgang ab. Im zweiten Fall erhält der Datensatz dabei die Markierung EOF (End Of File – Dateiende). EOF kann aber auch über einen Hakencode geschrieben werden – wir kommen später darauf zurück. Normalerweise wird die Cartridge in Abschnitten von 512 Bytes beschrieben.

Sehen wir uns nun die Hakencodes genauer an, die die Microdrives steuern.

- Hakencode 33 – startet den Motor des angegebenen Microdrives – die Drives haben die Nummern Eins bis Acht. Die Routine kann weiterhin die Motoren aller Microdrives gleichzeitig abstellen. Da nach dem Ablauf möglicherweise die Interrupts abgeschaltet sind, sollten Sie sie in jedem Fall mit dem Befehl „EI" reaktivieren. Für den Einsatz der Routine setzen Sie die Nummer des Drives, den Sie anschalten wollen, in das Register A und rufen mit RST#&8 den Hakencode auf. Der Wert 0 im Register A schaltet alle Drives ab. Dieser Ablauf läßt sich

jedoch auch mit einem Fehlercode des BASIC auslösen. Hakencodes können Fehler auf zwei Weisen erzeugen: Sie versuchen einen Drive anzuschalten, der nicht angeschlossen ist, oder sie sprechen eine Drive an, der keine formatierte Cartridge enthält.

● **Hakencode 34** – eröffnet eine Cartridgedatei. Die Routine prüft zunächst, ob auf der Cartridge eine Datei mit dem angegebenen Namen bereits existiert. Gibt es den Namen schon, dann wird die Datei zum Lesen eröffnet, gibt es ihn nicht, wird sie zum Schreiben vorbereitet. Das Schreib/Lese-Flag des angesprochenen Kanals zeigt mit Bit 0 auf 0 das Lesen und mit Bit 0 auf 1 das Schreiben einer Datei an.

Eine Datei kann mehrmals zum Lesen eröffnet werden. Dabei wird auch dann ein neuer Kanal belegt, wenn die zuvor eröffneten Kanäle noch aktiv sind.

