CLUB 80

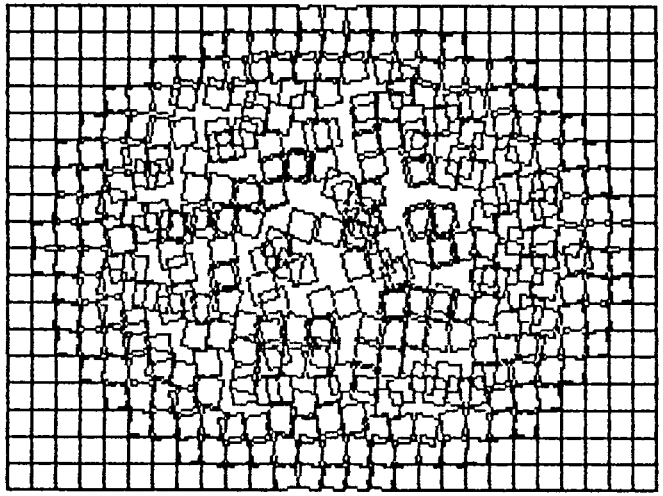
Clubinfo der

TANDY - GENIE - und KOMTEK - ANWENDER

7. AUSGABE

Kontaktadresse : CLUB 80 / Günther WARNER / Gartenstraße 9 / 5201 Neukirchen

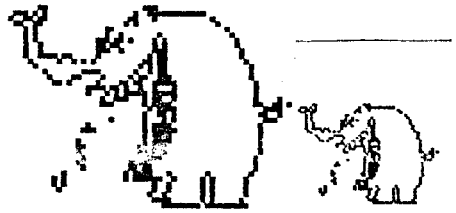
Tel.: 02245/3351 (18 - 20 Uhr)



UNORDNUNG und CRAZY MAMMUTS

erstellt mit

hochauflösender Grafik.



Inhaltsverzeichnis

Seite:

Clubinternes

Software

Hardware

Seite:

Börse

Sonstiges

Programmbibliothek

Club -Bücherei

Die letzten Seiten

Liebe Club-Freunde,

das 7. Info liegt mir im Entwurf vor - ich kann dem Jens dazu nur gratulieren und Ihm an dieser Stelle danken - danken für seine Bereitschaft, einen sehr wesentlichen Bestandteil des CLUB 80 (Gestaltung des Clubinfos und der damit verbundenen Arbeiten) verantwortlich zu übernehmen.

Das 7. Info und die zahlreichen Anrufe vom Jens zeigen sein großes Engagement um den Club. Doch auch er ist auf Euch Mitglieder angewiesen - nur wenn er genügend Beiträge bekommt, kann er auch ein wirklich gutes Clubinfo rausbringen. Auf dem Clubtreffen war es offensichtlich: Da ist soviel Wissen und Know-how da, da haben viele echt tolle Tips und Tricks auf Lager - doch veröffentlichen wollen Sie im Info nichts. Die einen trauen sich nicht, die anderen wollen nicht, manche meinen, es wäre nicht interessant genug.

Ich möchte also nun an a l l e Club-Mitglieder appellieren: Wenn Ihr was habt, so schreibt es auch - sei es auch noch so kurz. Der Jens freut sich bestimmt über jeden Beitrag.

Nun noch kurz zum Clubtreffen. Es hat prima eingeschlagen und findet nun Jährlich statt. Ich bin also doch noch 1 Jahr Vorstand - aber nur, weil sich der Jens bereit erklärt hat, den Löwenanteil der Arbeit zu übernehmen (den Vorstand mitzuübernehmen weigerte er sich leider - ein anderer fand sich nicht).

Ich hoffe, daß die neue Satzung von allen Clubmitgliedern anerkannt wird. Über 20 Mitglieder waren bei der Neugestaltung dabei und haben sich Gedanken gemacht, wie die neue Satzung auszusehen hat. Einig war man sich auch, den Beitrag auf 50 DM zu erhöhen und 10 DM nachzuzahlen. Diese nochmalige Erhöhung ist unbedingt nötig um den derzeitigen Umfang des Clubinfos beibehalten zu können.
Ich bitte also nun alle Mitglieder darum, den Stimmzettel am Ende dieses Clubinfos auszufüllen (er wurde übrigens so gestaltet, daß er nach üblicher Faltung in ein Fensterkuvert paßt). Bitte denkt daran - ein Info mit diesem Umfang und die Programm- und Bücherbibliothek sind nur dann möglich, wenn die Finanzierung gesichert ist. Diese ist nur dann gesichert, wenn wir den Beitrag auf 50 DM erhöhen. Die Nachzahlung in Höhe von 10 DM ist notwendig, damit wir dieses Jahr nicht wieder mit einem Minus abschneiden. Bitte beachtet auch die Seiten 3-5.

Übrigens: Den Stellenwert unseres (derzeit mindestens 64 Seiten umfassenden) Clubinfos können nachstehende Zeilen eindrucksvoll belegen. Peter Spieß stellte folgende Untersuchung an:

Er wollte herausbekommen, was die Zeitschrift "CHIP" an Informationen für den Leser bietet. Dazu hat er die Ausgabe 12/84 in die Einzelteile zerlegt. Was dabei herauskam findet Ihr in der folgenden Tabelle:

Seitenanzahl des Heftes: 360

121 Seiten ganzseitige Werbung
67 Seiten CHIP-Börse und Werbung
141 Seiten Informationen, wobei aber auf diesen Seiten teilweise kleinere Anzeigen abgedruckt sind.
31 Seiten Artrikel und Programme für Commodore VC 64

360
===

Das Heft kostet 6.00 DM; macht im Jahr 72 DM. Mit ein wenig Glück findet man auf Jahressicht gesehen einige wenige Seiten die die Computer der Firmen Tandy und EACA betreffen.

Der Club 80 kostet Jährlich 50 DM - es kommen 6 Infos heraus. Der Jahresumfang dürfte 400 Seiten übertreffen !!!

Ich glaube, die Entscheidung auf dem Stimmzettel dürfte nicht schwer fallen. Viel Spaß wünsche ich Euch mit diesem Info,

Euer Günther Wagner

S I N D W I R K R I M I N E L L ?
??????????????????????????????????????

Mit dieser Frage meine ich, ob wir Computer-Anwender kriminell sind ? - Wie ich dazu komme ? - Lest diesen Artikel und Ihr werdet verstehen, was ich meine. Am besten fange ich von vorne an.

ES WAR EINMAL ein 'liebes' Club- Mitglied namens Peter Schmidt, 7800 Freiburg - ein Schäflein unter vielen. Doch dieses Schäflein sollte sich schon bald als schwarzes Schaf herausstellen.
Denn - Peter Schmidt, Freiburg, war nur eine Deckadresse - eine Deckadresse für

" Hans Peter Schmid
Lenaustraße 2
6906 Leimen 2

Dieser vertreibt u.a. die Programme Newscript, Bugout, DOSPlus, Crashman, Faster, Accell, Edit, etc.

Und eben dieses Schmid'chen-Schleicher' hatte sich beim Hans König gemeldet. Er sei ein neues Club-Mitglied und suche schon lange die Programme Newscript und Bugout und ob er diese bekommen könnte.
Der Hans hat die Programme an den Schmid weitergegeben und dafür ein Schreiben vom Rechtsanwalt bekommen. Er würde Programme vertreiben; ein Streitwert von 12000 DM wurde genannt. Der Hans verweigerte natürlich jede Zahlung.

Was macht da unser 'lieber' Schmid ?

Nun - er stellt Strafanzeige gegen den Hans König und den Frank Smerling (dessen Name stand auf einer beigelegten Anleitung). Beiden wurde Ende April die Wohnung von der Polizei durchsucht - Disketten, Anleitungen und Schriftverkehr wurde mitgenommen!

Nun haben der Hans und der Frank wahrscheinlich ganz schönen Trouble - die Polizei hat die Adressen aller CLubmitglieder - und ich habe einen Teil meines Schriftverkehrs vernichtet. Mit diesem kurzen Artikel wollte ich informieren und warnen. Der Schmid hat seit Dezember 84 nichts mehr von mir erhalten (aber vielleicht ist ja unter uns ein weiteres schwarzes Schaf). Es könnte nicht schaden, wenn Jeder mal seinen Schriftverkehr ausmistet!

PS.: In der CHIP vom Mai 85 steht auf Seite 18:

"Es ist nicht verboten, Raubkopien privat zu nutzen. Aber vervielfältigen und weitergeben darf man sie nicht, auch nicht an einen Freund."

Bitte beiliegenden Stimmzettel am INFO-Ende beachten !!!

1. Mitgliedschaft, Organe und Aufgaben des Clubs

1.1 Der Club 80 ist die Gesamtheit der Mitglieder. Seine Aufgaben sind der Austausch von Erfahrungen, Programmen und Büchern durch Clubnachrichten, Versand und Clubtreffen.

1.2 Mitglied ist, wer auf Antrag aufgenommen wurde, die Clubsatzung anerkannt hat, die Aufnahmegebühr von DM 10,- und einen Jahresbeitrag von DM 50,- entrichtet hat.

1.3 Die Mitgliedschaft endet mit dem Ableben, dem Austritt oder dem Ausschluß. Bezahlte Beiträge können nicht zurückerstattet werden.

1.4 Der Ausschluß erfolgt
a) wenn nach zweimaliger Mahnung der Beitrag nicht entrichtet wurde
b) durch Beschluß der Mitglieder mit Dreiviertelmehrheit
c) bei Verstößen gegen die Clubsatzung.

Mitglieder, die nach b) oder c) ausgeschlossen wurden, können nicht wieder aufgenommen werden.

1.5 Organe des Clubs sind der Vorstand, das jährliche Clubtreffen und die Gesamtheit der Mitglieder.

1.6 Beschlüsse, die die Satzung betreffen, erfordern eine Dreiviertelmehrheit, andere Beschlüsse eine einfache Mehrheit der abgegebenen Stimmen.

1.7 Der Vorstand wird von der Gesamtheit der beim Clubtreffen anwesenden Mitglieder gewählt, nachdem der alte Vorstand entlastet worden ist.

1.8 Der Club wird auf Beschluß mit Dreiviertelmehrheit aufgelöst. Sein evtl. Vermögen wird gemeinnützigen Zwecken zugeführt.

2. Programmtausch

Der Programmtausch erfolgt über die Programmbibliothek des Clubs. Jedes Mitglied verpflichtet sich, nur Programme einzusenden, die frei von Rechten Dritter sind. Eine Überprüfung durch den Club kann nicht erfolgen. Die neuen Programme werden jeweils in den Clubnachrichten mit kurzer Beschreibung vorgestellt - eine Gesamtübersicht der Programmbibliothek erfolgt jeweils zum Jahreswechsel. Nur Clubmitglieder können Programme beziehen. Ein Verkauf von Programmen, die über den Club bezogen wurden, ist strengstens verboten, ebenso die Weitergabe an Dritte. Bei Zuwiderhandlung erfolgt sofortige Kündigung der Mitgliedschaft; strafrechtliche Verfolgung ist möglich.

3. Büchertausch

Computerbücher sind teuer, und oft sind für den Einzelnen nur wenige Seiten interessant. Es wird deshalb im Laufe der Zeit eine Bücherbibliothek angelegt. Jedes Clubmitglied kann jeweils ein Buch für vier Wochen kostenlos ausleihen. Ist das gewünschte Buch zur Zeit der Bestellung ausgeliehen, so wird das Mitglied auf eine Warteliste gesetzt. Das Buch wird auf Kosten des Entleihers versandt, der Besteller sendet das Buch auf seine Kosten zurück. Verlorengegangene Bücher müssen ersetzt werden. Wird die Ausleihfrist überschritten, erfolgt eine Mahnung in Höhe von DM 20,-, für jede weitere Mahnung DM 5,-. Die neuen Bücher werden in den Clubnachrichten veröffentlicht, eine komplette Liste erscheint jeweils zum Jahreswechsel.

4. Hardware und Verbrauchsmaterialien

Bei entsprechendem Interesse werden Sammelbestellungen für Hardware-Artikel und Verbrauchsmaterialien (z. B. Disketten) organisiert. Außerdem sind die Mitglieder aufgefordert, eigene Hardware-Entwicklungen auch dem Club zur Verfügung zu stellen.

5. Clubnachrichten

Diese erscheinen unregelmäßig (etwa alle zwei Monate). Der Inhalt ist ersichtlich aus den übrigen Punkten der Clubsatzung. Falls möglich, wird auch gewerbliche Reklame gegen Gebühr abgedruckt.

6. Mitgliedsbeitrag

Der Club verursacht Kosten, z. B. die Programmarchivierung, der Druck der Clubnachrichten, Porto usw.. Natürlich werden die Ausgaben so gering wie möglich gehalten, aber ein Mitgliedsbeitrag ist nötig. Der Aufnahmebeitrag beträgt DM 10,-, der zum Jahreswechsel fällige Jahresbeitrag DM 50,-. Erfolgt die Aufnahme in den Club nach dem 31. Juli, ist nur der halbe Jahresbeitrag für das erste Mitgliedsjahr zu entrichten.

7. Clubkasse

Aus der Clubkasse werden die laufenden Unkosten bezahlt. Größere Überschüsse werden zum Ankauf weiterer Programme und Bücher verwandt. In den Clubnachrichten erscheint zu gegebener Zeit eine Aufstellung der zu kaufenden Artikel. Ein Rechenschaftsbericht über die Clubkasse erfolgt jeweils zum Jahreswechsel in den Clubnachrichten.
Für die Mitglieder bestehen folgende Zahlungsmöglichkeiten:
a) Barzahlung
b) Scheck
c) überweisung auf Kto. Nr. 194 712 bei der
Sparkasse Rosenheim, BLZ 711 500 00, PSchKto. Nr. 8077-801

8. Austritt

Jedes Mitglied kann jederzeit aus dem Club austreten. Der Austritt sollte schriftlich und unter Angabe von Gründen erfolgen.

9. Änderung der Clubsatzung

Jedes Mitglied kann eine Änderung der Clubsatzung unter Angabe von Gründen beantragen. Dieser Antrag wird in den Clubnachrichten veröffentlicht, und die Mitglieder entscheiden sich für oder gegen die beantragte Satzungsänderung mit einer Dreiviertelmehrheit.

10. Gültigkeit der Satzung

Diese Satzung ist nach ihrer Genehmigung per Abstimmung über das Clubinfo Nr. 7 in Kraft und für alle Mitglieder verbindlich. Frühere Fassungen der Satzung sind danach ungültig. Ihre Gültigkeit bzw. eine geänderte Fassung (je nach Abstimmungsergebnis) wird im Clubinfo Nr. 8 bekanntgegeben.

Protokoll des Clubtreffens vom 23./24. März in Reinheim

Vorsitz: Günther Wagner
Protokoll: Arnulf Sopp

Anwesende: die obigen, Wolfg. Beckhausen, Ulrich Böckling, Gerald Dreyer, Manfred Held, Klaus Hermann, Dieter Kasper, Hans J. König, Josef Konrad, Jens Neueder, Hartmut Obermann, Walter Piller, Heinrich Rank, Christian Schrewe, Gerald Schröder, Frank Smerling, Holm Voigtländer, Alexander Wagner, Jürgen Wucherer, Walter Zwickel,
sowie Ehefrauen von Mitgliedern, die jedoch nur aus touristischen Gründen angereist waren und nicht mit abstimmten.

Beginn: 23. 3. 85, 17 Uhr
Ende: 24. 3. 85 gegen Mittag

Verlauf:

1. Beschlußfähigkeit:
Einstimmiger Beschluß, daß der Vorschlag zur Beschlußfähigkeit aus dem Clubinfo Nr. 6 zu befolgen ist: Das Clubtreffen erklärt sich als Organ des Clubs als entscheidungsbefugt, weil keine Einwände gegen den Info-Vorschlag vorliegen.

2. Programmbibliothek:
Hartmut verwaltet die Bibliothek. Sie umfaßt ca. 190 Programme ohne Rechte Dritter. Entsprechend Hartmuts Hardware können Disketten bis 40/SS/DD eingesandt und abgefordert werden.
Das Punktesystem wird mit einstimmigem Beschluß abgeschafft.
Es ist geplant, Ordner mit Listings herumzuschicken, von denen jedes Mitglied eines abtippen soll, um die Programmbibliothek zu erweitern.

3. Entlastung des Vorstands:
Wahl von Frank und Jürgen als Kassenprüfer. Nach Prüfung wird der Vorstand einstimmig entlastet.

4. Vorstand, Gestaltung des Clubinfos:
Die Tätigkeiten des bisherigen Vorstands werden verteilt: Ansprechadresse für alle den Club betreffenden Fragen und damit Vorstand bleibt Günther. Die Zusammenstellung des Clubinfos übernimmt Jens. Beide einstimmig gewählt. Wer die Kopien anfertigen wird, ist noch nicht geklärt. Mehrere Mitglieder erkundigen sich nach möglichst günstigen Angeboten.

5. Beiträge:
Der Mitgliedsbeitrag beträgt rückwirkend zum 1. 1. 85 DM 50,-. Dieser Betrag gilt für neue Mitglieder. Bisherige Mitglieder zahlen DM 10,- nach. Hierzu erfolgt noch ein endgültiger Beschluß über das Info.

6. Micro 80, Load 80:
Holm übersetzt Programmanleitungen und Artikel aus Micro 80 ins Deutsche. Ein Abonnement von Load 80 auf Clubkosten wird abgelehnt, weil der Nutzen zu gering ist.

7. Satzung:
Die neue Satzung (s. nächste Seiten) wird entworfen und einstimmig verabschiedet. Über die Gültigkeit entscheidet eine Abstimmung über das Clubinfo.

8. Hardware:
Walter Z. stellt eine Idee für eine universelle I/O-Karte vor. Mehrere

Clubkameraden stellen sich vor !!

Ich heiße Eckehard Kuhn, bin 21 Jahre alt und Schüler an einem Technischen Gymnasium. In diesem Jahr fange ich eine Ausbildung zum Informationselektroniker an.
Vor etwa zwei Jahren erwarb ich einen TRS-80 M 1 und legte mir bald darauf ein Expansion Interface zu. Zur Zeit suche ich ein günstiges Laufwerk. (Wer hätte eins?) Meinen Computer benutze ich vor allem zum Erstellen von Programmen, (Basic + Assembler) im Bereich der Chemie. Zum Beispiel: elektronische ph-Wert Messung über Computer.

Eckehard Kuhn

Beckhausen Wolfgang
Vuerfelser-Kaule 30
5060 Bergisch-Gladbach 1
02204/62781

Clubmitglied seit : 08.03.85
Punktestand = 60
System- und Drivekonfiguration :
Komtek 1 Super/ Epson FX-80/ 2 Laufwerke (je 40 Spuren/ ds/ sd
Schaltbox f r Leitungen bis 10 A
bevorzugtes Betriebssystem : NEWDOS 80

Schaefer Walter
Rathausstr. 4
8160 Miesbach
08025/1631

Clubmitglied seit : 11.03.85
Punktestand = 60
System- und Drivekonfiguration :
Genie I (Mod. 83)/ Star Gemini 10x/ 2 Laufwerke TCS 820/2 FC
bevorzugtes Betriebssystem : G-DOS 2.16

Schneider Manfred
Rheinkasseler Weg 11
5000 Koeln 71
0221/707044

Clubmitglied seit : 09.04.85
Punktestand = 60
System- und Drivekonfiguration :
TRS-80 Model I Level II mit 48 K/ Drucker NEC PC-8023 B-C/ 3 Laufwerke (40 Spuren/ 1 * ss/sd und 2 * ds/dd)/ RB RS-232/ RB HRG1B
bevorzugtes Betriebssystem : NEWDOS 80 V2

Spiess Peter
Trugenhofenerstr. 27
8859 Rennertshofen
08434/454

Clubmitglied seit : 16.03.85
Punktestand = 45
System- und Drivekonfiguration :
Video Genie II (Mod. 83)/ TRS-80 Model 3 und 4/ Drucker NEC 8023 B-C/ Itoh 8510A und 1550/ HRG1b/ RS-232/ Grafikkarte 512*512/ Modem (Tandy)/ 3 Laufwerke (40 Spuren und 80 Spuren jeweils ds/ dd)
bevorzugtes Betriebssystem : NEWDOS 80 und Multidos

Helmut Paulo

Lokale Variable beim TRS-80

Einer der Vorteile von Pascal ist die Möglichkeit, Variablen in einzelnen Programmteilen lokal verwenden zu können, ohne die Verwendung einer Variablen gleichen Namens in einem anderen Programmteil zu behindern. Der folgende Beitrag zeigt nicht nur einen Weg, so etwas auch in Basic zu realisieren, sondern auch, wie man Programme ohne Variablenverlust verketten kann.