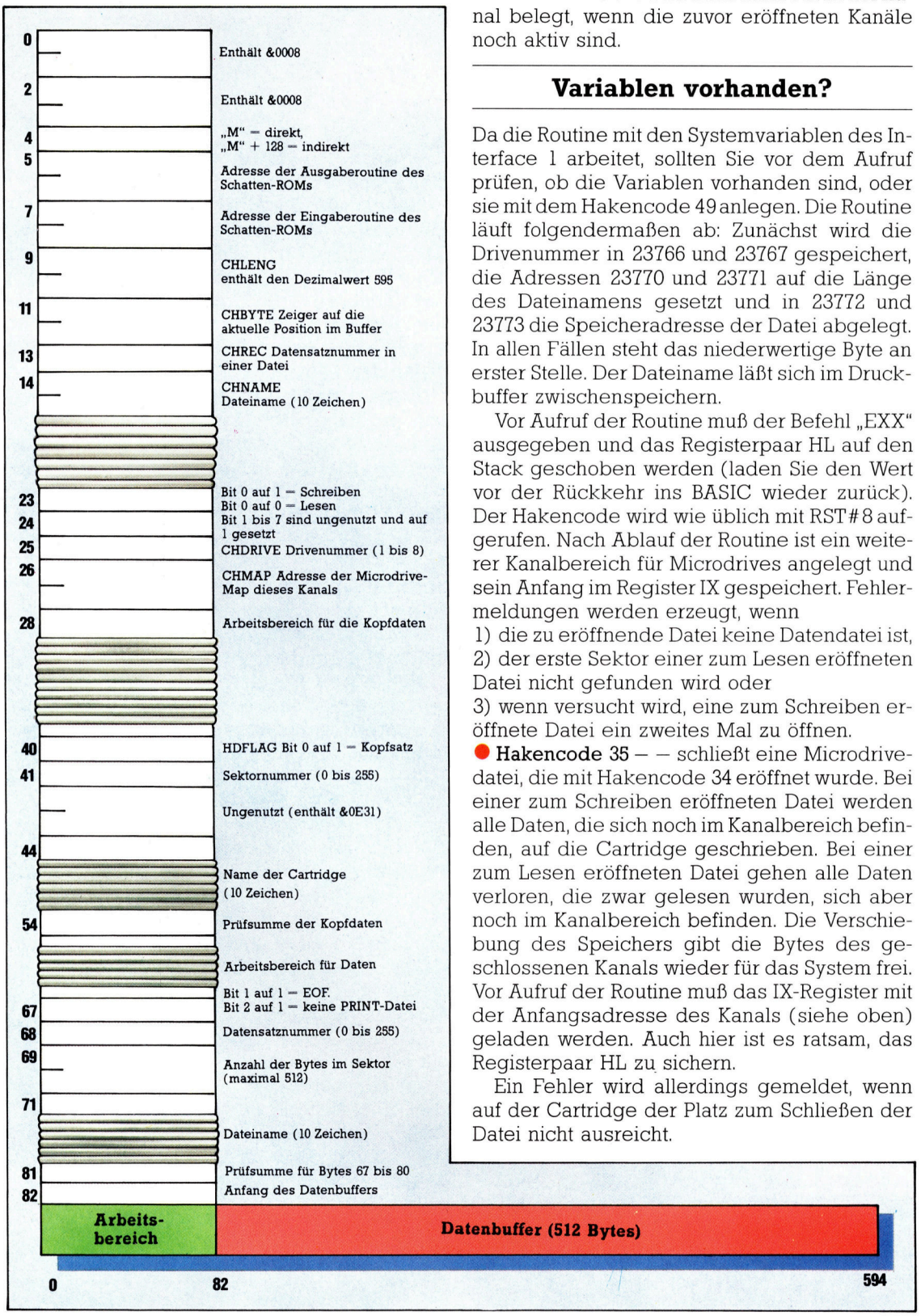
Variablen vorhanden?

Da die Routine mit den Systemvariablen des Interface 1 arbeitet, sollten Sie vor dem Aufruf prüfen, ob die Variablen vorhanden sind, oder sie mit dem Hakencode 49 anlegen. Die Routine läuft folgendermaßen ab: Zunächst wird die Drivenummer in 23766 und 23767 gespeichert, die Adressen 23770 und 23771 auf die Länge des Dateinamens gesetzt und in 23772 und 23773 die Speicheradresse der Datei abgelegt. In allen Fällen steht das niederwertige Byte an erster Stelle. Der Dateiname läßt sich im Druckbuffer zwischenspeichern.

Vor Aufruf der Routine muß der Befehl „EXX" ausgegeben und das Registerpaar HL auf den Stack geschoben werden (laden Sie den Wert vor der Rückkehr ins BASIC wieder zurück). Der Hakencode wird wie üblich mit RST#8 aufgerufen. Nach Ablauf der Routine ist ein weiterer Kanalbereich für Microdrives angelegt und sein Anfang im Register IX gespeichert. Fehlermeldungen werden erzeugt, wenn
1) die zu eröffnende Datei keine Datendatei ist,
2) der erste Sektor einer zum Lesen eröffneten Datei nicht gefunden wird oder
3) wenn versucht wird, eine zum Schreiben eröffnete Datei ein zweites Mal zu öffnen.

● **Hakencode 35** – – schließt eine Microdrivedatei, die mit Hakencode 34 eröffnet wurde. Bei einer zum Schreiben eröffneten Datei werden alle Daten, die sich noch im Kanalbereich befinden, auf die Cartridge geschrieben. Bei einer zum Lesen eröffneten Datei gehen alle Daten verloren, die zwar gelesen wurden, sich aber noch im Kanalbereich befinden. Die Verschiebung des Speichers gibt die Bytes des geschlossenen Kanals wieder für das System frei. Vor Aufruf der Routine muß das IX-Register mit der Anfangsadresse des Kanals (siehe oben) geladen werden. Auch hier ist es ratsam, das Registerpaar HL zu sichern.