Für einen Neuling kommt zur Einführung das Programmieren mit einem Pascal-Compiler überhaupt nicht in Frage, weil das Verfahren viel zu umständlich ist (editieren, speichern, compilieren, linken usw.), so daß ein Anfänger mit einem solchen System schnell seine Versuche frustiert beenden würde. Für den Anfängerunterricht gibt es also zur Zeit keine Alternative zu Basic. Hinzu kommt, daß man z. B. beim TRS-80 in Basic zahlreiche Vorteile hat, die in Pascal nicht gegeben sind (doppelte Genauigkeit, Grafik). Schließlich gibt es ja auch noch SBasic, womit ein strukturiert geschriebenes Programm erstellt werden kann. Da dieses Programm durch den Precompiler von SBasic in ein ganz gewöhnliches Basic-Programm verwandelt wird, ist gezeigt, daß man auch in Basic

```
FC00  21FF FB22 4940 C32D 40CD 7F0A 7D32 75FE  !.."I$.-$....2..
FC10  FE1E CA78 FDFE 28CA 7BFD FE0A CA61 FCFE  .....x..(.{....a..
FC20  14CA 35FC 110C 0021 E9FD 3C47 1910 FD22  ..5....!..<G...."
FC30  6FFE C32A FD2A A440 E523 2323 237E B7C2  ...*.*.$.####...
FC40  3CFC 23D1 7D12 137C 1211 FAFF CD2C 1823  <.#.....,..#
FC50  2322 F940 22FB 4022 FD40 2A49 4022 B140  #"."$".$*I$".$
FC60  C93A 71FE EE01 3271 FECA FAFC 2AB1 4022  .:....2.....*.$"
FC70  01FE 2AD6 4022 05FE 2AA0 4022 03FE 2114  ..*.$"..*.$"..!.
FC80  FDCD A728 3E02 32AF 40CD B31B 23CD 6C0E  ...(>.2.$...#...
FC90  2A21 4111 00FC CDC7 0B22 F5FD 2322 07FE  *!A....."..#"..
FCA0  2209 FE22 0BFE 1100 FEDD 210D FE3A F4FD  "..".......!..:..
FCB0  B7CA E5FC 472B DD75 00DD 7401 DD75 04DD  ....G+.....G+....
FCC0  7405 19DD 7502 DD74 0319 DD75 06DD 7407  ..............
FCD0  DD75 0BDD 7409 DD75 0ADD 740B D511 0C00  ..............
FCE0  DD19 D110 D02B 22F5 FD22 B140 11F5 FD21  .....+"..".$...!
FCF0  6DFE CDF6 FCC9 7323 72C9 21F9 4011 FBFD  .......#..!.$...

FD00  0106 00ED B02A D640 22F9 FD2A A040 22F7  .....*.$"..*.$".
FD10  FDC3 EDFD 4752 4F45 5353 4520 564F 4E20  ....GROESSE.VON.
FD20  4245 5245 4943 4820 3100 2A6D FE11 0400  BEREICH.1.*.....
FD30  19ED 5BD6 40CD F6FC 23EB 21F9 4001 0600  ..A.$...#.!.$...
FD40  EDB0 ED5B 6FFE 216D FECD F6FC 2A6F FE7E  ...A..!....*...
FD50  32B1 4023 7E32 B240 237E 32A0 4023 7E32  2.$#.2.$#.2.$#.2
FD60  A140 237E 32D6 4023 7E32 D740 2311 F940  .$#.2.$#.2.$#..$
FD70  0106 00ED B0C3 EDFD 5A24 003E 0132 74FE  ........Z$.>.2..
FD80  217B FDCD 0D26 131A 6F13 1A67 11B2 FED5  !....&.....
FD90  1176 FE7E FE20 CAA5 FDFE 22CA DAFD FE2C  .....".....",
FDA0  CAA9 FD12 1323 C393 FDAF 12D1 23E5 D521  .....#.....#..!
FDB0  76FE CD0D 263A 75FE FE2B CAE1 FDD5 E1D1  ....&:...(......
FDC0  2B2B 2B7E 1223 2323 1306 004F EDB0 E13A  +++..###...O...:
FDD0  74FE B7CA EDFD D5C3 90FD AF32 74FE C3A9  ...........2....
FDE0  FDE1 7E06 004F 23ED B0EB C3CE FD3E 0232  .....O#.....>.2
FDF0  AF40 C900 0300 0000 0000 00B2 6D00 0000  .$..............
```

Bild 1. Maschinenprogramm zum Umschalten zwischen bis zu acht Variablen-Bereichen für einen 48-KByte-TRS-80

Formulierung. Eine wichtige Eigenschaft von Pascal hat allerdings Basic nicht: die Verwendung von lokalen und globalen Variablen. Der folgende Beitrag zeigt, wie man beim TRS-80 auch diesen Mangel beseitigen kann.

Globale und lokale Variable

Benötigt wird das Maschinenprogramm (*Bild 1*), das als CMD-File auf Diskette zu speichern ist (START=TRA=FC00, END=FDFF). Nach Aufruf dieses Programms vom DOS aus geht man ins Basic. Das Basic-Programm (*Bild 2*) demonstriert, wie man lokale und auch globale Variable bekommt. Hierzu muß man eine USR-Funktion benutzen (DEFUSR =&HFC09), die mit unterschiedlichen Argumenten aufzurufen ist. Zunächst werden mit Hilfe von Zeile 40 die beiden Hauptbereiche 0 und 1 für die Variablen initialisiert, die CLEAR-Statements reservieren den gewünschten String-Platz. Danach befindet man sich im Bereich 0, in dem man Programme laden, ändern und speichern kann. Mit Zeile 70 wird der Stack initialisiert, und nun kann man jeweils durch Aufruf von Z=USR(n) in den jeweils gewünschten Variablenbereich Nr. n umschalten. Normalerweise stehen die beiden Hauptbereiche und drei weitere Bereiche mit je 1 KByte zur Verfügung. Die Anzahl dieser zusätzlichen Bereiche steht in Adresse FDF4 und kann bis acht erhöht werden. In allen Bereichen existieren die Variablen völlig unabhängig voneinander, so daß man beim Aufruf von Unterprogrammen unabhängig von den Variablennamen ist! Zum Beispiel stört die Verwendung der Variablen I im Bereich 0 nicht die Verwendung von I im Unterprogramm in Zeile 280 im Bereich 1. Der Vorteil dieses Verfahrens ist, daß man nun ein Programm wirklich modular aufbauen kann, ohne an bestimmte Variablennamen gebunden zu sein. Die Verwendung von lokalen Variablen z. B. in Unterprogrammen hätte wenig Wert, wenn es nicht möglich wäre, Variablenwerte von einem zum anderen Bereich zu übergeben. Dies geschieht mit dem Statement Z$=".....":Z=USR(30). Die in Z$ aufgeführten Variablen (Zeile 140) werden zur Übergabe bereitgestellt; nach Umschalten auf einen anderen Bereich werden sie mit Z$=".....":Z=USR(40) übernommen (Zeile 240). Hierbei müssen jedoch keineswegs die Variablennamen übereinstimmen, es kommt lediglich auf die Reihenfolge an, die allerdings typenentsprechend sein muß. Wenn man z. B. das Demonstrationsprogramm mehrmals laufen lassen will, so muß man nach dem ersten Durchgang nicht von vorne beginnen, sondern mit RUN 70 bei Zeile 70. Dieses Statement ist bei jedem neuen Start erforderlich, damit der Basic-Stack in den richtigen Bereich (und zwar Bereich 1) verlagert wird. Bekommt man aus irgendeinem Grund eine OM-Meldung, so startet man neu mit NEW und PRINT USR(20). Hierdurch wird das gelöschte Programm reaktiviert, und alle Variablenpointer werden in den Start-Zustand gebracht.

```
10    DEFUSR=&HFC09
20    CLS
30    'INITIALISIERUNG VON BEREICH 0 UND BEREICH 1
40    CLEAR800:Z=USR(10):CLEAR500:Z=USR(10)
50    '
60    'INITIALISIERUNG DES BASIC-STACKS
70    Z=USR(1):CLEAR:Z=USR(0)
80    ' ************************
90    CLS
100   A$="TESTVARIABLE"+""
110   Z=USR(1):X$="STRING IN BEREICH 1"+"":Z=USR(0)
120   FOR I=1 TO 3
130   INPUT"A,B";A,B
140   Z$="A,B,A$":Z=USR(30) :' VARIABLENUEBERGABE AN BEREICH 1
150   GOSUB220
160   Z$="C":Z=USR(40) :'      VARAIABLENUEBERNAHME VON BEREICH 1
170   PRINTA,B,C
180   PRINTA$
190   NEXT
200   END
210   ' ***********************
220   ' PYTHAGORAS
230   Z=USR(1) :'                UMSCHALTEN AUF BEREICH 1
240   Z$="U,V,P$":Z=USR(40) :' VARIABLENUEBERNAHME VON BEREICH 0
250   W=U*U+V*V
260   Z$="W":Z=USR(30) :'      VARIABLENUEBERGABE AN BEREICH 0
270   PRINTX$,P$
280   FOR I=1 TO 5: PRINTI;:NEXT:PRINT
290   Z=USR(0) :'                UMSCHALTEN AUF BEREICH 0
300   RETURN
```

Bild 2. Demonstrationsprogramm für die Verwendung lokaler und globaler Variablen in zwei Bereichen

Programm-Verkettung

Eine ganz andere Möglichkeit, ein Programm modular aufzubauen, ist die Programm-Verkettung. Normalerweise werden bei einem neuen RUN alle Variablen gelöscht. Dies kann man mit dem in *Bild 3* gezeigten System vermeiden. PROGR1 dient zur Initialisierung und wird nur einmal benötigt. Danach kann man beliebig viele Teilprogramme aufrufen. Mit dem ersten Statement Z=USR(1) schaltet man auf einen geschützten Variablenbereich, mit Z=USR(2) schaltet man zurück auf den Normalbereich, in dem man Programme laden, ändern und speichern kann, ohne daß die Variablen von Bereich 1 verlorengehen. So kann man auch bei geringer Speicherkapazität praktisch beliebig lange Programme laufen lassen.

```
10    ' PROGR1
20    ' ZUR INITIALISIERUNG
30    CLS
40    INPUT"RAM-GROESSE (16 - 32 -48)";R
50    IF R=48 THEN H=254
60    IF R=32 THEN H=190
70    IF R=16 THEN H=126
80    L=255
90    POKE 16561,L:POKE 16562,H: POKE 16457,L: POKE 16458,H: CLEAR 50
100   H=PEEK(16562): D=65536: IF H=126 THEN D=0
110   M=(H+1)*256-D
120   DEFUSR=M
130   FOR I=0 TO 99
140     READ A: IF A=255 THEN A=H+1
150     POKE M+I,A
160   NEXT
170   DATA 205,127,10,124,183,32,11,125
180   DATA 254,1,40,34,254,2,40,55
190   DATA 24,0,237,91,249,64,25,34
200   DATA 106,255,34,108,255,34,110,255
210   DATA 42,214,64,34,112,255,33,249
220   DATA 64,17,100,255,24,48,33,249
230   DATA 64,17,100,255,1,6,0,237
240   DATA 176,42,112,255,34,214,64,33
250   DATA 106,255,17,249,64,24,23,33
260   DATA 249,64,17,106,255,1,6,0
270   DATA 237,176,42,214,64,34,112,255
280   DATA 33,100,255,17,249,64,1,6
290   DATA 0,237,176,201
300   INPUT"MAXIMALE PROGRAMMLAENGE IN BYTES ";L
310   Z=USR(L)
320   RUN "PROGR2"
```

```
10    ' PROGR2
20    Z=USR(1)
30    DEFINT N
40    DIM A(50)
50    A(33)=100: N=50
60    C$="TEST"+""
70    D$=C$+"-PROGRAMM"
80    PRINT A(33),N
90    PRINTD$,C$
100   Z=USR(2)
110   RUN"PROGR3"
```

```
10    ' PROGR3
20    Z=USR(1)
30    DEFINT N
40    X=17.4
50    Z$=D$+"PROGRAMM3"
60    A(34)=3.4
70    PRINTA(33),N
80    PRINTC$,D$
90    PRINTX,Z$
100   Z=USR(2)
```

Bild 3. Beispiel für die Verkettung von drei Programmen, die einzeln auf Diskette gespeichert sind

BASIC-ÜBERLAGERUNGEN

VARIABLENAUSTAUSCH ZWISCHEN PROGRAMMEN

Immer wenn Sie das RUN- oder LOAD-Kommando benutzen, werden die Inhalte aller benutzten Variablen gelöscht, so daß das Programm mit einem bereinigtem Speicher beginnt. Aber es gibt viele Situationen, bei denen man die Variablenwerte von einem Programm in ein anderes übernehmen will.

Wenn Sie Varibalen zwischen Programmen austauschen können, ist es möglich, daß Sie ein Programm in kleinere Teilprogramme unterteilen. Mit kleineren Programmen haben Sie mehr Speicher zur Verfügung. Ein Programm könnte z.B. die Daten von Diskette laden oder die Eingabe über die Tastatur regeln, ein zweites Programm arbeitet mit den Daten und ein drittes Programm kümmert sich um den Ausdruck.

Bevor Sie die Hilfsroutinen für den Variablenaustausch benutzen können, müssen Sie wissen, daß die Variablen gleich nach dem BASIC-Programmtext im Speicher gespeichert werden. Als Beispiel wollen wir annehmen, daß Sie folgendes Programm geschrieben haben:

```
10  X%=1
20  A%=2
30  S$=STRING$(5,"X")
```

Wenn Sie das Programm starten wird der Inhalt von X% gleich nach der letzten Adresse des BASIC-Programmtextes abgespeichert. Der Inhalt von A% wird nach dem Inhalt von X% gespeichert, gefolgt von einem Zeiger, welcher die Länge und Lage des Inhalts von S$ anzeigt. Die fünf 'X' von S$ werden direkt unterhalb der höchsten Speicheradresse (Festlegung durch MEMORY SIZE) abgespeichert. Definieren Sie in einem Programm eins oder mehrere Felder, so werden diese unmittelbar nach den einfachen Variablen gespeichert.

Der Speicherbereich, der alle Variablennamen, Typencodes, Dimensionen, Zahlenwerte und String-Zeiger speichert, wird Variablenliste genannt. Da die Variablenliste unmittelbar nach dem Programmtext beginnt, ist der Beginn der Variablenliste abhängig von der Länge des geladenen Programms. Zur Übergabe von Variablen setzen wir uns über dieses Merkmal von BASIC hinweg und entscheiden uns für eine bestimmte Stelle an der die Variablenliste beginnen soll. Die von uns gewählte Stelle wird gleich nach der Endadresse des längsten benutzten Programms beginnen.

So findet man die erste verfügbare Adresse nach der Endadresse des längsten Programms:

1. Laden Sie Ihr längstes Programm und beantworten Sie die Frage 'HOW MANY FILES?' genauso, wie Sie es später in der praktischen Anwendung tun.

2. Geben Sie folgende Anweisungen ein:

   ```
   CLEAR
   PRINT CVI(CHR$(PEEK(&H40F9))+CHR$(PEEK(&H40FA)))
   ```

3. Addieren Sie 17 zu der angezeigten Zahl. Das Ergebnis ist die niedrigste Adresse, die Sie für den Beginn Ihrer Variablenliste verwenden sollten, falls Sie Variablen von einem Programm zu einem anderen übergeben wollen. In der Praxis wird man gewöhnlich noch einen gewissen Spielraum einräumen; Sie brauchen diesen z.B. wenn durch Änderungen das Programm länger wird. Addieren Sie also nochmals einen Wert hinzu, z.B. 300.

Wie schaffen wir es nun, daß die Variablen ab der von uns gewählten festen Adresse abgespeichert werden? Im ersten Programm das wir starten wird als eine der ersten Anweisungen ein 'GOSUB 52000' ausgeführt. Dieses GOSUB muß auf alle Fälle vor der Benutzung irgendeiner Variablen erfolgen. Die Hilfsroutine 52000 ändert 3 Zeiger im BASIC. Diese bestimmen Anfang und Ende der benutzten Variablen:

```
52000  A$="": FOR A%=1 TO 3: A$=A$+MKI$(30000): NEXT: AN$=
       "XXXXXX": POKE VARPTR(AN$)+1,&HF9: POKE VARPTR (AN$)
       +2,&H40: LSET AN$=A$: A$="": RETURN
```

Sie müssen anstelle der '30000' die Adresse schreiben, bei der Ihre Variablenliste beginnen soll.

BEACHTE: Die Hilfsroutine 52000 benutzt eine interessante Methode für das POKEN der neuen Zeiger in die 6 Bytes beginnend bei 40F9. Zunächst wird der String A$ erstellt, der die 6 Bytes enthält, welche gepoked werden sollen. Dann wird VARPTR von AN$ so geändert, daß AN$ auf die Adresse 40F9 zeigt. Schließlich führen wir ein LSET von A$ in AN$ aus. Das LSET-Kommando ergibt ein sofortiges POKEN der 6 Bytes. Hätten wir versucht, die 6 Bytes mit einzelnen POKE-Anweisungen zu poken, so hätte sich im BASIC ein Fehler ergeben, da der erste 2-Byte-Zeiger nach der ersten Anweisung nur 'zur Hälfte' gepoked gewesen wäre.

Das abschließende A$="" in der Hilfsroutine 52000 bewirkt, daß A$ die erste initialisierte Varibale wird. Die 'Variablen-Übergabe'-Hilfsroutine und die 'Variablen-Empfänger'-Hilfsroutine erwarten, daß A$ die erste Variable der Variablenliste ist.

Die Hilfsroutine 52100 ist die 'Variablen-Übergabe'-Hilfsroutine. Wenn Sie Variablen von einem Programm an ein anderes Programm übergeben wollen, so müssen Sie ein 'GOSUB 52100' ausführen. Anschließend starten Sie das neue Programm mit RUN. Die Hilfsroutine 52100 lädt A$ mit allen Zeigern die BASIC laufend unterhält. Unter anderem beinhalten die 104 Bytes, die in A$ geladen werden, die Startadressen der einfachen Variablen, Anfang und Ende beliebiger Felder welche aktiv sind, den gegenwärtigen Zustand im String-Speicher und die Typen-Vereinbarungen (DEFSTR, DEFINT, DEFSNG oder DEFDBL) die aktiv sein können.

```
52100 AN$="": POKE VARPTR(AN$),104: POKE VARPTR(AN$)+1,&HB3:
      POKE VARPTR(AN$)+2,&H40: A$=STRING$(104,0): LSET A$=
      AN$: RETURN
```

Die letzte Anforderung für die Technik der Variablen-Übergabe besteht darin, daß das Programm die Variablen wieder erreichen muß. Im neuen Programm sollte daher die erste Anweisung ein 'GOSUB 52200' sein. Die Programmzeile, welche die Hilfsroutine 52200 aufruft, darf keine weiteren Ausdrücke enthalten. Die Hilfsroutine 52200 ist die 'Variablen-Empfänger'-Hilfsroutine. Sie muß die feste Adresse kenne, die Sie als Beginn des Variablen-Speichers festgelegt haben. Dies und die Kenntnis darüber, daß A$ die erste definierte Variable im vorhergehenden Programm war, erlaubt der Routine die Rekonstruktion einer vorläufigen Variable A$ zur Wiedererlangung der 104 Bytes. In diesen 104 Bytes sind alle Zeiger enthalten, die im vorhergehenden Programm abgespeichert worden sind. Schließlich zeigt AN$ auf den Bereich der BASIC-Verwaltung und 'poked' sogleich die 104 Bytes mit einem LSET-Befehl zurück.

```
52200  A$="": FOR A%=0 TO 2: POKE VARPTR(A$)+A%, PEEK(30000
       +A%+3): NEXT: AN$="": POKE VARPTR(AN$),104: POKE
       VARPTR(AN$)+1,&HB3: POKE VARPTR(AN$)+2,&H40: LSET
       AN$=A$: RETURN
```

Sie müssen die '30000' in der Hilfsroutine 52200 in die Adresse abändern, die Sie als Beginn der Variablen-Liste festgelegt haben.

Zur Demonstration der Arbeitsweise der Variablen-Übergabe-Technik können Sie die zwei folgenden Programme verwenden. Das Programm VARPASS/DEM initialisiert die Variablenliste an der Speicherstelle 30000. Anschließend erzeugt das Programm verschiedene Variablen und zeigt diese an.

Schließlich ruft das Programm die 'Variablen-Übergabe'-Hilfsroutine auf und startet dann mit RUN das Programm VARPASS/RCV, welches zunächst die von VARPASS/DEM erzeugten Variablen wiederherstellt. Dies geschieht durch Aufruf der Hilfsroutine 52200. In Zeile 2 wird A$ in einen Null-String zurückgesetzt, da die 104 Bytes, welche wir für die Übergabe der BASIC-Zeiger benutzten, nicht länger gebraucht werden. Schließlich zeigt das Programm die wiederhergestellten Variablen an.

Sie müssen sich bewußt sein, daß das Programm VARPASS/RCV erst gestartet werden kann, wenn durch das Programm VARPASS/DEM die Hilfsroutinen 52000 und 52100 angesprungen worden sind.

```
   0  'VARPASS/DEM
   1  CLEAR 150
   2  GOSUB 52000

  20  C$="CAT"+"": D$="DOG"+""
  30  DATA 1,2,3,4,5,6,7,8,9,10
  31  FOR X=1 TO 10: READ A%(X): NEXT
  40  A!=123: A#=456

 100  CLS
 110  PRINT "PROGRAMM 1 - VARIABLEN SIND:"
 120  PRINT "C$="; C$; TAB(20); "D$="; D$
 130  PRINT "A%()=";: FOR X=1 TO 10: PRINT A%(X);: NEXT: PRINT
 140  PRINT "A!="; A!; TAB(20); "A#="; A#
 200  GOSUB 52100: RUN "VARPASS/RCV"

52000  A$="": FOR A%=1 TO 3: A$=A$+MKI$(30000): NEXT: AN$=
       "XXXXXX": POKE VARPTR(AN$)+1,&HF9: POKE VARPTR(AN$)+2,
       &H40: LSET AN$=A$: A$="": RETURN
52100  AN$="": POKE VARPTR(AN$),104: POKE VARPTR(AN$)+1,&HB3:
       POKE VARPTR(AN$)+2,&H40: A$=STRING$(104,0): LSET A$=
       AN$: RETURN
```

```
   0  'VARPASS/RCV
   1  GOSUB 52200
   2  A$=""

 100  CLS
 110  PRINT "PROGRAMM 2 - VARIABLEN SIND:"
 120  PRINT "C$="; C$; TAB(20); "D$="; D$
 130  PRINT "A%()=";: FOR X=1 TO 10: PRINT A%(X);: NEXT: PRINT
 140  PRINT "A!="; A!; TAB(20); "A#="; A#
 200  END

52200  A$="": FOR A%=0 TO 2: POKE VARPTR(A$)+A%,PEEK(30000+A%
       +3): NEXT: AN$="": POKE VARPTR(AN$),104: POKE VARPTR
       (AN$)+1,&HB3: POKE VARPTR(AN$)+2,&H40: LSET AN$=A$:
       RETURN
```

Hartmut Bulling

TRS-80: Eingaberoutine mit Komfort

Die Funktion INKEY$ beim TRS-80 und beim Video-Genie eignet sich nur bedingt zur Eingabe von Zeichenketten von der Tastatur aus. Bei vielen Anwendungen ist der Basic-Interpreter für diesen Zweck zu langsam.

Das Programm KBREAD/SOU (*Bild 1*) behebt diese Mängel. Im einzelnen hat es folgende Eigenschaften:

- von Basic aus definierbare Eingabefeldlänge,
- alle Sonderzeichen erlaubt (inkl. „Komma"),
- Umlaut „ä" von der Tastatur erreichbar,
- akustische Eingabekontrolle (Frequenz und Tonlänge einstellbar),
- Sonderfunktionen (Clear, Break, Shift-Rechtspfeil, Ctrl) ausgeblendet,
- direkte Übernahme der Eingabe in Basic-String.

Das Programm arbeitet in jedem Speicherbereich und kann daher in allen Speicherkonfigurationen verwendet werden. Einzige Voraussetzung ist ein Level-II-ROM, da auf einige ROM-Routinen zurückgegriffen wird.

KBREAD/SOU ist eine modifizierte Version des Tastaturtreibers im ROM (Adresse 05D9H...0673H). Alle gegenüber dem Original-ROM ausgelassenen Zeilen wurden für weitergehende Änderungen bzw. für die Reaktivierung der ausgeblendeten Tasten als Kommentarzeilen in das Listing übernommen. Vorsicht: Die Taste Shift-Rechtspfeil (ASC 19H) sollte desaktiviert bleiben! Folgende ROM-Routinen wurden verwendet:

- Adresse 0033H: zeigt den Inhalt des A-Registers an. Steuerzeichen sind erlaubt.
- Adresse 0049H: Rücksprung, sobald eine Taste gedrückt wurde. Der ASCII-Wert der gedrückten Taste steht im A-Register. Benutzt A-, Status- und DE-Register.
- Adresse 01C9H: Löscht den Bildschirm, wählt 64 Zeichen/Zeile und setzt den Cursor an den linken, oberen Bildrand.
- Adresse 0A7FH: Bringt die von Basic übergebene Variable in das HL-Register.
- Adresse 0A9AH: Übergibt den Wert des HL-Registers an Basic.

Mit Ausnahme des zweiten Unterprogramms (Adresse 0049H) benutzen die Routinen sämtliche Register.

Von Basic aus leicht zu benutzen

Das Beispielprogramm KBREAD/BAS (*Bild 2*) liest mit Hilfe der neuen Funktion einen Adreßsatz und druckt ihn anschließend.

Das Maschinenprogramm muß vorher in einen geschützten Speicherbereich geladen worden sein. Sollte Ihnen kein Assembler zur Verfügung stehen, können Sie das Programm mit Hilfe von KBREAD/DAT (*Bild 3*) in den Speicher bringen.

Der Aufruf erfolgt mit dem USR-Befehl (Bild 2, Zeilen 1020 und 10030). Die im USR-Aufruf übergebene Variable „L" bestimmt die maximale Länge der zu lesenden Zeichenkette. Bei Rückkehr ins Basic enthält die Variable A$ den gelesenen String.

Wenn Sie weniger Speicher haben

Die vorgestellten Programme setzen 48 KByte RAM voraus. Sollten Sie über weniger Speicherplatz verfügen, so brauchen Sie lediglich den ORG-Befehl (Zeile 300 in KBREAD/SOU) entsprechend Ihrer Konfiguration zu ändern. Wollen Sie das Programm per POKE-Befehl in den Speicher bringen, müssen Sie in Zeile 500 von Bild 3 die benötigten Adressen eintragen. Weiterhin ändern Sie den Wert 255 bis inkl. Zeile 1120 jeweils in 127 für die 16-KByte-Version bzw. in 191 für die 32-KByte-Version.
Die Zahlenfolge 234,254 in Zeile 1030 ändern Sie in 234,126 bzw. in 234,190. HIMEM bzw. MEMSIZE setzen Sie jeweils 296 Byte unter das Ende Ihres Speichers.

Programmänderungen leicht durchzuführen

Tonfrequenz und Tondauer können Sie leicht verändern. Sollten Sie im Besitz eines Assemblers sein, ändern Sie in KBREAD/SOU die Zeilen 2120 und 2130 entsprechend Ihren Vorstellungen.
Im Programm KBREAD/DAT ist der Wert 4 in Zeile 1130 ein Maß für die Tondauer, der Wert 120 in den Zeilen 1130 und 1140 ein Maß für die Tonfrequenz.

Den Namen der Variablen, in die die Zeichenkette gelesen wird, finden Sie in KBREAD/SOU in Zeile 2300, in KBREAD/DAT in Zeile 1140 (...,65,36).
Das Ein- bzw. Ausblenden bestimmter Tasten setzt einen Assembler voraus. Vergleichen Sie hierzu die Zeilen 580...800, 900...910, 1070...1110 sowie 1180...1190 im Assemblerlisting. Die Taste Shift-Rechtspfeil sollte, wie eingangs bereits erwähnt, ausgeblendet bleiben. Für das Verkleinern bzw. Vergrößern des Puffers (voreingestellt sind 64 Zeichen) benötigen Sie ebenfalls einen Assembler. Wählen Sie hierzu in Zeile 2330 von KBREAD/SOU die neue Puffergröße und ändern Sie Zeile 300 entsprechend. Vergessen Sie nicht, in KBREAD/BAS in Zeile 1020 die Einsprungadresse zu korrigieren.

Literatur

[1] Farvour, James: Microsoft Basic decoded & other Mysteries. ISBN 0-936200-01-4.
[2] Heidenreich, Ulrich: VARPTR umgekehrt. mc 1983, Heft 10, S. 53.


```
100 '* * *   K B R E A D  /  D A T    48 KByte Version    * * *
110 '
120 'Sollte Ihnen kein Assembler zur Verfügung stehen, so können
130 'Sie KBREAD/CIM mit Hilfe dieses Programms in den Speicher laden.
140 '
150 'HIMEM muß mindestens 0FED7H = 65239 betragen !
160 '
170 '
500 FOR I = -296 TO -1
510 '
520 '16 KByte:  FOR I =  32472 TO  32767
530 '32 KByte:  FOR I = -16688 TO -16385
540 '
550 READ D : POKE I,D
560 NEXT I
570 END
580 '
590 '
1000 DATA 245,197,213,229,205,127,10,69,33,192,255,229,62,14
1010 DATA 205,51,0,72,205,73,0,254,10,32,2,62,123,254,25,48,243
1020 DATA 254,31,48,239,254,1,48,235,254,32,48,25,254,13,202
1030 DATA 106,255,17,234,254,213,254,8,40,39,254,24,48,30,254,9
1040 DATA 48,51,254,10,192,209,119,120,183,40,201,126,35,205,51
1050 DATA 0,205,162,255,5,24,198,205,55,255,43,126,35,120,185
1060 DATA 32,246,201,120,185,200,43,126,254,10,35,200,43,62,0
1070 DATA 205,51,0,205,162,255,4,201,205,72,3,230,7,47,60,198
1080 DATA 8,95,120,183,200,62,32,119,35,213,205,51,0,209,5,29
1090 DATA 204,162,255,200,24,236,55,245,62,13,119,205,51,0,205
1100 DATA 162,255,62,15,205,51,0,121,144,71,241,225,33,191,255
1110 DATA 112,205,141,255,225,209,193,241,205,154,10,201,33,188
1120 DATA 255,205,13,38,213,58,191,255,10,17,192,255,227,35,115
1130 DATA 35,114,225,201,245,197,14,4,6,120,62,1,211,255,16,254
1140 DATA 6,120,62,2,211,255,16,254,13,32,237,193,241,201,65,36
1150 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1160 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0
```

Bild 3. Wer keinen Assembler hat, kann das Programm von Bild 1 mit dieser Laderoutine abspeichern

```
100 '* * *   K B R E A D  /  B A S    48 KByte Version    * * *
110 '
120 'Das Programm KBREAD/BAS demonstriert die Zusammenarbeit
130 'zwischen BASIC und dem Maschinenprogramm KBREAD/CIM.
140 '(KBREAD/CIM ist die assemblierte Version von KBREAD/SOU).
150 '
160 'HIMEM muß mindestens 0FED7H = 65239 betragen !
170 '
180 '
190 '
1000 DEFINT A-Z
1010 CLEAR 500
1020 DEFUSR0 = -296     ' = 0FED8H bzw. POKE 16526,216 : POKE 16527,254
1030 '
1040 '16 KByte:  DEFUSR0 =  32472 bzw. POKE 16256,216 : POKE 16527,126
1050 '32 KByte:  DEFUSR0 = -16688 bzw. POKE 16256,216 : POKE 16527,190
1060 '
1070 '
1080 CLS
1090 PRINT$140,"E i n g a b e   A d r e ß s a t z"
1100 '
1110 X= 0 : Y= 5 : S$="Vorname:" : GOSUB 8000    'Aufbau der Maske
1120 X=36 : Y= 5 : S$="Name:"    : GOSUB 8000    '  *    *    *
1130 X= 0 : Y= 8 : S$="Straße:"  : GOSUB 8000    '  *    *    *
1140 X= 0 : Y=10 : S$="Plz:"     : GOSUB 8000    '  *    *    *
1150 X=15 : Y=10 : S$="Ort:"     : GOSUB 8000    'Aufbau der Maske
1160 '
1170 '
2000 'Eingabe der Daten
2010 '
2020 X= 9 : Y= 5 : L=20 : GOSUB 10000 : V0$=A$
2030 X=42 : Y= 5 : L=20 : GOSUB 10000 : NA$=A$
2040 X= 8 : Y= 8 : L=30 : GOSUB 10000 : ST$=A$
2050 X= 5 : Y=10 : L= 4 : GOSUB 10000 : PL$=A$
2060 X=20 : Y=10 : L=32 : GOSUB 10000 : O$=A$
2070 X= 0 : Y=14 : S$="Eingaben Korrekt (J/N) ?" : GOSUB 8000
2080 X=25 : Y=14 : L=1 : GOSUB 10000 : AN$=A$
2090 '
2100 IF AN$ <> "J" AND AN$ <> "N" THEN 2080
2110 IF AN$="N" THEN 1080
2120 '
2130 '
3000 CLS
3010 PRINT$129,"Eingegebene Daten:"
3020 PRINT
3030 PRINT"Vorname: ";V0$
3040 PRINT"Name   : ";NA$
3050 PRINT"Straße : ";ST$
3060 PRINT"Plz    : ";PL$
3070 PRINT"Ort    : ";O$
3080 END
3090 '
3100 '
8000 'Berechne Bildschirmposition und drucke Feldnamen
8010 PO=X+64*Y
8020 PRINT $PO,S$
8030 RETURN
8040 '
8050 '
10000 'Hole Daten mit Hilfe von KBREAD/CIM
10010 PO=X+64*Y          'Bildschirmposition des Eingabefelds
10020 PRINT $PO,"";      'Positioniere auf Eingabefeld
10030 X=USR0(L)          'Hole String mit maximaler Länge L
10040 A$=A$              'Fixiere den Wert von A$
10050 RETURN             'Zurück in's Hauptprogramm
```

Bild 2. So kann der neue Tastaturtreiber in ein Basic-Programm eingebunden werden

„Der Kopf ist jener Teil unseres Körpers, der uns am häufigsten im Wege steht."

Gabriel Laub,
tschechischer Satiriker
(geboren 1928)

```
                00010 ;*  *  *  *     K B R E A D  /  S O U     *  *  *  *
                00020 ;*                                                    *
                00030 ;*                                                    *
                00040 ;* Autor: H. Bulling                 Datum: 22.01.84   *
                00050 ;*                                                    *
                00060 ;* Das Programm KBREAD/SOU liest, nachdem es alle     *
                00070 ;* Register auf den Stack gerettet hat, einen max. 64 *
                00080 ;* Zeichen langen String ein und schreibt ihn in den  *
                00090 ;* Speicherbereich ab 0FFC0H. Die Lange des Strings   *
                00100 ;* befindet sich an Speicherstelle 0FFBFH.            *
                00110 ;* Das Unterprogramm GETSTR schreibt diesen String in *
                00120 ;* die Variable A$.                                   *
                00130 ;*                                                    *
                00140 ;* Ausgeblendete Tasten:                              *
                00150 ;*                          1. SHIFT -->  (ASC 16H)   *
                00160 ;*                          2. CLEAR      (ASC 1FH)   *
                00170 ;*                          3. BREAK      (ASC 01H)   *
                00180 ;*                                                    *
                00190 ;* Geanderte Tasten:                                  *
                00200 ;*                          1. CTRL (ASC 0AH) in "&"  *
                00210 ;*                   \                                *
                00220 ;* Die Lange des Puffers (kleiner 48H) wird vom        *
                00230 ;* BASIC ubernommen.                                  *
                00240 ;*                                                    *
                00250 ;*                                                    *
                00260 ;**********************************************************
                00270 ;
                00280 ;
                00290 ;
FED0            00300           ORG     0FED0H
                00310 ;
                00320 ; 16 KByte:  ORG 07ED0H
                00330 ; 32 KByte:  ORG 0BED0H
                00340 ;
FED0 F5         00350           PUSH    AF              ;Rette Register
FED1 C5         00360           PUSH    BC              ;  "     "
FED2 D5         00370           PUSH    DE              ;  "     "
FED3 E5         00380           PUSH    HL              ;Rette Register
FED4 CD7F0A     00390           CALL    0A7FH           ;BASIC-Parameter nach HL
FED7 45         00400           LD      B,L             ;Länge des Puffers nach B
FED8 21C0FF     00410           LD      HL,STRBUF       ;Pufferadresse nach HL
                00420 ;
                00430 ; Hier beginnt das Level II Rom, Adresse 05D9H
                00440 ;
FEE3 E5         00450 KEYBD     PUSH    HL              ;HL zeigt auf den Puffer
FEE4 3E0E       00460           LD      A,0EH           ;Kontrollzeichen "Cursor an"
FEE6 CD3300     00470           CALL    0033H           ;Cursor an
FEE9 48         00480           LD      C,B             ;C = Lange des Puffers
FEEA CD4900     00490 L05E0H    CALL    0049H           ;warte auf Eingabe
                00500 ;
                00510 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                00520 ;
                00530 ; Zusätzliche Zeilen. Andert CTRL (ASC 0AH) zu "&" (ASC 7BH)
                00540 ; und blendet CLEAR (ASC 1FH), BREAK (ASC 01H) und SHIFT Rchts-
                00550 ; pfeil (ASC 19H) aus.
                00560 ;
                00570 ;
FEED FE0A       00580           CP      0AH             ;Teste auf CONTROL
FEEF 2802       00590           JR      NZ,CONT         ;Sprung, falls ungleich CONTROL
FEF1 3E7B       00600           LD      A,7BH           ;andere CONTROL zu "&"
FEF3 FE19       00610 CONT      CP      19H             ;Teste auf SHIFT Rechtspfeil
FEF5 28F3       00620           JR      Z,L05E0H        ;Blende Taste aus
FEF7 FE1F       00630           CP      1FH             ;Teste auf CLEAR
FEF9 28EF       00640           JR      Z,L05E0H        ;Blende Taste aus
FEFB FE01       00650           CP      01H             ;Teste auf BREAK
FEFD 28EB       00660           JR      Z,L05E0H        ;Blende Taste aus
                00670 ;
                00680 ;
                00690 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Bild 1. Dieses Programm ersetzt den Tastaturtreiber, der im ROM von Adresse 5D9H bis Adresse 0673H gespeichert ist