Ein Fehler wird allerdings gemeldet, wenn auf der Cartridge der Platz zum Schließen der Datei nicht ausreicht.



Der Microdrivekanal belegt 595 Speicherbytes. Nach Eröffnung einer Cartridgedatei befindet sich die Anfangsadresse des Kanalbereichs im Register IX.

● **Hakencode 36** – – löscht eine Microdrivedatei – unabhängig davon, ob sie Daten oder Programme enthält. Dateiname, Adresse, Länge und Drivenummer werden wie beim Hakencode 34 angelegt.

Wie läßt sich eine eröffnete Datei nun beschreiben oder lesen? Microdrivekanäle lassen sich zwar jedem Strom zuordnen, doch ist es am einfachsten, den Microdrivekanal als „aktuellen" Kanal für das Lesen und Schreiben zu definieren.

Ist ein Microdrivekanal erst einmal als Ausgabekanal definiert, können Sie mit RST# 10 leicht Daten darauf schreiben. Dazu muß die Systemvariable CURCHL (bei 23633 und 23634) auf die Anfangsadresse des Microdrivekanals gesetzt werden. Hier der Befehl:

LD (23633), IX

(In IX ist die vom Hakencode 34 gelieferte Anfangsadresse des Kanals gespeichert.) RST# 10 schreibt nun seine Datenbytes in den Microdrivekanal statt sie zum Bildschirm zu senden. Hier zeigt sich, wie flexibel das Strom- und Kanalsystem des Spectrums ist – ein Aspekt, der nur selten berücksichtigt wird. Sobald der Buffer voll ist, wird er auf die Cartridge übertragen. Die folgende Routine schreibt die Datei FRED:

```
                ;write file "FRED" to microdrive
3E01            ld   a,1                ;drive number
32D65C          ld   (23766),a          ;store drive number
AF              xor  a                  ;zero a register
32D75C          ld   (23767),a
212B29          ld   hl,name            ;get name in hl
22DC5C          ld   (23772),hl         ;store filename
210400          ld   hl,4               ;4=name length
22DA5C          ld   (23770),hl         ;store name length
D9              exx
E5              push hl                 ;preserve hl'
D9              exx
CF              rst  #8
22              defb 34                 ;hook code 34
DD22515C        ld   (23633),ix         ;CURCHL=drive channel
06FF            ld   b,255              ;255 bytes infile
C5        loop: push bc                 ;preserve counter
3E00            ld   a,CHAR             ;user would need to
                                        ;insert routine to get
                                        ;character from wherever
                                        ;into a register
D7              rst  #10                ;write to channel
C1              pop  bc                 ;restore counter
10F6            djnz loop
CF              rst  #8
23              defb 35                 ;close file
3E02            ld   a,2
CD0116          call #1601              ;output to screen
D9              exx
E1              pop  hl                 ;restore HL
D9              exx
FB              ei                      ;reenable interrupts
C9              ret
46524544  name: defm "FRED"             ;filename
```

Wenn Sie mehrere Dateien eröffnen, müssen Sie selbstverständlich die Anfangsadressen jedes Kanals in einem sicheren Speicherbereich unterbringen, damit die Dateien auch geschlossen werden können etc. Unsere Routine erzeugt auf der Cartridge eine Datei von 255 Zeichen Umfang.

Auch das Lesen der Daten ist einfach: Die Datei wird wie beschrieben eröffnet und CURCHL auf die Anfangsadresse des Microdrivekanals gesetzt. Danach wird mit der Routine bei # 15E6 (im Haupt-ROM des Spectrum) ein Byte aus dem Kanal gelesen und anschließend im Register A gespeichert.