```
                00700 ;
FEFF FE20       00710           CP      20H             ;Teste auf Blank
FF01 3019       00720           JR      NC,L060CH       ;Weder Blank noch Kontrollzeichen
FF03 FE0D       00730           CP      0DH             ;Teste auf ENTER
FF05 CA6AFF     00740           JP      Z,L0662H        ;Sprung, falls ENTER gedrückt
                00750 ;
                00760 ;         CP      1FH             ;Teste auf CLEAR
                00770 ;         JR      Z,L0619H        ;CLEAR ausgeblendet
                00780 ;         CP      01H             ;Teste auf BREAK
                00790 ;         JR      Z,L0661H        ;BREAK ausgeblendet
                00800 ;
FF08 11EAFE     00810           LD      DE,L05E0H       ;Lade Rucksprungadresse
FF0B D5         00820           PUSH    DE              ;Rucksprungadresse auf Stack
FF0C FE08       00830           CP      08H             ;Teste auf SHIFT Linkspfeil
FF0E 2027       00840           JR      Z,L0630H        ;Sprung, falls SHIFT Linkspfeil
FF10 FE18       00850           CP      18H             ;Teste auf Linkspfeil
FF12 281E       00860           JR      Z,L062BH        ;Sprung, falls Linkspfeil
FF14 FE09       00870           CP      09H             ;Teste auf Rechtspfeil
FF16 2033       00880           JR      Z,L0646H        ;Sprung, falls Rechtspfeil
                00890 ;
                00900 ;         CP      19H             ;Teste auf SHIFT Rechtspfeil
                00910 ;         JR      Z,L0641H        ;SHIFT Rechtspfeil ausgeblendet
                00920 ;
FF18 FE0A       00930           CP      0AH             ;Teste auf CONTROL
FF1A C0         00940           RET     NZ              ;Zurück zu L05E0H, falls kein CONTROL
FF1B D1         00950           POP     DE              ;Hole Rucksprungadresse vom Stack
FF1C 77         00960 L060CH    LD      (HL),A          ;Druckbares Zeichen empfangen, sichern
FF1D 78         00970           LD      A,B             ;248 - Anzahl empfangener Zeichen
FF1E B7         00980           OR      A               ;Setze Status
FF1F 28C9       00990           JR      Z,L05E0H        ;Ignorieren, falls Puffer voll (außer CR)
FF21 7E         01000           LD      A,(HL)          ;Hole letzten Buchstaben
FF22 23         01010           INC     HL              ;Ernohe Pufferadresse um 1
FF23 CD3300     01020           CALL    0033H           ;Drucke letzten Buchstaben
FF26 CDA2FF     01030           CALL    SOUND           ;BEEP
FF29 05         01040           DEC     B               ;
FF2A 18BE       01050           JR      L05E0H          ;Hole nächstes Zeichen
                01060 ;
                01070 ;L0619H CALL    01C9H           ;Die nächsten 5 Zeilen werden
                01080 ;       LD      B,C             ;von CLEAR benötigt. Diese Taste
                01090 ;       POP     HL              ;ist ausgeblendet.
                01100 ;       PUSH    HL              ;
                01110 ;       JP      L05E0H          ;
                01120 ;
FF2C CD37FF     01130 L0622H    CALL    L0638H          ;Warte auf nächste Taste
FF2F 2B         01140           DEC     HL              ;Geh zum letzten Zeichen (1 vor ENTER)
FF30 7E         01150           LD      A,(HL)          ;Hol es und teste auf CONTROL
FF31 23         01160           INC     HL              ;Korrigiere Pufferadresse
                01170 ;
                01180 ;         CP      0AH             ;Teste auf CONTROL
                01190 ;         RET     Z               ;CONTROL ausgeblendet
                01200 ;
FF32 78         01210 L062BH    LD      A,B             ;Puffer voll?
FF33 B9         01220           CP      C               ;Erhaltene Zeichen - Pufferlange
FF34 20F6       01230           JR      NZ,L0622H       ;Sprung, falls Platz für neue Daten
FF36 C9         01240           RET                     ;Zurück, Puffer voll
FF37 78         01250 L0638H    LD      A,B             ;B=erhaltene Zeichen, C=Pufferlänge
FF38 B9         01260           CP      C               ;Puffer voll?
FF39 C8         01270           RET     Z               ;Ja, Rucksprung
FF3A 2B         01280           DEC     HL              ;Zeiger auf vorhergehendes Zeichen
FF3B 7E         01290           LD      A,(HL)          ;Hole es
FF3C FE0A       01300           CP      0AH             ;Teste auf CONTROL
FF3E 23         01310           INC     HL              ;Zeiger auf zuletzt erhaltenes Zeichen
FF3F C8         01320           RET     Z               ;Rucksprung, falls CONTROL
FF40 2B         01330           DEC     HL              ;überschreibe letztes Zeichen
FF41 3E08       01340           LD      A,08H           ;Backspace Kontrollzeichen
FF43 CD3300     01350           CALL    0033H           ;Drucke Backspace
FF46 CDA2FF     01360           CALL    SOUND           ;BEEP
FF49 04         01370           INC     B               ;Korrigiere Zeichenzähler
FF4A C9         01380           RET                     ;Rucksprung
                01390 ;
                01400 ;L0641H LD      A,17H           ;Benutzt von SHIFT Rechtspfeil
                01410 ;       JP      0033H           ;Taste ausgeblendet
                01420 ;
```

```
FF4B CD4BB3     01430 L0646H    CALL    0B34BH          ;Wafle auf nächste Taste
FF4E E607       01440           AND     07H             ;Isoliere die unteren 3 Bits
FF50 2F         01450           CPL                     ;Invertiere Akku
FF51 3C         01460           INC     A               ;Gibt 1 <= X <= 8
FF52 C608       01470           ADD     A,08H           ;Losche die oberen Bits des Zählers
FF54 5F         01480           LD      E,A             ;Sichere Zahl der zu addierenden Blanks
FF55 78         01490 L0650H    LD      A,B             ;Hole Große des verbleibenden Platzes
FF56 B7         01500           OR      A               ;Puffer voll?
FF57 C8         01510           RET     Z               ;Ja, Rucksprung
FF58 3E20       01520           LD      A,20H           ;Lade Blank
FF5A 77         01530           LD      (HL),A          ;Space in den Puffer
FF5B 23         01540           INC     HL              ;Zeige auf nachstes Pufferzeichen
FF5C D5         01550           PUSH    DE              ;Sichere DE-Register des Hauptprogramms
FF5D CD3300     01560           CALL    0033H           ;Drucke Blank
FF60 D1         01570           POP     DE              ;Hole DE-Register des Hauptprogramms
FF61 05         01580           DEC     B               ;Verringere Zeichenzähler um 1
FF62 1D         01590           DEC     E               ;Genugend Blanks gedruckt?
FF63 CCA2FF     01600           CALL    Z,SOUND         ;BEEP
FF66 C8         01610           RET     Z               ;Ja, Rucksprung
FF67 18EC       01620           JR      L0650H          ;Nein, weiter bis genügend gedruckt
FF69 37         01630 L0661H    SCF                     ;CARRY-Flag gesetzt, falls BREAK gedruckt
FF6A F5         01640 L0662H    PUSH    AF              ;ENTER gedruckt
FF6B 3E0D       01650           LD      A,0DH           ;Lade CR
FF6D 77         01660           LD      (HL),A          ;Stelle CR in den Puffer
FF6E CD3300     01670           CALL    0033H           ;Drucke CR
FF71 CDA2FF     01680           CALL    SOUND           ;BEEP
FF74 3E0F       01690           LD      A,0FH           ;Lade Cursor-aus Kode
FF76 CD3300     01700           CALL    0033H           ;Cursor aus
FF79 79         01710           LD      A,C             ;C = Pufferlange
FF7A 90         01720           SUB     B               ;Pufferlange - erhaltene Zeichen
FF7B 47         01730           LD      B,A             ;Hole Anzahl der Zeichen in Puffer
FF7C F1         01740           POP     AF              ;Rekonstruiere AF-Register
FF7D E1         01750           POP     HL              ;HL = Startadresse des Puffers
                01760 ;         RET                     ;Original war Unterprogramm
                01770 ;
                01780 ;Hier endet das Level II Rom, Adresse 0673H
                01790 ;
FF7E 21BFFF     01800           LD      HL,LENBUF       ;Hole Pufferadresse
FF81 70         01810           LD      (HL),B          ;Empfangene Zeichen (ohne CR)
FF82 CD8DFF     01820           CALL    GETSTR          ;Stelle Puffer nach A$
FF85 E1         01830           POP     HL              ;Rekonstruiere Register
FF86 D1         01840           POP     DE              ;  "     "
FF87 C1         01850           POP     BC              ;  "     "
FF88 F1         01860           POP     AF              ;Rekonstruiere Register
FF89 CD9A0A     01870           CALL    0A9AH           ;Stelle HL in BASIC-Variable
FF8C C9         01880           RET                     ;Zurück in's BASIC
                01890 ;
                01900 ; Unterprogramm GETSTR            vgl. NC 10/83
                01910 ;
268D            01920 CREVAR    EQU     268DH           ;Suche Variablenadresse
FF8D 218CFF     01930 GETSTR    LD      HL,NAME         ;Adresse des gew. Variablennamens
FF90 CD8D26     01940           CALL    CREVAR          ;Erzeuge String A$
FF93 D5         01950           PUSH    DE              ;Sichere Variablenadresse
FF94 3ABFFF     01960           LD      A,(LENBUF)      ;Lade Stringlange und
FF97 12         01970           LD      (DE),A          ;schreibe in Variablentabelle
FF98 11C0FF     01980           LD      DE,STRBUF       ;Pufferadresse nach DE
FF9B E3         01990           EX      (SP),HL         ;Stelle Puffer nach A$
FF9C 23         02000           INC     HL              ;  "     "
FF9D 73         02010           LD      (HL),E          ;  "     "
FF9E 23         02020           INC     HL              ;  "     "
FF9F 72         02030           LD      (HL),D          ;Stelle Puffer nach A$
FFA0 E1         02040           POP     HL              ;Rekonstruiere Programmzähler
FFA1 C9         02050           RET                     ;Zurück ins Hauptprogramm
                02060 ;
                02070 ; Unterprogramm SOUND
                02080 ; Dieses Unterprogramm erzeugt auf dem Casettenport (0FFH)
                02090 ; einen kurzen Piepslaut.
                02100 ;
00FF            02110 PORT      EQU     0FFH            ;Casettenport
0004            02120 DUR       EQU     004H            ;Tondauer
0078            02130 FREQ      EQU     078H            ;Tonfrequenz
FFA2 F5         02140 SOUND     PUSH    AF              ;Sichere Register
FFA3 C5         02150           PUSH    BC              ;Sichere Register
```

```
FFA4 0E04       02160           LD      C,DUR           ;Stelle Tonlänge nach C
FFA6 0678       02170 LOOP1     LD      B,FREQ          ;Hole Frequenzwert
FFA8 3E01       02180           LD      A,1             ;1. Ausgabewert
FFAA D3FF       02190           OUT     (PORT),A        ;Wert durch Port ausgeben
FFAC 10FE       02200 LOOP2     DJNZ    LOOP2           ;Warte bis entsprechende Freq. erzeugt
FFAE 0678       02210           LD      B,FREQ          ;Hole Frequenzwert
FFB0 3E02       02220           LD      A,2             ;2. Ausgabewert
FFB2 D3FF       02230           OUT     (PORT),A        ;Wert durch Port ausgeben
FFB4 10FE       02240 LOOP3     DJNZ    LOOP3           ;Warte, bis entsprechende Freq. erzeugt
FFB6 0D         02250           DEC     C               ;Verringere Tonlangenzähler um 1
FFB7 20ED       02260           JR      NZ,LOOP1        ;Zurück, bis Zähler auf 0
FFB9 C1         02270           POP     BC              ;Rekonstruiere Register
FFBA F1         02280           POP     AF              ;Rekonstruiere Register
FFBB C9         02290           RET                     ;Zurück in's Hauptprogramm
FFBC 41         02300 NAME      DEFM    'A$'            ;Name der Zeichenkette
FFBE 00         02310           DEFB    0               ;Ende des Namens
0001            02320 LENBUF    DEFS    0001H           ;Reserviere Platz für Pufferzähler
0040            02330 STRBUF    DEFS    0040H           ;Reserviere Platz für Puffer
0000            02340           END
```

Programmbibliotek

Die meisten von Euch haben die neue Liste über die CLUB-Programme schon beim CLUB-Treffen erhalten. Die nichtanwesenden Mitglieder bekommen die Liste mit dem INFO zugeschickt.

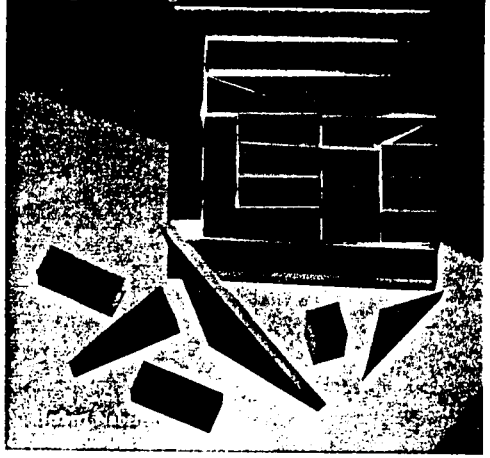
Im nächsten INFO wird dann wieder ausführlich über neue Programme in der Bibliotek informiert.

„Die Fehlersuche wird wesentlich erleichtert, wenn intakte Geräte zur Verfügung stehen."

Aus einer Bundeswehrvorschrift

Strukturierte Programmierung

Hilfsmittel zur professionellen Programmierung



Wenn Programme für den kommerziellen oder wissenschaftlichen Einsatz geplant sind, ergeben sich Aspekte, die eine ausführliche Dokumentation und eine leicht nachvollziehbare Struktur erfordern. Inwieweit sich hierfür die Strukturierte Programmierung — mit wenigen Einschränkungen — anbietet, beleuchtet dieser Beitrag.

Wer sich mit der Programmierung von Computern beschäftigt, hat sicher schon folgendes erlebt. Man hat eine Idee für eine Problemlösung, einige kleine Skizzen bestätigen: es wird funktionieren. Die Idee läßt sich schnell umsetzen, das Programm läuft — und jetzt beginnt die Arbeit erst richtig.

Die kurze, schnelle Lösung kann zwar der Programmschreiber bedienen, zur Vereinfachung sind jedoch in vielen Fällen Ein- und Ausgaben, diverse Konstanten, eventuell sogar Daten im Programm festgelegt. Absicherungen gegen fehlerhafte Bedienung fehlen völlig, Kommentare sind in der Regel sehr kurz gehalten. Eventuell muß das Programm für jeden einzelnen Anwendungsfall neu assembliert oder compiliert werden.

Soll ein anderer dieses Programm benutzen, so muß er sich erst in die Funktion des Programmes einarbeiten, um die Änderungen auszuführen. Damit wird der Kreis der Benutzer auf die Profis begrenzt, die ohnehin in der Lage wären, ein derartiges Programm selbst zu schreiben. Was muß geschehen, damit auch ein unbedarfter Bediener mit dem Programm zurechtkommt? Es muß eine Bedienerführung her, die Ein- und Ausgaben müssen sinnvoll aufgebaut werden. Die Abspeicherung auf die Diskette muß so geschaltet sein, daß nicht das gesamte System abstürzt, wenn ein Fehler gemacht wird.

Sind alle Arbeiten ausgeführt, so hat die ursprüngliche Lösung nur noch einen Anteil von wenigen Prozent des gesamten Programms.

Nebensachen?

Typisch für die unstrukturierte Arbeitsweise ist, daß der Programmschreiber sich durchaus Gedanken über das Programm gemacht hat; nur hat er sich dabei auf die Lösung eines Teilproblems (vermutlich auf 'den' Lösungsalgorithmus) beschränkt. Die unwichtigen 'Nebensächlichkeiten' wurden bestenfalls mit einem flüchtigen Gedanken bedacht! Dabei entsteht meistens ein Programm, das die Zusammenarbeit mit anderen Funktionen nicht berücksichtigt. An die damit vorgegebene Struktur müssen sich alle späteren Erweiterungen halten. Prinzipiell kann man zwar auch den schon geschrieben, wenn sich dies als notwendig erweist. Die Praxis zeigt jedoch, daß man nur äußerst ungern bereits getestete Programmteile wieder umschreibt, wobei oftmals neue Fehlerquellen entstehen. Diese Vorgehensweise führt dann zu den (heute fast zur Regel gehörenden) Terminüberschreitungen bei Software-Projekten. Hinzu kommt noch, daß bei vielen praktischen Projekten am Anfang nicht einmal eine exakte Vorstellung über die Funktion des Systems besteht.

Das soll nicht heißen, daß darüber nicht nachgedacht wurde, sondern daß unter dem Gesichtspunkt der Programmerstellung und der technischen Realisierungsmöglichkeiten durchaus wesentliche Aspekte einfach nicht erkannt beziehungsweise übersehen wurden. Schon Kleinigkeiten können erhebliche Probleme verursachen, wenn sie an den technischen Grenzen der betreffenden Systeme Änderungen bewirken. Dies gilt insbesondere für Echtzeitanforderungen!

Ein Rechner, dessen Arbeitsgeschwindigkeit gerade ausreicht, um einen Analog-/Digitalumsetzer mit maximaler Datenrate zu betreiben, wird nicht ohne wesentliche Verringerung der Abtastrate in der Lage sein, zusätzlich eine Tastatur auf eine bestimmte Taste hin abzufragen. Wenn der Hardware-Entwurf die Möglichkeit des Interrupts durch diese Taste nicht berücksichtigt, so ist entweder eine Hardware-Änderung nötig oder eine Verringerung der Abtastrate muß in Kauf genommen werden. Gleichzeitig steigt der Entwurfsaufwand in diesem Fall wesentlich an, da der gesamte Programmteil zeitlich zu optimieren ist.

Allen geschilderten Problemen ist eines gemeinsam: Vor Beginn der Arbeiten wurden nicht alle Probleme des Hard- und Software-Designs ausreichend durchdacht. Infolgedessen schaffen die schon realisierten Teilbereiche Randbedingungen, die in allen späteren Arbeiten berücksichtigt werden müssen. Bei der Systemplanung nicht berücksichtigte Eigenheiten können den Arbeitsaufwand eines Projektes äußerst negativ beeinflussen, falls sie beispielsweise eine Änderung an fertiggestellten Programmteilen oder an der Hardware erzen ist auch der zusätzliche Testaufwand, der zur Prüfung des Zusammenspiels zwischen vorhandenen und neuen Routinen notwendig ist.

Wurde für das Design zum Beispiel eine höhere Sprache verwendet, so kann eine Systemerweiterung unter Umständen erfordern, größere Teile des Programms in Assembler zu schreiben. Dies bringt aber einen erheblichen Zeitaufwand zur Lösung von Problemen mit sich, die schon einmal unter anderen Gesichtspunkten gelöst worden waren.

Ähnliche Probleme treten im laufenden Einsatz von Systemen und Programmen auf. Im Laufe der Lebensdauer müssen Anpassungen an neue Peripheriesysteme (wie Plotter, Drucker) erfolgen, Teile der Lösung sind oftmals zu ersetzen, weil sich Anforderungen geändert haben (zum Beispiel gesetzliche Vorschriften, die Verarbeitung anderer Materialien). Im Laufe der Zeit sind einzelne Bauelemente nicht mehr zu beschaffen, und der Ersatz (zum Beispiel AD-Umsetzer) erfordert eine neue Anpassung. Aus wirtschaftlicher Sicht ist daher die einmalige Erstellung eines Programmes nicht der einzige Faktor, der bei der Programmierung entscheidend ist.

Es müssen die Kosten für die gesamte Lebensdauer eines Programms beziehungsweise Systems berücksichtigt werden. Ein geringfügiger Mehraufwand entsteht durch die Erstellung ausführlicher Unterlagen und einem prüfungsfreundlichen Gesamtdesign. Das zahlt sich in der Regel aber über die gesamte Einsatzzeit eines Programmes aus. Wenn Programme das Wohl und Wehe von Menschen beeinflussen können, sogenannte sicherheitsrelevante Software (medizinische Technik, Maschinensteuerung), ist eine ausführliche, unmittelbar verständliche Dokumentation sowieso zwingende Voraussetzung. Sie haben sonst bei der gutachterlichen Prüfung keine Chance, zu bestehen. Bei größeren Projekten (die schon in Mannjahren gemessen werden) kann man auf eine gründliche Vorplanung ohnehin nicht verzichten, so daß hier durch die Anwendung der Strukturierten Programmierung kein zusätzlicher Aufwand entsteht.

Ausnahmen

Es sei allerdings nicht verschwiegen, daß es Gründe geben kann, ganze Programme oder Teile davon nicht strukturiert auszuführen. Das sind:

Kostenfragen bei Großserien.

Ein Beispiel dafür sind die Homecomputer, Matrixdrucker und Typenradschreibmaschinen. Wenn Serien mit Millionenstückzahlen zum Einsatz kommen, kann der Unterschied zwischen einem oder zwei notwendigen ROMs oder dem nächst größeren EPROM oder ROM schon mehrere Millionen kosten. Alle möglichen Tricks sind erlaubt, wenn nur dadurch das Programm verkürzt wird. Selbst wenn der 'Spaghetti-Code' nur sehr schwer zum Laufen zu bringen ist und damit die Entwicklung entsprechend teuer wird, so ist dies durch die Kosteneinsparung in der Serienproduktion zu rechtfertigen.

Zeitanforderungen.

Muß eine Hardware bis an die Grenzen ihrer Leistungsfähigkeit ausgenutzt werden, so ist jeder Weg, der dieses Ziel erreicht, sinnvoll. Zusätzliche Befehle sind hier überflüssig, wenn sie nur der Prüfbarkeit dienen. In diesem Fall sollte man sich jedoch die Frage stellen, ob nicht ein genereller Designfehler vorliegt. Bei kleineren Serien sollte die Hardware nämlich nie bis an die möglichen Grenzen ausgenutzt werden, da dabei der Programmier-, Test- und Wartungsaufwand beträchtlich steigt. Außerdem wird oftmals die gesamte Programmierkunst aufgewendet, um einen Algorithmus zeitoptimal zu kodieren. Wenn man aber statt dessen den Algorithmus selbst optimierte, könnte auch die Strukturierte Programmierung zu wesentlich schnelleren Ergebnissen führen.

Kurze Testroutinen,

die nur einmal benutzt werden, um einen vermuteten Fehler einzukreisen.

Rationell und wartungsfreundlich

Wie kommt man zu einem rationellen, wartungsfreundlichen Design? Die Antwort ist das 'Strukturierte Programm' oder das 'Strukturierte Design'.

Da kaum eine Programmieraufgabe ohne Berücksichtigung der Hardware auskommt und im industriellen Bereich aus Kostengründen nicht einfach eine Hardware vorgegeben werden kann, hängen in vielen Anwendungsfällen Programmierung und Hardware-Entwicklung voneinander ab. Das Geheimnis der Strukturierten Entwurfsverfahren liegt darin, daß man die einzelnen Komponenten einer technischen Lösung mehrfach auf verschiedenen Ebenen durchdenkt und genaue Vorstellungen über möglichst alle Eigenschaften des gesamten Programms oder Systems entwickelt, bevor(!) die erste Programmzeile geschrieben und der erste Schaltungsentwurf in Angriff genommen wird.

An eine gute Software beziehungsweise an ein gutes Systemdesign sollten die folgenden Forderungen gestellt werden:

Einfache Prüfbarkeit.

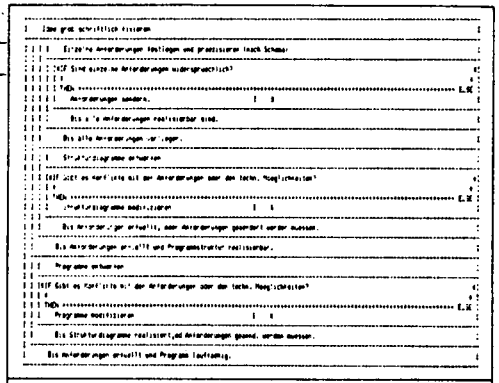
Ideal wäre die Prüfung allein durch die logische Nachverfolgung der Entwurfsunterlagen. Praktisch muß aber gefordert werden, daß der Zugriff auf die einzelnen Routinen und Variablen (bei Systemen auch der Hardware-Baugruppen) einfach möglich ist (z. B. aktivierbare Statusausdrucke).

Gute Dokumentation.

Nur auf diese Weise läßt sich sicherstellen, daß mehrere Personen gleichzeitig an dem Projekt arbeiten können, und später Änderungen auch von anderen Personen durchführbar sind. Eine Dokumentation kann nur dann aktuell sein, wenn sie laufend dem Stand der Arbeiten angepaßt wird. Dies wiederum wird in der Regel nur dann geschehen, wenn die Bearbeiter

Schritt 1: Arbeitsweise des zu entwickelnden Systems grob beschrieben.
Schritt 2: Technische Randbedingungen festlegen (Algorithmen, Ein- und Ausgaben, Speicherplatzbedarf...).
Schritt 3: Festlegung der Arbeitsabläufe (Schnittstellen zwischen den Programmen festlegen, Testmöglichkeiten vorsehen).
Schritt 4: Erstellung und Test des Programmes.

Tabelle 1. Hierarchische Anordnung der Entwurfsschritte



Programm 1. Ablauf eines Programmentwurfes

einen Nutzen in der Erstellung und Aktualisierung der Dokumentation sehen.

Geplante Reaktion auf unvorschriftsmäßige Betriebsfälle.

So sollten falsche Eingaben (zum Beispiel Buchstaben anstelle von Ziffern) erkannt werden, zu einer Fehlermeldung führen und die Wiederholung der Eingabe zulassen. In diesem Sinne gibt es keine Eingaben beziehungsweise Eingangszustände, die 'in der Praxis' nicht vorkommen. Eine Selbstzerstörung des Programms durch falsche Eingaben (zum Beispiel Überschreiben von Eingabepuffern) darf es nicht geben!

Maßnahmen

Mit der Zeichnung von Nassi-Shneidermann-Diagrammen (Strukturdiagramme) allein können die vorgestellten Forderungen nicht erfüllt werden. Auch Strukturdiagramme garantieren aus sich heraus nicht, daß alle wesentlichen Punkte des Designs bedacht werden. Dies gilt besonders für komplexe Projekte, an denen mehrere Personen gleichzeitig arbeiten sollen. Es gilt auch für den Anfänger und den weniger geübten System-Designer bei kleineren Programmen.

Das erste Ziel muß es sein, den Ansatzpunkt für die sinnvolle Erstellung von Strukturdiagrammen zu finden. Es ist zu bedenken, daß wohl kein Design nur nach streng logischen Gesichtspunkten aufgebaut werden kann. Viele Entscheidungen (zum Beispiel Befehlssequenzen zur Steuerung eines zu entwerfenden Systems) sind einzig dem Entwickler und seiner Erfahrung zu überlassen. Nicht in jedem Falle gibt es Kriterien, die eine Bewertung dieser Entscheidungen erlauben. Weitere Freiheitsgrade ergeben sich durch die Auswahl der Hardware. Sie kann eigentlich erst erfolgen, wenn alle Anforderungen definiert sind.

Andererseits ist es unzweckmäßig, die Anforderungen unabhängig von jeder Hardware zu definieren, da dann unter Umständen sehr schnell eine Situation entsteht, in der zur Problemlösung ein Cray-Rechner (sehr schneller und teurer Array-Prozessor) benötigt wird, aber möglichst nur zum Preis eines einzigen Personal Computers. Praktisch muß man während der Festlegung der Anforderungen ständig prüfen, ob die vorgesehene Hardware die Forderungen (unter anderem die Kosten) auch erfüllen kann.

Alle Schritte sollten schriftlich fixiert werden, damit bei notwendigen Änderungen die bereits getroffenen Entscheidungen noch nachvollziehbar sind. Diese Forderung mag zunächst nicht notwendig erscheinen. Bedenkt man jedoch, daß ein Software-Projekt durchaus mehrere Monate oder Jahre dauern kann und daß dabei unter Umständen mehrere Personen mit der Programmierung befaßt sind, so erscheint eine schriftliche Fixierung notwendig. Wer hat nicht schon einmal ein selbsterstelltes älteres Programm ändern müssen und sich dabei über die mangelnde Dokumentation geärgert. Ganz zu schweigen davon, daß die eigene Erinnerung an die Absichten, die bei der Programmerstellung zugrunde gelegen haben, nur noch lückenhaft ist.

Es hat sich gezeigt, daß ein Design vom Problem hin zur Lösung wachsen muß (Tabelle 1),

Problem möglichst genau bekannt sein. Im nächsten Schritt werden die technischen Randbedingungen der Lösung festgelegt (wie Algorithmen, Zahl und Art der Schnittstellen). Der folgende Schritt legt die Arbeitsabläufe fest (Strukturdiagramme) und bereitet damit den letzten Schritt, die Kodierung des Problems vor.

Programm 1 gibt den Ablauf des Designs als vereinfachtes Strukturdiagramm wieder. Die darin enthaltenen Iterationsmöglichkeiten stellen den praktischen Fall dar, bei dem meist Korrekturen an den Ausgangsannahmen notwendig sind.

Dabei sollte man bedenken, daß der Aufwand für eine Korrektur immer größer wird, je später sie erfolgt!

Der systematische Entwurf eines Systems soll diese Iterationen auf das machbare Mindestmaß reduzieren. Ein weiterer Vorteil dieser Vorgehensweise ist eine genaue Systembeschreibung, die es nicht nur dem Entwickler, sondern auch anderen möglich macht, den Sinn eines Programmes oder Systems ohne hohen Arbeitsaufwand zu verstehen.

Wie die Schritte 1 und 2 aussehen können, wird am Schluß dieses Beitrags an einem Beispiel demonstriert.

Anwendungen

Der Schritt 3 bedarf einer weitergehenden Erläuterung. Prinzipiell läßt sich ein Strukturiertes Programm auch durch Flußdiagramme (Kästchendiagramme) dokumentieren. Allerdings verführen diese Diagramme dazu, 'Spaghetti-Code', das heißt einen sehr vermaschten Code zu erzeugen. Außerdem bieten die Kästchendiagramme nur wenig Platz für eine Beschriftung, so daß häufig eine zusätzliche Beschreibung des Flußdiagramms notwendig ist. Übergeordnete Strukturen sind zwar darstellbar, bieten sich durch die Art der Darstellung jedoch nicht ohne weiteres an. Hierbei zu einer sinnvollen Aufteilung zu kommen, ist daher mit zusätzlichem Arbeitsaufwand verbunden.

Die Darstellungsart nach Nassi und Shneidermann geht einen anderen Weg. Ausgehend von der Forderung eines modularen Programmaufbaues und der Forderung nach nur einem Moduleinund nur einem Modulausgang wird eine grafische Darstellung mit nur drei Grundelementen möglich:

— das Modul
— die Schleife
— die Verzweigung

Ein Modul kann dabei ein weiteres Strukturdiagramm repräsentieren, es kann aber auch nur einen einzelnen Vorgang (Anweisung) enthalten.

Eine Schleife wiederholt ein Modul so lange, bis eine vorgegebene Bedingung erfüllt wurde.

Die Verzweigung erlaubt die Auswahl zwischen zwei Submodulen innerhalb eines Moduls und damit die Ausführung von Entscheidungen.

In der Praxis werden einige weitere Konstruktionen verwendet, zum Beispiel mehrfache Fallunterscheidungen (switch, CASE) und abweisende beziehungsweise nichtabweisende Schleifen (siehe auch 'Planvoll programmieren kann jeder' in diesem Heft).

Allein durch die Anordnungsmöglichkeiten schränkt diese Art der Darstellung die Entwurfsfreiheit bei der Programmierung ein. Sie verhindert schon in der Entwurfsphase ein seitliches Einspringen in Programme (mehrfache Einsprungpunkte). Es ist auch nur ein Programmende möglich. Der Assemblerprogrammierer wird hier sicher den Kopf schütteln und auf den erhöhten Programmieraufwand hinweisen, den die Realisierung dieser Struktur auf Assemblerebene erfordert (Programm 2). Gleiches gilt für BASIC- und FORTRAN-Anwender.

Bei höheren Programmiersprachen wie ADA, C oder PASCAL ist diese Art des Programmendes ohnehin selbstverständlich und kann vom Anwender nicht geändert werden.

Aber auch in Assemblerprogrammen gibt es Gründe für den etwas aufwendigeren Code. Schon die einfache Routine im Programm 2a zeigt das Problem sehr deutlich auf. Mit Hilfe eines Debuggers kann in der Routine kein Haltepunkt gesetzt werden, es sei denn, man würde es in Kauf nehmen, bei jedem Durchlauf den Haltepunkt neu zu setzen (zum Beispiel mit DDT). Bei wenigen Durchläufen mag dies noch möglich sein, bei Rechenschleifen wird es aber zur Last.

Will man feststellen, welche Wirkung die Routine gehabt hat, so muß man zuerst den Programmaufruf in der übergeordneten Routine ausfindig machen und nach der Rückkehr aus dem Unterprogramm die Programmausführung stoppen. Meistens wird die Routine von mehreren Stellen aufgerufen, so daß alle in Frage kommenden Routinenaufrufe mit Haltepunkten versehen werden müssen. Das ist jedoch sehr arbeitsintensiv und fehlerträchtig. Einfache Debugger wie DDT lassen ohnehin nur wenige Haltepunkte zu, so daß diese oftmals nur für eine Routine ausreichen.

Eine andere Möglichkeit ist die Untersuchung bestimmter Variablen, die am Ende verschiedener Routinen ausgegeben werden (zum Beispiel die Register des Mikroprozessors). Auch hier entstünde nur ein Zahlenfriedhof, erzeugte man in jedem Durchlauf einen Ausdruck, wie es bei der Routine nach Programm 2a nötig wäre. Auch hier bietet sich die Form nach Programm 2b an, die nur dann einen Ausdruck erzeugt, wenn die Routine beendet ist.

Selbstverständlich kann man auch hier Lösungen finden, die dann aber der jeweiligen Routine anzupassen sind. Ein Unterprogrammaufruf läßt sich einfach durch ein MACRO realisieren, das nach erfolgter Programmprüfung lediglich noch NOPs oder gar keine Befehle mehr enthält. Sonderlösungen müssen einzeln aus dem Programm entfernt werden. Dabei sollte man nicht unterschätzen, daß derartige Änderungsaktionen manchmal auch Code mitentfernen, der noch benötigt wird, und dann beginnt die Fehlersuche erneut.

Der Aufbau eines Strukturierten Programmes beginnt mit der höchsten hierarchischen Ebene, der Hauptprogrammschleife. Diese Schleife soll die wesentlichen Programmblöcke zusammenfassen. Es empfiehlt sich dabei, kein Strukturdiagramm größer als ein DIN- A4-Blatt zu machen, da es sonst unübersichtlich wird.

Weiterhin hat sich gezeigt, daß pro Diagramm maximal sechs Module nebeneinander dargestellt werden sollten. Die einzelnen Module sind dann in getrennten Diagrammen weiter zu präzisieren. Schon diese Darstellung verlangt bei vielen Programmen die Festlegung von Prioritäten, die allerdings nicht immer eindeutig sein müssen. In der weiteren Arbeit wird nun ein Modulblock nach dem anderen als Strukturdiagramm dargestellt, wobei zunächst jeweils nur Blöcke der gleichen Ebene erstellt werden. Diese Arbeit setzt sich fort bis zu einer Ebene, die die einzelnen Aufgaben eindeutig festlegt (zum Beispiel 'Bestimme das Maximum zweier Werte und weise es X zu'). Dabei ist zu beachten, daß in den einzelnen Modulblöcken der untersten Ebene nicht etwa der Programmcode stehen sollte, sondern immer die Beschreibung der Funktion des Codes. Im Falle mathematischer Operation kann das jedoch identisch sein.