Hakencode 37 – Wenn IX beim Aufruf diese Codes die Anfangsadresse eines Kanals enthält, liest diese Routine den nächsten Datensatz einer Datei und legt die Daten im Datenbereich des Kanals ab. CHREC wird automatisch inkrementiert, wenn der Lesevorgang erfolgreich war. Die gelesenen Daten überschreiben die Daten des Kanalbuffers.

● **Hakencode 38** – schreibt alle Daten eines Kanals auf Band. IX muß zuvor mit der Anfangsadresse des gewünschten Kanals geladen werden. Beachten Sie, daß jeder Datenbereich, der weniger als 512 Bytex enthält, aber nicht mit einer EOF-Markierung als Dateiende gekennzeichnet ist, vom OS als freier Sektor angesehen wird. Sie sollten daher immer den EOF-Hakencode aufrufen, wenn der letzte Datenblock weniger als 512 Bytes enthält. Nach Aufruf von Hakencode 38 wird der aktuelle Inhalt des Datenbereichs in die Datei geschrieben.

● **Hakencode 39** – liest einen bestimmten Datensatz einer Datei. Dafür wird IX mit der Anfangsadresse des angesprochenen Kanals geladen und CHREC mit der Datensatznummer.

Alles der Reihe nach

Die weiteren Microdriveroutinen lassen sich kaum für die normale Programmierung einsetzen. So liest der Hakencode 40 die Daten eines bestimmten Sektors (in CHREC muß die gewünschte Sektornummer stehen), während Hakencode 41 den nächsten Sektor liest und Hakencode 42 Daten entsprechend in den nächsten freien Sektor schreibt.

Die letzten beiden Hakencodes beeinflussen die Microdrivekanäle. Hakencode 43 definiert einen Kanal und eine Cartridge-Map. In der ersten ROM-Version des Interface 1 war diese Routine jedoch fehlerhaft und löste den Ablauf von Hakencode 34 aus. Hakencode 44 löscht den Kanal, dessen Anfangsadresse in IX steht.

Zum Schluß noch eine Warnung: Setzen Sie die Hakencodes sehr vorsichtig ein! Der Schreibschutz der Microdrives ist softwaregesteuert – ein Programmfehler kann den Inhalt einer gesamten Cartridge zerstören.

Fachwörter von A bis Z

Output Device = Ausgabegerät
Jede Peripheriekomponente, die vom Rechner verarbeitete Informationen für den Benutzer zugänglich macht, wird als Ausgabegerät bezeichnet. Für Microcomputer sind alle möglichen Ausgabegeräte verfügbar; am gängigsten sind Bildschirmgeräte in Gestalt von Monitoren, Fernsehern oder Drucker. Der technische Fortschritt und das zunehmende Eindringen von Rechnern in immer neue Anwendungsfelder bringen es mit sich, daß das Angebot an Ausgabegeräten ständig vielfältiger wird. Die industrielle Fertigung z. B. bedient sich in wachsendem Ausmaß rechnergesteuerter Roboter, und für die nächste Zukunft sagen die Fachleute bei der Ausgabe-Peripherie vor allem einen Entwicklungsschub auf dem Gebiet der Sprachsynthese voraus.

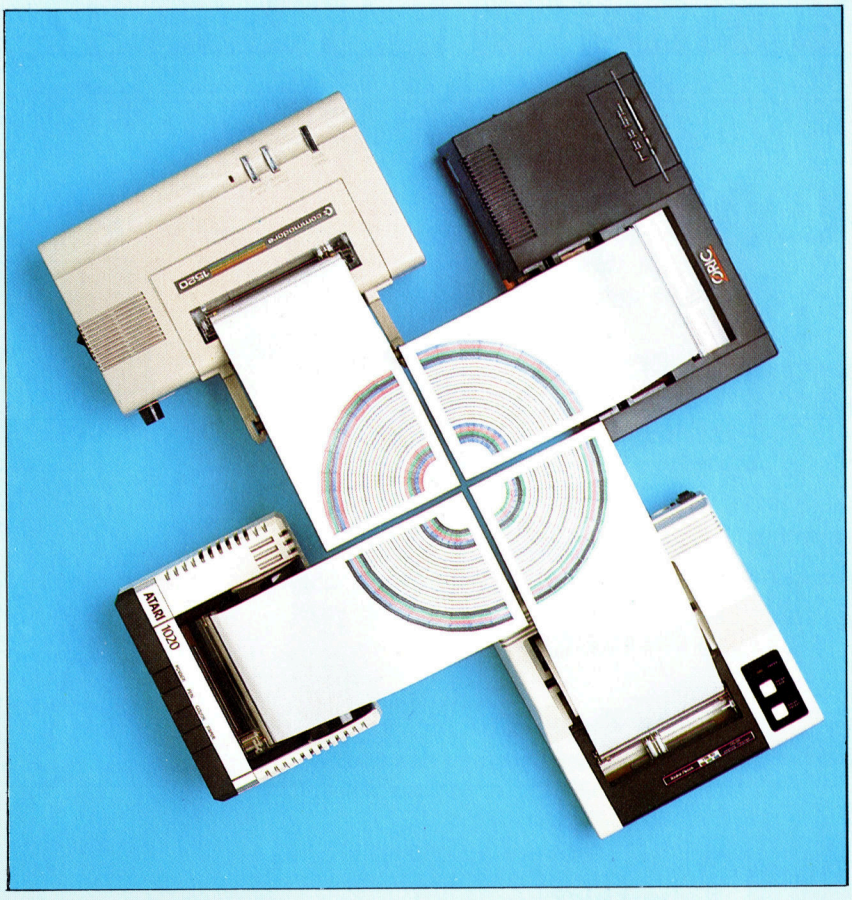
Overflow = Überlauf
Ein Überlauf tritt ein, wenn das Ergebnis einer Rechenoperation die vorgesehene Bitkapazität überschreitet; der Begriff wird dabei auch für den „übergelaufenen" Informationsteil selbst verwendet. Der Computer reklamiert den Überlauf entweder als Fehler, oder aber er beachtet ihn nicht – da beides zu Fehlergebnissen führt, sollte das Auftreten eines Überlauf von vornherein vermieden werden.

Packing = Packen
Beim Packen werden Daten aus Speicherplatzgründen so eng wie möglich zusammengeschoben. Dafür kann sich natürlich jeder Programmierer seine eigenen Tricks einfallen lassen, aber es gibt ein paar bewährte Verfahren: Beim Packen von Dezimalzahlen werden beispielsweise je zwei Ziffern in einem Byte untergebracht; Boolesche Variable, die ja nur ein Bit benötigen, passen zu acht in ein Byte, und beim „Tokenizing" werden häufig auftretende Informationseinheiten durch einen Kurzcode verschlüsselt. Generell kostet das Packen zusätzliche Entflechtungszeit bei der jeweiligen Programmausführung.

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.

Bildnachweise
2017, 2018: Science Photo Library
2019, 2020, 2035: Caroline Clayton
2022: Liz Heany
2028, 2040, 2041, 2043: Kevin Jones
2029, 2031, 2036, U3: Chris Stevens
2039: BBC Hulton Picture Library

Page Printer = Seitendrucker
Ein „Seitendrucker" gibt ähnlich wie eine Druckerpresse in einem Arbeitsgang gleich eine ganze Seite an Information aus, während ein Zeilendrucker (Line Printer) ganze Zeilen, ein serieller Drucker (z. B. mit Typenrad) dagegen nur einzelne Zeichen nacheinander zu Papier bringt. Seitendrucker arbeiten ohne mechanischen Anschlag und können mehrere DIN-A4-Seiten pro Sekunde liefern. Sie funktionieren fast durchweg elektrostatisch, wobei durch einen Laser auf einer Halbleitertrommel ein Ladungsbild erzeugt wird. Dies läßt sich durch Tonerpulver auf Papier übertragen und dort einbrennen. Die sog. „Laserdrucker" gibt es mittlerweile schon zum durchaus für viele User interessanten Preis eines Personal-Computers.