```
                RET NG
                RET Z
                . . . .
                . . . .        (CALL PRINT)
                JP SCHLEIFE
EINSPRUNG:      . . . .
                . . . .
SCHLEIFE:       . . . .        Programm-
                . . . .        anweisungen
                . . . .
                JP NC,ENDE
                JP Z,ENDE
                . . . .
                . . . .
                JP SCHLEIFE
ENDE            . . . .        (CALL PRINT)
                RET
```

Programm 2. Beispiel für 'Strukturierten Assemblercode', a) kurzer unstrukturierter Code, b) strukturierte Version

Strukturiertes Design: ein Beispiel

Ein einfaches Anwendungsbeispiel soll die Methode erläutern helfen. Der Gesamtumfang der Überlegungen würde allerdings den gesetzten Rahmen sprengen, so daß wir uns nur auf einen geringen Teil beschränken. Die Zweige sind deshalb lediglich bis zur vorletzten Ebene des Strukturdiagramms dargestellt; die eigentliche Programmierung bleibt unbehandelt. Eine Reihe von Vorgaben, wie etwa die Befehle des verwendeten Systems, können völlig frei gewählt werden. Die jeweilige Auswahl hängt dann aber von den Erfahrungen des Bearbeiters ab. Nicht jeder Designer ist in der Lage, jede dargelegte Entscheidung zu treffen, oder er würde abhängig vom eigenen Wissensstand vielleicht zu anderen Entscheidungen kommen. Es ist ein Merkmal fast aller Software-Entwicklungen, daß es keinen 'einzig richtigen' Weg gibt!

Der erste Schritt ist eine Kurzbeschreibung des Systems (siehe Kasten S. 62). Sie ist zunächst sehr oberflächlich und gibt nur einige Details an. Dieser Stand entspricht in etwa einer Produktidee, die zum Beispiel in einem Gespräch zwischen Entwicklung und Verkauf herauskommen könnte, wenn ein neues Produkt gesucht wird. Tatsächlich beginnen viele Entwicklungsaufgaben mit derartig 'dünnen' Vorgaben, dies gilt besonders dann, wenn der Partner in der Entwicklung aus einer völlig anderen Branche stammt und selbst keine Erfahrungen mit Rechnern hat.

Für den Beginn der Programmierung oder den Entwurf und die Auswahl einer Hardware müssen die vorliegenden Angaben präzisiert werden. Folgendes Schema hat sich beim Verfasser bewährt:

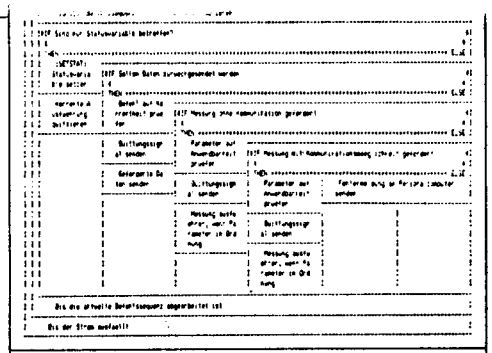
— Festlegen der Systemfunktionen (Reaktion auf zu verarbeitende Daten, Befehlssatz, anzuwendende Rechenverfahren, gesetzliche Vorschriften)
— Darstellung der speziellen Anforderungen (Speicherplatzbedarf, Arbeitsgeschwindigkeit, Interrupt-Reaktionen)
— Beschreibung der zu verarbeitenden Daten und Befehle (Befehlsbeispiel, Wortbreite, Art der Daten, Bedingungen der Befehlsübergabe, mögliche Konflikte bei mehreren Dateneingaben)
— Festlegung der physikalischen Ein- und Ausgaben (analoge Pegel, Datenprotokolle, Geschwindigkeiten, zu erwartende Probleme)
— Bekannte, aber ungelöste Problemstellungen (wie fehlende Mindestanforderungen an die Arbeitsgeschwindigkeit, unbekannte Übergabeprotokolle ...)
— Abschätzung der zu erwartenden Erweiterungen und Lösungsmöglichkeiten (zum Beispiel weitere sinnvolle Algorithmen, weitere Befehle, höherer Datendurchsatz)
— Zur Auswahl stehende Baugruppen der Hard- und Software (Rechnersysteme, Peripheriegeräte, Softwarepakete)

Da die Systemdefinitionen auf dieser Ebene noch sehr änderungsanfällig sind, empfiehlt es sich, den Entwurf auf einem Textverarbeitungssystem durchzuführen. Im wesentlichen entspricht diese Vorgehensweise der Schaffung eines Pflichtenheftes, wobei die Dokumentation sehr viel differenzierter ausfallen muß. Die Ausführung dieser Überlegungen ist im Kasten wiedergegeben, es ist allerdings anzumerken, daß hier der 'Endstand' nicht dargestellt werden kann, da der Umfang zu groß würde.

Diese Vorgaben befassen sich im wesentlichen mit der Funktion des Systems. Technische Details der Realisierung gehören hier nicht hinein, soweit sie nicht als unbedingt verbindlich gelten. Mit Hilfe dieser Vorgaben kann man nun darangehen, das erste Strukturdiagramm zu zeichnen.

Die höchste hierarchische Ebene ist im Programm 3 wiedergegeben. Selbstverständlich erfolgt zunächst eine Initialisierung, deren Einzelheiten noch nicht festgelegt werden müssen. Es folgt die Hauptschleife, die so lange laufen soll, wie das System in Betrieb ist. Daran schließt sich ein Block an, der auf einen Befehlssatz wartet. Dieses Modul bekommt den Namen :WAITBEF:, der später auch im Programm auftreten sollte. Das System wird also von sich aus nichts weiter tun, bis der Personal Computer eine gültige Befehlssequenz abgibt. In der Zuständigkeit dieses Moduls liegt es auch, eventuelle Abbruchbefehle oder Wiederholungsbefehle zu verarbeiten. Erst wenn alle Bedingungen dieses Moduls erledigt sind, wird der Befehlssatz an das nächste Programm übergeben.

Obwohl das erste Modul :WAITBEF: im Moment noch gar nicht geschrieben sein muß, kann man sich schon dem nächsten Modul zuwenden. Dafür muß der Befehlssatz im Detail festgelegt sein. Alle weiteren Aktionen des Systems ergeben sich aus den folgenden Verzweigungsanweisungen, die die weiteren Aufgaben entsprechend der vorliegenden Befehlssequenz (zum Beispiel K1T10N1000) ausführt. Es sind insgesamt fünf Fälle denkbar. Dies sind die reinen Zuweisungen auf Statusvariablen, beispielsweise Vorgaben für die Zahl der Meßwerte, oder die Dauer der Abtastzeit. Weiterhin gibt es Datenanforderungen, das heißt, Daten werden an den steuernden Personal Computer zurückgegeben.



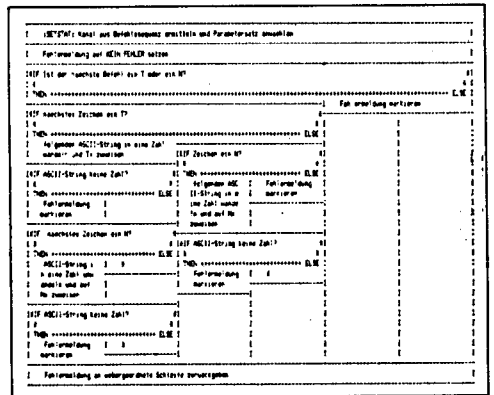
Programm 3. Strukturdiagramm der Hauptprogrammschleife

Bei den Meßaufträgen sind ebenfalls zwei Fälle zu unterscheiden. Die Messung mit und ohne Kommunikationsmöglichkeit. Es fällt in die Zuständigkeit der Module, zu prüfen, ob die entsprechenden Voraussetzungen gegeben sind, und gegebenenfalls eine Fehlermeldung an den Prozessor abzusetzen. Das letzte Modul wird nötig, um falsche Befehlssätze zu verarbeiten beziehungsweise eine entsprechende Fehlermeldung abzusetzen. Auf diese Weise werden die restlichen Module vor formalen Fehlern im Befehlssatz geschützt und müssen nur noch auf Parameterfehler untersuchen.

Ein weiteres Problem ist die Quittierung eines ausführbaren Befehlssatzes, die im Interesse eines automatischen Betriebes notwendig ist. Manche Betriebssysteme können nur eine vordefinierte Zahl von Bytes lesen, und sie warten, bis die entsprechende Zahl von Bytes empfangen worden ist. Das erfordert eine definierte Vorgabe, damit der Anwender sein Steuerprogramm darauf einstellen kann, ohne in die Interna seines Personal Computers einsteigen zu müssen. Da die Meldungen je nach Modul unterschiedlich sind, werden sie von den jeweiligen Modulen selbst erzeugt.

Stufe für Stufe

In der nächsten Arbeitsstufe sind für die bisher definierten Module eigene Strukturdiagramme zu erstellen. Es sollen dabei nur die Strukturdiagramme dieser Hierarchie-Ebene bearbeitet werden. Die Bearbei-



Programm 4. Strukturdiagramm der Statusvariablenzuweisung

Systembeschreibung des Beispielproblems

Kurzbeschreibung:

Es soll ein Datenerfassungssubsystem für Personal Computer entwickelt werden, das die Möglichkeit schafft, analoge Daten aufzunehmen und an den Personal Computer zu überspielen. Das System soll selbständig nach vorgegebenen Zeiten oder Triggerkriterien Daten aufnehmen, zwischenspeichern und auf Abruf an den Personal Computer zur Auswertung senden können. Weiterhin soll das System in der Lage sein, die Datenaufnahme bei einem vorgegebenen Kriterium zu beenden. Die Verbindung zum Personal Computer erfolgt über den IEC-Bus oder eine RS-232-Schnittstelle.

Die Arbeitsweise soll durch den Personal Computer voll programmierbar sein. Befehle dürfen nur ASCII-Zeichen enthalten, da nicht alle Treiber bei Personal Computern die Steuerzeichen voll unterstützen. Die Daten sollen wahlweise als ASCII-Zeichen mit programmierbarem Endekennzeichen (z. B. Ctrl-Z) oder als binärer Datensatz programmierbarer Länge an den Personal Computer übergeben werden.

Es sollen acht Kanäle mit einer Auflösung von 12 Bit abgetastet werden können. Die Abtastrate muß im Bereich von mindestens 100 μs pro Kanal bis 100 s pro Kanal einstellbar sein.

Erweiterungen auf die Linearisierung von Kurven sind denkbar.

Als Anwendungsbereich kommt die Abtastung von mechanischen Vibrationen, die Erfassung von Temperaturen, die Erfassung von Störungen an Stromversorgungsgeräten in Frage.

Festlegen der Systemfunktion:

Das System erhält analoge Daten. Die Daten setzt ein Analog-/Digitalumsetzer in binäre Werte um, die dann entsprechend den vorliegenden Vorgaben weiterverarbeitet oder abgespeichert werden.

Weitere Eingangsdaten sind die Steuersignale vom Personal Computer, die als ASCII-Zeichenkette übergeben werden.

Als Befehl wird ein Datensatz erkannt, der das Befehlskennzeichen benutzt. Alle Daten zwischen Befehlskennzeichen und Befehlsende werden als Befehl akzeptiert und vom System interpretiert. Das Auswertungssystem speichert die Befehlsdaten zwischen, bis das Befehlsende erkannt wird. Erkennt es anstelle des Ende-Kennzeichens das Abbruchkennzeichen, so ignoriert es den gesamten Befehlssatz. Mit Hilfe eines Wiederholungskennzeichens läßt sich das System auffordern, den kompletten Datensatz an den Personal Computer zurückzusenden. Erst wenn ein kompletter Datensatz vorliegt, wird der Datensatz interpretiert.

Als Rechenoperationen sind vorgesehen: Vergleich von Meßwert und Vorgabewert, Mittelwertbildung, Minimum- und Maximumbildung.

Bei der Hardware sind die entsprechenden VDE/FTZ-Bestimmungen einzuhalten.

Spezielle Anforderungen:

Für den vorgesehenen Anwendungsbereich erscheinen 10 000 Abtastungen pro Sekunde ausreichend zu sein. Ein großer Teil der Anwendungen wird mit wesentlich niedrigeren Abtastraten auskommen. Im Bereich der niedrigeren Abtastraten wäre die gleichzeitige Kommunikation mit dem Personal Computer wünschenswert.

Der Speicherbereich sollte mindestens 20 000 Meßwerte aufnehmen können. Im Bereich der höheren Abtastraten wird während der Messung keine Kommunikation mit dem Personal Computer möglich sein.

Laufende Aktivitäten werden durch jedwede Steuersequenz des Personal Computers unterbrochen.

Die Hardware muß in der Lage sein, die maximale Datenrate des ADU zu verarbeiten.

Zu verarbeitende Daten und Befehle:

Als Eingabedaten sind acht analoge Kanäle vorzusehen. Alle analogen Eingänge werden über einen Multiplexer abgefragt. Multiplexer und Analog-/Digitalumsetzer werden vom Rechner gesteuert, der auch die Daten übernimmt und in seinem Speicher ablegt.

Weiterhin erhält das System Daten vom Personal Computer, die über eine RS-232-Schnittstelle oder den IEC-Bus anstehen. Bei dem Entwurf des Befehlssatzes ist zu beachten, daß nicht alle Personal Computer sämtliche Control-Zeichen ausgeben können. Der Befehlssatz darf daher nur ASCII-Zeichen verwenden.

Eine Eingabe durch den Personal Computer kann die folgende Form haben:

":::35K1T10N1000K2T12N1500K3T200N3000S•••"

Damit würden die Kanäle 1, 2 und 3 mit jeweils einer Abtastzeit von 10, 12 und 200 ms voreingestellt (T), 1000, 1500 und 3000 Messungen angeordnet (N) und die Messung aktiviert (S).

Vor der Ausführung jeder Messung wird auf Wunsch ein Quittungssignal an den Steuerrechner zurückgesendet. Der Personal Computer kann die gemessenen Werte per Befehl abrufen: ":::M1N500•••". Diese Befehlssequenz ruft dann die ersten 500 Meßwerte des Kanals 1 ab.

Inhalt und Blocklänge der übertragenen Daten können per Programm vorgegeben werden. Dies ist notwendig, da manche Rechner mit einer gepufferten Eingabe arbeiten und so lange in der Eingabeschleife warten, bis der Buffer gefüllt ist.

Eine Ausgabe kann die folgende Form haben:
"+ + + K1:00001:1275;1300;..."
Hinter der Kanalangabe folgt einmalig im Datensatz die Adresse des ersten Meßwertes.

Es wird nur ein kompletter Block übergeben; füllen die angeforderten Daten den Block nicht aus, so wird der Rest mit 0 oder einem programmierbaren Zeichen ergänzt.

Festlegung der physikalischen Ein- und Ausgaben:

Das System wird über eine RS-232-Schnittstelle oder den IEC-Bus angesprochen. Die Übertragungsgeschwindigkeit der RS-232-Schnittstelle soll programmierbar sein, das heißt, das System wird durch DIL-Schalter auf die Baudrate des Personal Computers eingestellt. Es müssen Baudraten im Bereich von 300 bis 4800 Baud zur Verfügung stehen. Die unteren Baudraten sind zum Beispiel für Osborne beziehungsweise den C64 nötig.

Auf der Analogseite sollen 8 Eingänge vorhanden sein, die jeweils ±5 V verarbeiten können.

Die Analog-/Digitalwandlerkarte ist über einen Port mit dem Arbeitsprozessor verbunden.

Es ist zu erwarten, daß die gleichzeitige Verarbeitung von Befehlen und analogen Daten zu schwankenden Abtastzeiten führt.

Bekannte, aber ungelöste Probleme:

Nicht festgelegt sind im Moment der volle Befehlsumfang und die tatsächlichen Abtastraten. Offen bleibt auch der volle Umfang der mathematischen Arbeiten, die der Prozessor ausführen soll. Eventuell notwendige Umrechnungen der digitalisierten Daten (Skalierung) müssen auch noch geklärt werden.

Zu erwartende Erweiterungen:

Da das System auch für die Erfassung von Temperaturen eingesetzt werden kann, muß unter Umständen später eine Linearisierung von Kennlinien möglich sein.

Weiterhin ist auch eine direkte Druckerausgabe ohne den Umweg über den Personal Computer denkbar. Dabei sind auch grafische Ausgaben erwünscht (Bit 8 bei EPSON & Co.). Für diesen Zweck muß dann auch eine Bedienung direkt am System möglich sein.

Auch sollte später eine FFT (Fast Fourier Transformation) realisierbar sein.

Hard- und Softwareauswahl:

Es wird auf eine vorhandene Hardware (ADU, Einkartenprozessor, RS-232- bzw. IEC-Bus-Schnittstelle zurückgegriffen.

Die Programme sollen weitgehend in der Sprache C geschrieben werden, ein entsprechender C-Compiler steht zur Verfügung.

tung weiterer Submodul-Ebenen empfiehlt sich zu diesem Zeitpunkt noch nicht.

Als Beispiel sei hier nur das Strukturdiagramm für das Setzen der Statusvariablen vorgestellt.

Programm 4 zeigt die Realisierung der Datensatzdekodierung. Es setzt nun schon eine Reihenfolge der Parameterübergaben voraus, weiterhin wird zugelassen, daß einzelne Befehle (T oder N) nicht unbedingt vorhanden sein müssen. Der Test der numerischen Parameter auf ihre Zulässigkeit kann an dieser Stelle eingefügt oder aber auch wie im Beispiel den Meßprogrammen überlassen werden. Diese könnten eine derartige Untersuchung vornehmen, bevor die Messung beginnt. Werden die Variablen erfolgreich übernommen, so wird ein entsprechendes Quittungssignal gesendet, sonst erfolgt eine Fehlermeldung. Es ist vorteilhafter, erst alle Strukturdiagramme der gleichen Ebene zu erstellen, als von einem Modul aus gleich weiter in größere Tiefen vorzudringen. Dabei gefällte Entscheidungen, die auch andere Module betreffen, können nämlich leicht an anderer Stelle zusätzliche Probleme einhandeln. Liegen hingegen alle Strukturdiagramme einer Ebene vor, so lassen sich diese Entscheidungen mit viel mehr Übersicht treffen.

Es sind in der Regel bei diesem Verfahren etliche Änderungen zu erwarten, bis alle Strukturdiagramme der ersten zwei Stufen vorliegen. Werden diese Änderungen auf der Ebene der Schritte 1, 2 und 3 durchgeführt, so sind normalerweise nur Bleistift, Radiergummi und Texteditor notwendig. Es besteht noch nicht die Notwendigkeit, mühsam erzeugten und getesteten Programmcode zu ändern. Der Änderungsaufwand hält sich in Grenzen, und es wird ein guter Überblick über Möglichkeiten des Systems geschaffen. Eventuelle Erweiterungen können im Ansatz berücksichtigt werden und führen später nicht zu größeren Umschreibeaktionen.

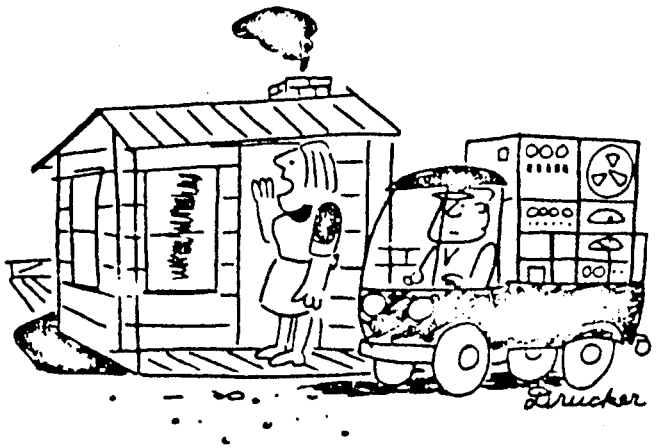
Das vorgestellte Beispiel versuchte, einen Einblick in die Arbeitsweise bei der Erstellung von Strukturierten Programmen zu geben, anhand eines Projektes im Anfangsstadium. Der Leser hat Gelegenheit, dieses Projekt je nach seinen Vorstellungen weiter zu durchdenken oder sich an einem entsprechenden Beispiel zu erproben. Es sei darauf hingewiesen, daß nur die tatsächliche Erprobung des Verfahrens eine sinnvolle Beurteilung erlaubt. Werden die Ergebnisse komplett präsentiert, so kommt man leicht zu der Meinung, daß man dies auf jeden Fall alles auch gewußt hätte — *nachdem* man es gelesen hat.

Kurz und bündig

Strukturiertes Programmieren (oder Strukturiertes Design) ist ein Verfahren zur sinnvollen Erarbeitung eines Programmes (Systems). Ziel des Verfahrens ist es, möglichst viele Entscheidungen schon vor dem Schreiben der ersten Programmzeile kritisch an den Erfordernissen des Gesamtsystems zu prüfen, kritische Punkte im voraus zu erkennen und durch Optimierung des Hardware-/Software-Einsatzes zu entschärfen.

Ein weiterer wesentlicher Punkt ist die Abschätzbarkeit des Aufwandes, die sich wesentlich verbessert, wenn wenigstens die erste und zweite Strukturdiagramm-Ebene vorliegt. Letztendlich wird die Wartung der Programme (Fehlersuche) durch die verbesserte Dokumentation wesentlich vereinfacht und damit langfristig der notwendige Software-Aufwand reduziert. Als Nachteil steht ein höherer Aufwand ins Haus, bevor die erste Programmzeile geschrieben wird. Bezogen auf das gesamte Projekt ergibt sich in der Regel aber ein deutlicher Zeitgewinn, der noch mit dem Projektumfang steigt.

Wer allerdings Spaß daran findet, mit einer Idee im Kopf den Personal Computer einzuschalten und das Programm direkt wachsen zu sehen, der soll es auch weiterhin tun — rationelles Arbeiten und Spaß an der Arbeit müssen nicht unbedingt mit den gleichen Mitteln zu realisieren sein. □

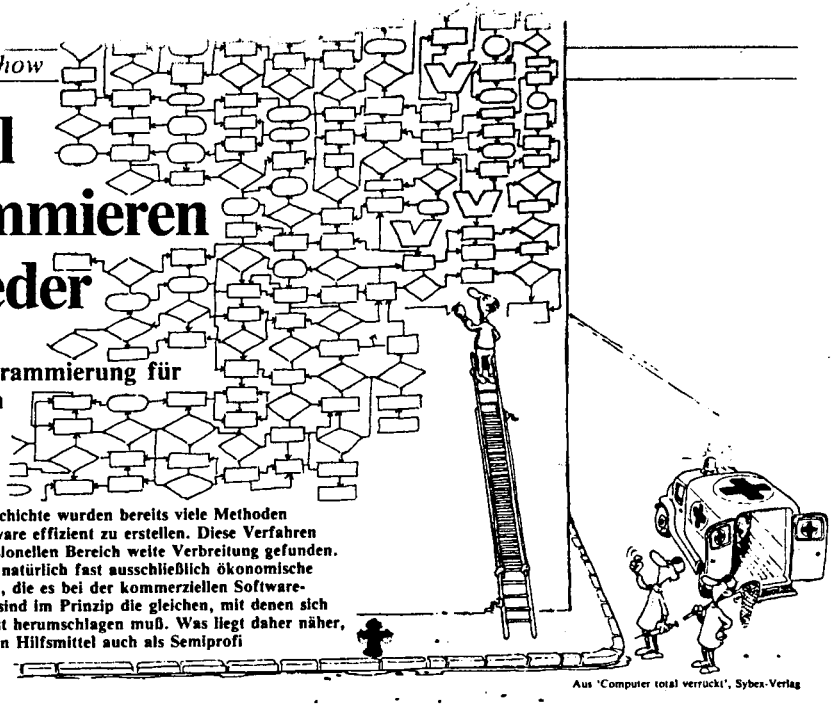


„Charlie! Wo soll der neue Computer hin?"

Planvoll programmieren kann jeder

Strukturierte Programmierung für den Hausgebrauch

Florian Sachse



Aus 'Computer total verrückt', Sybex-Verlag

Im Laufe der Computergeschichte wurden bereits viele Methoden entwickelt, um 'gute' Software effizient zu erstellen. Diese Verfahren haben vor allem im professionellen Bereich weite Verbreitung gefunden. Das hat auf diesem Sektor natürlich fast ausschließlich ökonomische Gründe, aber die Probleme, die es bei der kommerziellen Software-Entwicklung zu lösen gilt, sind im Prinzip die gleichen, mit denen sich auch der Software-Hobbyist herumschlagen muß. Was liegt daher näher, als sich diese professionellen Hilfsmittel auch als Semiprofi zunutze zu machen.

Die Strukturierte Programmierung ist eine Methode (unter anderen) zur Erstellung von Programmen, die

— einfach zu testen sind,

— leicht zu lesen sind und auf dieser Ebene bereits eine qualitative Beurteilung des Programmes zulassen (Beispiel: eine Dokumentation, die nicht ausschließlich dem Programmurheber seine Kreation ins Gedächtnis zurückruft),

— leicht zu warten sind (einfache Fehlersuche),

— die mit geringem Aufwand verändert oder erweitert werden können.

Der wohl am häufigsten beschrittene Weg der Programmentwicklung beim Hobbyisten (vermutlich aber nicht nur bei diesen) ist, ein Programm sofort in den Computer (ohne ein Stückchen Papier) nach den ersten groben Vorstellungen einzutippen. Dann wird es durch 'Hinbiegen' zum Laufen gebracht, und irgendwann werden eventuell noch ein paar Gedanken an eine Dokumentation verschwendet.

Es soll nicht behauptet werden, daß nicht auch auf diese Art und Weise Programme gemäß der aufgeführten Anforderungen erstellt werden können. Erfahrene Programmierer müßten dazu eigentlich in der Lage sein. Es soll hier aber deutlich betont werden, daß es wirklich viel Erfahrung und auch sehr viel Selbstdisziplin erfordert, mit dieser Methode anspruchsvolle — und dabei vor allem umfangreiche — Programme zu erstellen.

Wer zum Beispiel durch die besonderen 'Vorzüge' von BASIC verwöhnt ist (hier noch ein Zeilchen einfügen, dort noch ein GOTO, dann ein Kosmetik-Renumber), wird bei der ersten Verbesserung (und die gibt es eigentlich immer), sagen wir mal nach zwei Monaten des Vergessens, unter Garantie auf die Nase fallen.

Erst denken, dann tippen

Die erste und wichtigste Aufgabe, die es zu lösen gilt, besteht darin, einen vorgegebenen Algorithmus in funktionale Einheiten zu zerlegen. Diese Einheiten nennt man Aufgaben oder Funktionen (nicht zu verwechseln mit dem mathematischen Begriff einer Funktion). Um später auch wirklich zu einem strukturierten Programm zu kommen, darf diese Einteilung natürlich nicht willkürlich vorgenommen werden. Vielmehr versucht man, den Algorithmus erst einmal in grobe Teilaufgaben zu zerlegen. Jede Teilaufgabe kann natürlich ebenfalls in Unteraufgaben zerlegt werden. Von Stufe zu Stufe verlieren die Funktionen an Komplexität. Die 'schrittweise Verfeinerung' wird abgebrochen, wenn die Funktionen so einfach geworden sind, daß sie sich problemlos in eine Programmiersprache umsetzen lassen. Um nicht vom Weg zum strukturierten Programm abzukommen, sollte jede Funktion

— eine logisch abgeschlossene Aufgabe beschreiben (zum Beispiel Einlesen von Eingabedaten, Aufbau einer Bildschirmmaske),

— genau einen Eingang und einen Ausgang haben. Auf Programmebene betrachtet heißt das: Ein Unterprogramm hat nur eine Einsprungadresse, und auch der Rücksprung erfolgt nur an einer Stelle. (Das hat nicht nur formale, sondern auch rein praktische Gründe. Beim Austesten von Maschinenprogrammen bieten sich beispielsweise solche Stellen zum Setzen von Breakpoints an, da sie unter jeder Bedingung angesprungen werden. Außerdem sind häufig vor Verlassen eines Unterprogramms einige abschließende Operationen auszuführen, wie das Wiederherstellen von Registerinhalten.),

— einfach aufgebaut sein. Zu komplexe Funktionen werden durch weitere Unterteilung in Untermodule vereinfacht.

Der Algorithmus kann nun durch die Teilaufgaben beschrieben werden, in die er zerlegt worden ist. Die Funktionen auf der obersten Stufe liefern eine grobe Beschreibung des Algorithmus. Von Stufe zu Stufe wird diese Beschreibung verfeinert, bis hin zu elementaren Anweisungen, die direkt in eine Programmiersprache übertragen werden können.

Bildhaft

Nicht nur für die Programmentwicklung, sondern auch für die Programmwartung ist dieses Konzept von Nutzen. Für den ersten Überblick über ein Programmpaket genügt ein Blick auf die Hauptfunktionen. Sind Änderungen nötig, kann man sich auf die entsprechenden Teilaufgaben konzentrieren. Das setzt natürlich voraus, daß die Funktionen und die Abhängigkeiten zwischen ihnen leicht zu überblicken sind. Als grafisches Hilfsmittel für die erste Planungsstufe haben sich vor allem die Hierarchie-Diagramme bewährt. Jede Funktion wird als Kästchen dargestellt, jede Abhängigkeit durch eine Verbindung. Die Grafik in Bild 1 ist zum Beispiel so zu interpretieren:

Der Algorithmus besteht aus drei Hauptfunktionen M1, M2 und M3. Die Funktion M2.3 setzt sich zusammen aus den Funktionen M2.3.1 und M2.3.2, die selbst nicht weiter unterteilt sind. Aus der Grafik wird auch ersichtlich, daß zum Beispiel M2.2 keine Unterfunktion von M1 ist, da es zwischen beiden keine Verbindung gibt.

Besonders auffallend ist die baumartige Struktur, die sich bei der schrittweisen Verfeinerung (auch Top-Down Entwicklung genannt) von allein ergibt. Werden die Funktionen in einem Programm als Unterroutinen realisiert, dann folgt aus der baumartigen Struktur, daß jede Unterroutine nur aus einer 'darüberliegenden' Routine aufgerufen wird (und das häufig auch nur einmal).

So manchem Programmierer wird bei diesem Gedanken gar nicht wohl zumute sein, aber die Verletzung der baumartigen Struktur kann letztlich zu einem Gebilde führen, wie es Bild 2 zeigt. Das Diagramm sieht vielleicht auf den ersten Blick recht harmlos aus, aber schon die Frage, wie viele Pfade von Funktion A nach Funktion B führen, ist nicht mehr so leicht zu beantworten.

Funktionen, die mehr als einmal benutzt werden, sollten im Hierarchie-Diagramm für jeden Aufruf als eigenständige Funktionen (wenn auch mit gleichem Namen) dargestellt werden (Bild 3). So wird verhindert, daß im Hierarchie-Diagramm die Verbindungen kreuz und quer verlaufen. Im späteren Programm-Code tauchen die Funktionen, die mehrfach benötigt werden, natürlich nur einmal als Unterprogramm auf.

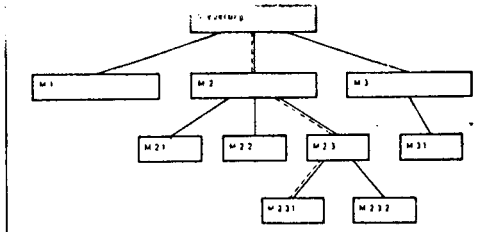


Bild 1. Hierarchie-Diagramme stellen nur die Abhängigkeiten von Programm-Modulen untereinander dar. Sie bilden die planerische Vorstufe zu Flußdiagramm oder Struktogramm.

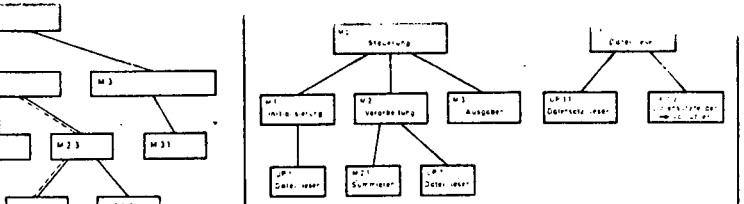


Bild 3. Eine typische Aufteilung einer Programmieraufgabe in logisch unabhängige Funktionseinheiten.

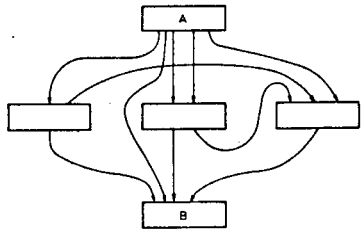


Bild 2. Bei Anwendung der Strukturierten Programmierung gibt es nur genau eine Verbindung zwischen dem aufrufenden und dem aufgerufenen Programm-Modul (wie in Bild 1). Dann können Gebilde wie dieses hier gar nicht vorkommen.

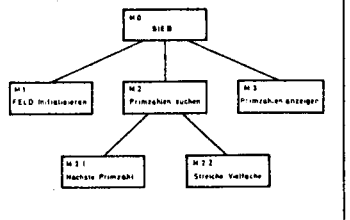


Bild 4. Ein konkretes Beispiel: So läßt sich der Algorithmus zum Ermitteln von Primzahlen (Sieb des Eratosthenes) in Funktionsblöcke aufteilen.

Sieben

Es soll das 'Sieb des Eratosthenes' zur Berechnung aller Primzahlen in einem gegebenen Intervall programmiert werden. Tabelle 1 zeigt die Arbeitsweise des Algorithmus. Auf der obersten Ebene kann der Algorithmus in drei Teilfunktionen zerlegt werden:

1. Feld initialisieren,
2. Primzahlen suchen,
3. Primzahlen ausgeben.

Die Teilfunktionen 1. und 3. können unmittelbar in Programm-Module umgesetzt werden, Funktion 2. sollte aber noch weiter verfeinert werden:

2.1. Nächste Primzahl suchen,

2.2. Alle Vielfachen streichen.

Aus dieser Einteilung ergibt sich ein Hierarchie-Diagramm wie in Bild 4.

So weit, so gut. Der Algorithmus ist in handliche Funktionen zerlegt, und die Abhängigkeiten zwischen ihnen sind auch geklärt, aber über den Kontrollfluß sagt das Hierarchie-Diagramm nichts aus. Zum Beispiel: Werden die Funktionen 2.1 und 2.2 hintereinander ausgeführt (Sequenz), oder entweder 2.1 oder 2.2 (Verzweigung), oder werden sie in einer Schleife so lange wiederholt durchlaufen (Schleife)?

Diese Frage läßt sich bislang nur mit Hilfe der allgemeinen Beschreibung aus Tabelle 1 beantworten: Die Funktionen 1, 2 und 3 bilden eine Sequenz, zuerst die Tabelle initialisieren, dann alle Primzahlen suchen und zuletzt die gefundenen Primzahlen ausgeben. Die Funktionen 2.1 und 2.2 werden in einer Schleife so lange wiederholt, bis das Ende der Tabelle erreicht ist.

Strukturhilfe

Mit den drei Strukturelementen Sequenz, Verzweigung und Schleife kann man nicht nur den Kontrollfluß zwischen den Funktionen beschreiben, sondern auch die Funktionen selbst. Der Kontrollfluß eines

Das Sieb des Eratosthenes beschreibt ein Verfahren, mit dem man alle Primzahlen zwischen 1 und n bestimmen kann. Dabei macht man sich zunutze, daß die ganzzahligen Vielfachen von Primzahlen selber keine Primzahlen sind.

Man geht von einer Zahlenkette mit den ganzen Zahlen zwischen 1 und n aus.

1 2 3 4 5 6 7 8 9 ... n

Da die 1 per Definition keine Primzahl ist, kann sie gestrichen werden. Ab der ersten Zahl der Reihe wird eine noch nicht durchgestrichene Zahl (Primzahl) gesucht ...

~~1~~ 2 3 4 5 6 7 8 9 ... n

... und auch gefunden. Die 2 ist eine Primzahl, deren Vielfachen sind selber keine Primzahlen, sie müssen also gestrichen werden.

1 (2) 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ... n

Die nächste Primzahl, die gefunden wird, ist die 3, deren Vielfachen werden ebenfalls gestrichen.

1 (2) (3) ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ... n

Danach werden die 5, 7, 11 etc. gefunden. Ist die Zahlenreihe vollständig abgearbeitet, so enthält sie nur noch Primzahlen.

Tabelle 1. Der Algorithmus zu 'Sieb des Eratosthenes'

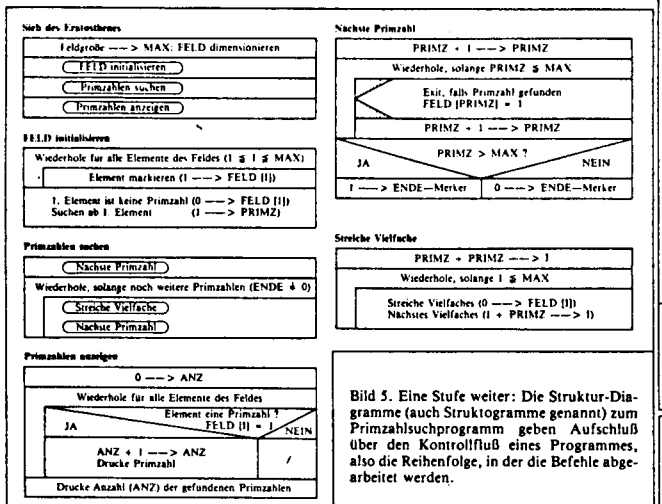


Bild 5. Eine Stufe weiter: Die Struktur-Diagramme (auch Struktogramme genannt) zum Primzahlsuchprogramm geben Aufschluß über den Kontrollfluß eines Programmes, also die Reihenfolge, in der die Befehle abgearbeitet werden.

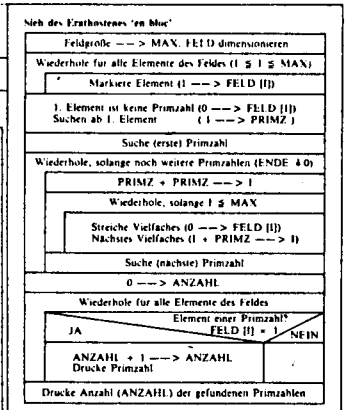


Bild 7. Module wieder einlagern: Man muß beim Strukturierten Programmieren stets einen Mittelweg zwischen Komplexität (zu viele verschiedene Funktionen in einem Modul) und Übersichtlichkeit (zu viele Blätter Papier) wählen.

Bild 6. Verbesserungen oder Änderungen beschränken sich bei modularen Programmen meistens auf sehr wenige Module. Hier eine Alternative zum Modul 'Nächste Primzahl' aus Bild 5.

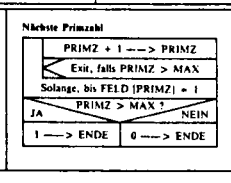
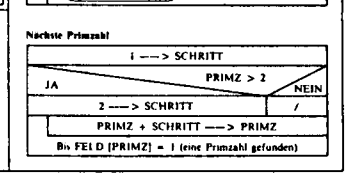


Bild 8. Die Module 'Primzahlen suchen' und 'Nächste Primzahl' lassen sich verbessern. Hier die optimierten Versionen.



Algorithmus läßt sich zum Beispiel mit einem Programmablaufplan beschreiben. Für unsere Zwecke besser geeignet sind aber Struktogramme (auch Nassi-Shneiderman-Diagramme genannt), da sie zu jedem Strukturelement eine grafische Entsprechung haben. Tabelle 2 zeigt die drei Grundkonstrukte und einige Erweiterungen im Vergleich zu Elementen des Programmablaufplans sowie deren Umsetzung in BASIC- und Pascal-Befehle.

Die Strukturblöcke haben — ebenso wie Funktionen — nur einen Ein- und einen Ausgang. Jeder mit einem Großbuchstaben gekennzeichnete Block kann wiederum einen eigenständigen Strukturblock enthalten. Aus diesen Elementen lassen sich die Struktogramme für das Primzahlproblem zusammensetzen (Bild 5). Die Umsetzung des Algorithmus in

```
1000 REMark Hauptprogramm (Steuerung der Module)
1020   MAX = 100
1030   DIM FELD(MAX)
1040   GO SUB 2000
1050   GO SUB 2200
1060   GO SUB 2400
1070 STOP
1080 :
1090 :
2000 REMark MODUL 'Feld initialisieren'
2010   FOR I = 1 TO MAX
2020       FELD(I) = 1
2030   NEXT I
2040   FELD(1) = 0
2050   PRIMZ   = 1
2060 REturn
2070 :
2080 :
2200 REMark MODUL 'Primzahlen suchen'
2210   GO SUB 2600
2220   IF ENDE = 0 THEN GO SUB 2800:GO SUB 2600:GO TO 2220
2230 REturn
2240 :
2250 :
2400 REMark MODUL 'Primzahlen anzeigen'
2410   ANZAHL = 0
2420   FOR I = 1 TO MAX
2430       IF FELD(I) = 1 THEN ANZAHL = ANZAHL + 1:PRINT I
2440   NEXT I
2450   PRINT
2460   PRINT "Zwischen 1 und "+MAX$+" gibt es "+ANZAHL$+" Primzahlen"
2470 REturn
2480 :
2490 :
2600 REMark MODUL 'Naechste Primzahl'
2610   PRIMZ = PRIMZ + 1
2620   IF PRIMZ > MAX THEN GO TO 2660
2630     IF FELD(PRIMZ) = 1 THEN GO TO 2660
2640     PRIMZ = PRIMZ + 1
2650   GO TO 2620
2660   ENDE = 0
2670   IF PRIMZ > MAX THEN ENDE = 1
2680 REturn
2690 :
2700 :
2800 REMark MODUL 'Streiche Vielfache'
2810   I = PRIMZ + PRIMZ
2820   IF I <= MAX THEN FELD(I) = 0:I = I + PRIMZ:GO TO 2820
2830 REturn
```

Programm 1. Auch in BASIC kann man wunderbar strukturiert programmieren. Dieses Programm korrespondiert mit Bild 5.

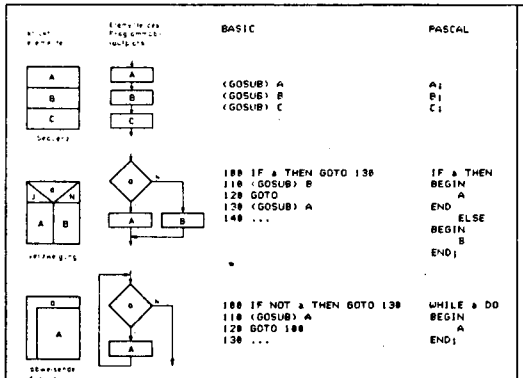
```
2600 REMark MODUL 'Naechste Primzahl'
2610   PRIMZ = PRIMZ + 1
2620     IF PRIMZ > MAX THEN GO TO 2640
2630   IF FELD(PRIMZ) = 1 THEN GO TO 2610
2640   ENDE = 0
2650   IF PRIMZ > MAX THEN ENDE = 1
2660 REturn
```

Programm 2. Die alternative Lösung zum Struktogramm Bild 6.

BASIC (Programm 1) zeigt einige typische Besonderheiten:

Um die Logik der Struktogramme nicht zu verändern, kommt im Modul 'Nächste Primzahl' die Anweisung 'PRIMZ = PRIMZ + 1' zweimal vor (Zeilen 2610 und 2640). Ohne die Logik des Algorithmus zu verletzen, könnte auf eine der beiden Anweisungen verzichtet werden. Das führt aber natürlich zu einer anderen Definition der Funktion 'Nächste Primzahl' (Bild 6, Programm 2).

Die Schleifen in 'Nächste Primzahl' und 'Streiche Vielfache' werden im BASIC-Programm nicht durch eine FOR-Schleife gebildet, obwohl dies eigentlich naheläge. Das Problem besteht



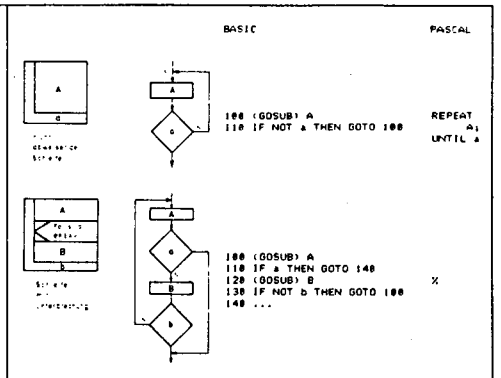
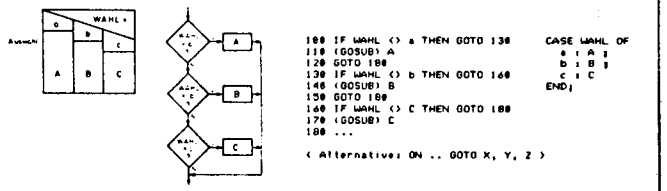


Tabelle 2. Die grafischen Elemente der Strukturierten Programmierung. Parallel dazu die entsprechenden Flußdiagramm-Symbole und eine beispielhafte Realisierung in BASIC und Pascal. Das 'Break'-Symbol (für Schleifenabbrüche) ist eine 'neuere' Schöpfung, auf die man prinzipiell verzichten könnte (daher gibt es auch keine Entsprechung in Pascal). Sie ist aber auf Assembler-Ebene zum Beispiel sehr komfortabel, um schnell auf Sonderbedingungen (Fehler) reagieren zu können. Beachten Sie, daß dieses Symbol keinen Freibrief für das 'Querausteigen' in benachbarte Modulzweige darstellt. Es muß an das Schleifenende gesprungen werden!



darin, daß FOR-Schleifen, je nach Interpreter, abweisende (zum Beispiel Sinclair) oder nichtabweisende Schleifen (zum Beispiel Apple BASIC) bilden. Abweisende und nichtabweisende Schleifen verhalten sich im großen und ganzen gleich: außer wenn schon vor dem ersten Eintritt in die Schleife die Schleifenbedingung nicht erfüllt ist.

In diesem Falle wird die abweisende Schleife überhaupt nicht durchlaufen. Im Gegensatz dazu wird die nichtabweisende Schleife auf jeden Fall einmal durchlaufen, da die Schleifenbedingung erst am Ende des Blocks abgefragt wird. An all jenen Stellen, wo im Programm eine abweisende Schleife unbedingt notwendig ist (um zum Beispiel Feldüberschreitungen zu verhindern), sollten deshalb FOR-Schleifen nur mit Bedacht eingesetzt werden.

In dieser ersten Lösung sind die einzelnen Funktionen, die ja durch das Hierarchie-Diagramm vorgegeben sind, als eigene Struktogramme beziehungsweise Unterprogramme umgesetzt worden. Die Programm-Module müssen aber nicht unbedingt 'physikalisch' (neues Blatt Papier) ausgelagert werden. Wichtig ist nur, daß sie logisch abgeschlossen sind (genau einen Ein-/Ausgang haben). Ohne die Prinzipien der Strukturierten Programmierung zu verletzen, können deshalb die 'Unter-Struktogramme' an den Stellen im Struktogramm eingefügt werden, wo sie sonst aufgerufen würden.

Als Beispiel zeigt Bild 7 die zweite Lösung mit einem zusammengefaßten Struktogramm. Nur das Untermodul zum Suchen der nächsten Primzahl bleibt ausgelagert, da es an zwei verschiedenen Stellen aufgerufen wird.

Interessanter als das 'Einlagern' von Modulen ist natürlich das 'Auslagern'. Ein wesentlicher Vorteil der Strukturierten Programmierung ist es, daß Programmteile, die im Laufe der Entwicklung und Anpassung immer umfangreicher geworden sind, einfach ausgelagert werden können, ohne daß sich die Programmlogik ändert. Der modulare Aufbau läßt sich aber auch auf andere Weise nutzen. Programmteile können ohne große Probleme durch andere oder gleichwertige Module ersetzt werden. Voraussetzung ist natürlich, daß die Datenschnittstellen (Parameterübergabe und Speicherdefinition) übereinstimmen. So arbeitet das Alternativ-Modul aus Bild 6, eingebunden in das Primzahlprogramm, problemlos.

Der Parameterübergabe zwischen Programm-Modulen muß man vor allem auf Assembler-Ebene sehr viel Aufmerksamkeit schenken, da diese oft über Register abgewickelt wird. Wer zum Beispiel nicht von vornherein festlegt, welche Register verändert werden dürfen und welche Register für Übergaben an untergeordnete beziehungsweise übergeordnete Module verwendet werden sollen, der sollte sehr intensiv dokumentieren. Es empfiehlt sich, diese sogenannte Software-Schnittstellen-Beschreibung jedem Modul — sowohl auf Struktogrammebene als auch auf Code-Ebene — in Form von expliziten Kommentaren voranzustellen.

Besser als bei Programmablaufplänen lassen sich in Struktogrammen Programmteile auffinden, die häufig durchlaufen werden (Schleifen) und bei denen Verbesserungen am meisten fruchten. Das Modul kann zu Testzwecken separat untersucht werden, zum Beispiel ob sich vielleicht Teile davon optimieren oder die Anzahl der Schleifendurchläufe verringern lassen.

Da sich die Veränderungen nur auf das Modul beziehen, braucht man um die Logik des gesamten Programms nicht zu bangen, solange die Logik des optimierten Moduls erhalten bleibt. Außerdem läßt sich die Korrektheit eines veränderten Moduls immer noch leichter überprüfen als die eines ganzen (womöglich unstrukturierten) Programms.

Modulweise optimieren

Auch unser Primzahlprogramm läßt sich natürlich verbessern. Schon Eratosthenes war auf die Idee gekommen, daß man nicht das ganze Feld nach Primzahlen durchsuchen muß. Wenn man nämlich alle Primzahlen zwischen 1 und Wurzel(n) gefunden und deren Vielfache markiert hat, kann

die Suche nach weiteren Primzahlen abgebrochen werden. Alle restlichen Zahlen im Feld, die noch nicht markiert sind, sind Primzahlen. Auch das Suchen nach der nächsten Primzahl läßt sich beschleunigen. Alle Primzahlen größer als zwei sind ungerade. Da man nur ungerade 'Kandidaten' als Primzahlverdächtige ins Auge fassen muß, kann man in Zweier-Schritten das Feld durchsuchen und gerade Zahlen überspringen.

Die erste Verbesserung wirkt sich hauptsächlich auf das Modul 'Primzahlen suchen' aus. Dort ändert sich die Schleifenbedingung. Da die Ende-Meldung aus dem Modul 'Nächste Primzahl' nicht mehr benötigt wird, kann dieses Modul sogar vereinfacht werden. Die zweite Änderung bezieht sich ausschließlich auf das Modul 'Nächste Primzahl'. Bild 8 und die Programme 4 und 5 zeigen die optimierten Module.

Aber auch wenn es darum geht, die Korrektheit eines Programms zu überprüfen, bietet die Strukturierte Programmierung Vorteile. Das beginnt schon bei den Struktogrammen, die die Logik des Programms besser veranschaulichen als Programmablaufpläne.

Während in Struktogrammen der Kontrollfluß sehr differenziert dargestellt werden kann, sind die Darstellungsmöglichkeiten in den Programmablaufplänen eher bescheiden. Sowohl die Bedingungen als auch die Schleifen werden durch die Verzweigung realisiert. Das kann natürlich beim Analysieren eines Programmablaufplans zu Interpretationsproblemen führen. Außerdem werden, im Gegensatz zu Struktogrammen, Operationen und den Programmfluß steuernde Anweisungen vermengt.

Es sind ja nur die Anweisungen, also zum Beispiel Zuweisungen oder Rechenoperationen, die zum Programmergebnis führen. Die kontrollierenden Anweisungen steuern 'nur' die Reihenfolge, in der die Anweisungen ausgeführt werden müssen. Dies wird bei Struktogrammen besonders deutlich: Die Blöcke enthalten die Operationen und sind überlagert von Kontrollanweisungen, die die Abarbeitung steuern.

Ein anderes Problem, wenn es um die Analyse eines Programms geht, ist der Unterschied zwischen dem statischen Programmtext und dem dynamischen Verhalten des Programms während der Ausführung. Durch die Einschränkungen der Kontrollstrukturen besteht jedoch ein enger Zusammenhang zwischen der statischen und dynamischen Struktur eines Programms. Anders kann es bei einem nicht strukturierten Programm aussehen, wo wilde Sprünge das dynamische Verhalten in ein 'gordisches Chaos' (siehe Zeichnung am Anfang des Artikels) verwandeln können. Es läßt sich dann auch nicht mehr an Hand des Programmtestes oder Programmablaufplans entwirren.

Bei der Analyse eines strukturierten Programms kann man, wie bei der Programmentwicklung, 'Top-Down' arbeiten. Man geht dabei davon aus, daß die Module auf den logisch tieferen Ebenen korrekt sind und überprüft erst, ob das zu untersuchende Modul diese Untermodule auch wirklich so einsetzt, daß das richtige Ergebnis berechnet wird. Im Fehlerfalle werden die Untermodule, jedes für sich, genauso analysiert.

Diesen Vorteilen der Strukturierten Programmierung, die sich mit wachsender Programmkomplexität noch verstärken, stehen natürlich auch einige Nachteile gegenüber, die hier nicht verschwiegen werden sollen:

Programmiersprachen wie BASIC und Entwicklungsmethoden wie die Programmablaufpläne, die die Strukturierte Programmierung nicht unterstützen, fordern vom Programmierer einiges an Selbstdisziplin, damit er nicht vom 'strukturierten' Weg abkommt. Deshalb ist der umgekehrte Weg, nämlich mit Struktogrammen zu arbeiten, wesentlich sinnvoller. Denn deren Anwendung läßt die Entstehung unstrukturierter Programme nicht zu! (Siehe auch 'Keine Meinung zur Struktur?')

Häufig muß die Modularität eines Programms mit mehr Programm-Code oder mit längeren Programmlaufzeiten erkauft werden. Dies hängt in erster Linie damit zusammen, daß auf Bedingungen nicht immer an den Stellen im Programm reagiert werden darf, wo sie auftreten.

So darf im Beispielprogramm in Bild 5 nicht einfach von Modul 'Nächste Primzahl' aus in das Modul 'Primzahlen anzeigen' verzweigt werden, nachdem das ganze Feld abgearbeitet worden ist. Statt dessen wird die Bedingung in einem 'Merker' festgehalten (hier die Variable ENDE), um dann an passender Stelle abgefragt zu werden (Zeile 2010, Programm 1).

Neuerdings greifen immer mehr Programmierer zur Hochsprache (Pascal, C), also zu Compilern, um maschinennahe Probleme zu lösen. Damit handelt man sich allerdings Laufzeiterhöhungen (auch bei C-Compilern) von Faktor 10 bis 100 (bei sehr kleinen Programmen) im Vergleich zur Assembler-Codierung ein. Daß diese Software-Werkzeuge dennoch immer größeren Anklang finden (Turbo-Pascal zum Beispiel), zeigt, daß die Mikroprozessor-Programmierer der ersten Stunde langsam von einer neuen Generation abgelöst werden.

In einer Zeit, in der Speicher immer größer und ebenso wie die Prozessoren immer schneller werden, kann man sich planvolles Programmieren leisten. Probieren Sie es doch mal. Aller Anfang ist zugegebenermaßen schwer, vor allem, wenn man es vorher anders gelernt hat. Hat man's aber erstmal drauf, kann man über Spaghetti-Code und Trickprogrammierung nur noch milde lächeln. □