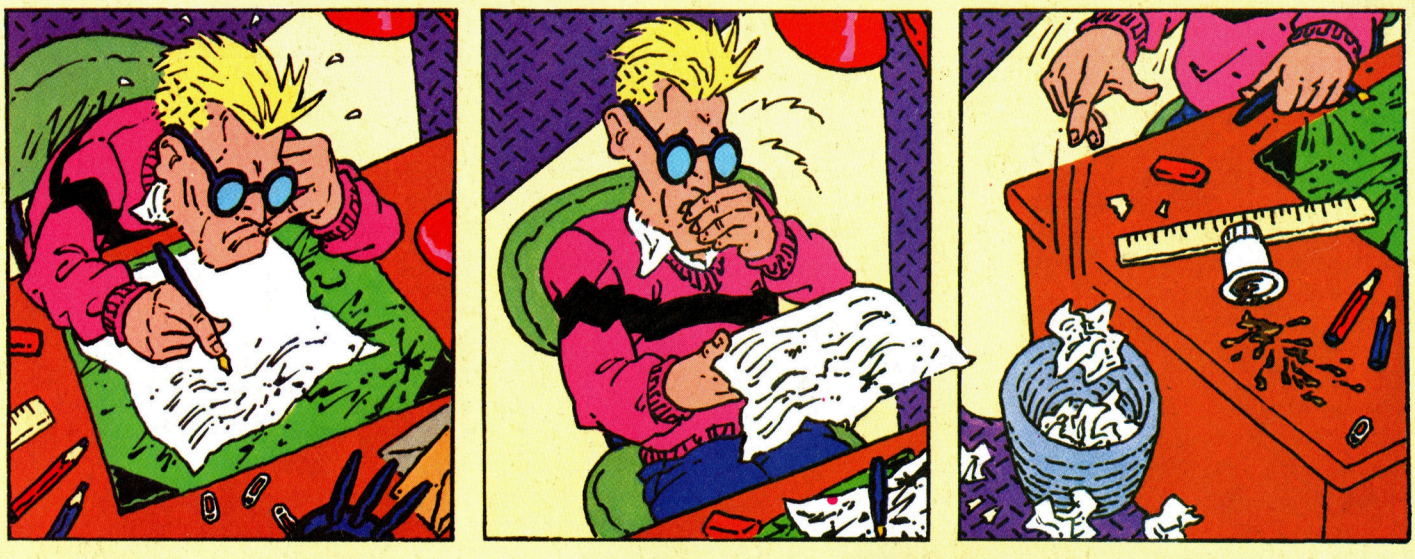


Besonders nützlich sind Ausgabegeräte wie diese „Printer-Plotter", die nicht nur Schriftzeichen, sondern auch Grafik wiedergeben können, und zwar mit Hilfe von Kugelschreiberpatronen. Die Daten vom Rechner werden zweidimensional in Papiervorschub und Schreibkopfbewegung umgesetzt, womit sich beliebige Muster erzeugen lassen.

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

computer kurs Heft 74

Manchmal müssen gleichlautende Briefe an verschiedene Adressaten geschickt werden. Kein Problem mit unserem Texteditor, den wir im nächsten PROGRAMMIER-SERVICE ausführlich vorstellen.



Die Textmaus

Als recht wirkungsvoll erweist sich das Textverarbeitungsprogramm McWrite für den Apple Macintosh.

Das Dateiprinzip

Die Sprache C bietet eine praktische Lösung des Problems freier und exakt definierter Befehlsformate.

Seriell gesteuert

Das Interface 1 des Spectrum eröffnet dem Maschinencode-Programmierer vielfältige Möglichkeiten.

Feine Zutaten

Vor dem Zusammenbau des Multimeters beschäftigen wir uns mit der Pinbelegung der Chips.

Kabelsalat

BTX-Systeme senden qualitativ hochwertige Bilder über Telefonleitungen. Wir erklären einige Systeme.