```
140 FOR I = 1 TO MAX
150   FELD(I) = 1
160 NEXT I
170 FELD(1) = 0
180 PRIMZ   = 1
190 :
200 REMark MODUL 'Primzahlen suchen'
210 GO SUB 420
220 IF ENDE = 1 THEN GO TO 320
230   :
240   REMark MODUL 'Streiche Vielfache'
250   I = PRIMZ + PRIMZ
260   IF I <= MAX THEN FELD(I) = 0:I = I + PRIMZ:GO TO 260
270   :
280   GO SUB 420
290   :
300 GO TO 220
310 :
320 REMark MODUL 'Primzahlen anzeigen'
330 ANZAHL = 0
340 FOR I = 1 TO MAX
350   IF FELD(I)=1 THEN ANZAHL = ANZAHL + 1:PRINT I
360 NEXT I
370 PRINT
380 PRINT "Zwischen 1 und ";MAX;" gibt es ";ANZAHL;" Primzahlen"
390 :
400 STOP
410 :
420 REMark MODUL 'Naechste Primzahl'
430 PRIMZ = PRIMZ + 1
440 IF PRIMZ > MAX THEN GO TO 480
450   IF FELD(PRIMZ) = 1 THEN GO TO 480
460   PRIMZ = PRIMZ + 1
470 GO TO 440
480 ENDE = 0
490 IF PRIMZ > MAX THEN ENDE = 1
500 RETurn
```

Programm 3

```
2200 REMark MODUL 'Primzahlen suchen'
2210   GRENZE = INT(SQRT(MAX))
2220   IF PRIMZ < GRENZE THEN GO SUB 2600:GO SUB 2800:GO TO 2220
2230 RETurn
```

Programm 4

```
2600   PRIMZ = PRIMZ + 1
2640   IF FELD(PRIMZ)     1 THEN GO TO
2650 RETurn
```

Programm 5

Die Programme 3 bis 5 zeigen die teilweise integrierte Programmlösung nach Bild 7 und 8, ebenfalls wieder in BASIC kodiert.

Literatur:

[1] Arno Schulz
Methoden des Softwareentwurfs und strukturierte Programmierung
de Gruyter 1978
ISBN 3-11-007336-6

[2] Gewald, Haake, Pfadler
Software Engineering
R. Oldenbourg Verlag 1979
ISBN 3-486-21492-6

[3] Gerhard Platz
Methoden der Softwareentwicklung
Carl Hanser Verlag 1983
ISBN 3-446-13040-3

[4] Sneed
Softwareentwicklungsmethoden
Verlagsgesellschaft
Rudolf Müller GmbH 1980
ISBN 3-481-36271-4

[5] Frank Heubach
Strukturierte Programmierung auch bei Mikrocomputern
ELEKTRONIK 1977
Heft 10, S. 113—118

Keine Meinung zur Struktur?

Detlef Grell

Die Strukturierte Programmierung ist eigentlich mehr als nur eine Methode zur Programmentwicklung, fanatische Anhänger sprechen deshalb auch bereits von einer Philosophie. Daher kann man sich über dieses Thema auch unglaublich verbreiten, und Kenner der Materie werden (trotz zweier Beiträge in diesem Heft) sicherlich Aspekte vermissen, die sie für besonders wichtig oder gar ausschlaggebend für Anwendung (oder auch Nichtanwendung) der Strukturierten Programmierung halten. Anregungen und Ergänzungen seitens der Leser sind uns daher sehr willkommen.

Lassen Sie mich an dieser Stelle einigen grundlegenden Mißverständnissen über das, was 'Strukturierte Programmierung' sei, zu Leibe rücken. Leicht hysterische Struktur-Fans vermuten ja bereits Rufmordkampagnen, so geballt schlägt ihnen gelegentlich die Ablehnung der Strukturierten Programmierung vor allem aus Kreisen der 'Maschinen-Code-Quetscher' und 'Hacker' entgegen. Läßt man sich dann eine Kurzdefinition dessen, was für Strukturierte Programmierung gehalten wird, geben, stößt man auf faszinierendes Halbwissen. In diesem Sinne also:

Was ist überhaupt Strukturierte Programmierung? Oder andersherum gefragt: 'Kann es überhaupt ein unstrukturiertes Programm, also ein Programm ohne irgendwie geartete Struktur geben?' Selbstverständlich hat jedes Programm eine Struktur, und wenn wir in diesem Heft von Strukturierter Programmierung reden, dann meinen wir die Programmentwicklungstechnik, die die Herren Nassi und Shneidermann sich darunter vorstellen. Der Begriff 'Strukturierte Programmierung' ist der Name für ein ganz konkret definiertes Verfahren (deswegen schreiben wir auch das Adjektiv groß), und man sollte Wendungen wie 'das Programm hat eine gute oder schlechte Struktur' vermeiden, weil sie ganz einfach falsch sind.

Im Sinne der Strukturierten Programmierung kann es nämlich nur Programme geben, die sich an die Vorschriften dieses Verfahrens halten (dann sind sie 'strukturiert') oder aber nicht — und dann sind sie eben 'nicht strukturiert'.

Im Grunde genommen besteht der wesentliche Teil der Strukturierten Programmierung aus der regelgerechten Anwendung der Struktursymbole von Nassi-Shneidermann. Und schon geht das Jammern los: 'Das sind doch diese blöden, unverständlichen Kästen'. Nein, liebe Leser, da muß ich ernsthaft widersprechen:

Mit diesen Symbolen (die keinen Deut komplizierter sind als die gemeinhin gepriesenen Flußdiagramme), ist den Herren Nassi und Shneidermann nämlich ein echter Geniestreich gelungen. Zunächst haben sie nachgewiesen, daß sich mit nur drei Darstellungsformen (Sequenz, Schleife, Verzweigung) jedes programmtechnische Problem lösen läßt.

Aber viel wichtiger ist die Konsequenz, die sich aus den von Nassi-Shneidermann geschaffenen Symbolen ergibt:

Bei der Anwendung dieser Struktursymbole ist es definitiv unmöglich, ein nichtstrukturiertes Programm (im Sinne der Erfinder) zu erstellen. Man kann das natürlich bei Kenntnis der zugrundeliegenden Regeln auch auf andere Weise erreichen. Das ist aber weitaus schwieriger, für einen Programmieranfänger eigentlich sogar unmöglich. Deswegen kann man die Einarbeitung in den Umgang mit Struktursymbolen nur wärmstens empfehlen, denn das hält Sie immer auf dem richtigen Weg.

Oft hört man die Behauptung, man könne nur in ganz bestimmten Programmiersprachen strukturiert programmieren, vornehmlich in Pascal. BASIC sei völlig ungeeignet, und immer wieder geistert das Schlagwort vom angeblichen 'GOTO-Verbot' in der Strukturierten Programmierung umher.

Erstens: In absolut jeder Programmiersprache kann man strukturiert programmieren. Das läßt sich einfach 'beweisen': In jeder Programmiersprache kann man Sequenzen, Verzweigungen und Schleifen (wie bequem oder umständlich auch immer) programmieren. Um das zu zeigen, sind die Beispiele im vorangegangenen Beitrag auch extra in BASIC geschrieben. Es gibt auf der anderen Seite natürlich Programmiersprachen, die die Strukturierte Programmierung wesentlich vereinfachen beziehungsweise gar nichts anderes zulassen. Zu den sehr förderlichen Sprachen gehört ganz sicher Pascal, denn ALGOL, Pascals Ursprung, wurde im Prinzip für die Strukturierte Programmierung entwickelt.

Zweitens: Direkte Sprünge, also vor allem das umstrittene GOTO, benötigt man in Pascal eigentlich gar nicht. Man kann im Prinzip mit den verfügbaren Kontrollstrukturen auf Struktogramm-Ebene programmieren. Ein Pascal-Lehrer kann es sich somit erlauben, ein 'GOTO-Verbot' auszusprechen, um seine Schützlinge vor einem Abgleiten auf die 'schiefe Bahn' zu bewahren.

Und so unrecht hat er nicht, denn der Mißbrauch der direkten Sprünge ist des Pudels Kern: Modulwechsel quer und rückwärts machen natürlich viel Freude. Kann man doch Programme damit so schön unlesbar und sich als Programmentwickler schnell unentbehrlich machen.

Dennoch, der Gebrauch der direkten Sprünge ist auf Assembler-Ebene natürlich unverzichtbar, wie sollte man sonst Schleifen oder Verzweigungen kodieren? Wenn man sich aber beim Einsatz von Sprüngen darauf beschränkt, nur die in sich abgeschlossenen Konstrukte der Strukturierten Programmierung nachzubilden, ist das völlig legitim.

Kommen wir zum letzten Punkt, bei dem sich die Gemüter am erbittertsten erhitzen:

Strukturierte Programme sind viel zu lang, umständlich und vor allem langsam. Das sind die Lieblingsargumente der codequetschenden High-Speed-Low-Memory-Fetischisten, wenn es darum geht, die Einführung der Strukturierten Programmierung abzuwimmeln.

Die Zeiten 'kurzer Programme mit allen Mitteln' sind vorbei. Ob ein Programm ein oder zwei Kilobyte lang ist, ist heutzutage weitgehend irrelevant, außerdem wird man beispielsweise auf Assembler-Ebene kaum mehr als 20 Prozent Diskrepanz erhalten.

Die Frage, wann ein Programm elegant und wann es umständlich ist, läßt sich wohl nicht objektiv beantworten. Es bleibe jedem selbst überlassen, ob er sich lieber durch hundert verschlungene Befehle durchbeißt oder lieber hundertzwanzig klar gegliederte Befehle auf Anhieb nachvollzieht.

Strukturierte Programme werden geringfügig langsamer und länger, wenn man weitgehend gleiche Algorithmen und Kodierungen zugrunde legt. Es kommen üblicherweise einige verschachtelte Unterprogrammaufrufe hinzu, bei denen hauptsächlich die Parameterübergaben und gelegentlich mehrfaches Testen desselben Flags Zeit kosten. Zum Teil verliert man auch Zeit, weil bestimmte Befehle nicht verwendet werden dürfen.

Hier sollte sich der Programmierer gefordert sehen, durch eine geeignete Programmanlage und die Verwendung optimierter Algorithmen die Strukturierte Lösung zu beschleunigen. Es gilt also, 'bewußtseinsverändernd' zu wirken. Sicher, planvolles, nachvollziehbares Programmieren (die Strukturierte Programmierung ist ja nicht das einzige Verfahren dazu) kostet auch Schweiß fernab von der Tastatur. (Manche Programmierer bekommen richtiggehend Terminal-Entzugs-Ausschlag, wenn man ihnen mit solchen praxisfernen Vorstellungen kommt.) Ich neige immer mehr zu der Unterstellung, daß sich oft psychologische Barrieren (ich vermeide bewußt das Wort 'Bequemlichkeit') hinter den Argumenten der Struktur-Gegner verbergen.

Vielleicht hilft ein bißchen Motivation? Raffiniertes Trickprogrammieren ist out. Selbstmodifizierender Code ist ätzend. Sprünge per RETurn über einen gefälschten Stack sind einfach widerlich! Oder etwa nicht? □

Programmieren heißt mit Dateien arbeiten

Basic kennt in seiner Grundausstattung bei der Dateiverwaltung eigentlich nur den sequentiellen oder den wahlfreien Zugriff. Alle anderen Verfahren muß man selber stricken oder als Zusatzprogramm kaufen.

Das englische Wort File wird laut Wörterbuch mit Akte, Ordner oder Ablage übersetzt. Der Computer-Begriff File ist damit etwas umfassender als das deutsche Wort Datei. Es gibt unterschiedliche Typen von Files, wie zum Beispiel auch Ordner und Schnellhefter sich unterscheiden. Nicht der Inhalt bestimmt den Filetyp auf einer Diskette, sondern die Art, wie die Daten abgelegt sind.

In einem Büro kann die Ablage ein Karton sein, in den alle Schreiben, so wie sie kommen, einfach hineingestapelt werden. Es dürfen aber auch Aktenschränke mit Schubladen und Ordnern sein, alles mit viel System. Es ist einzusehen, daß die erste Art der Ablage sehr billig ist und wenig Raum kostet, die zweite hingegen schon kräftig ins Geld geht. Ganz anders sieht es aber bei der Zugriffszeit aus. Methode eins zwingt, den Papierstapel von Anfang bis Ende zu durchsuchen, jedenfalls so weit, bis das gewünschte Schreiben gefunden ist. Bei Methode zwei genügen wenige Handgriffe. In der EDV nennt man die beiden Verfahren »Sequentiell« und »Random«.

Sequentielle Files sind typisch für endlose Medien wie Lochstreifen oder Magnetbänder, sie kommen aber auch auf Disketten vor.

Sinnvoll sind sie sicherlich bei Programmen, denn ein solches wird immer im ganzen aufgezeichnet beziehungsweise geladen. Und damit hätten wir ein Kriterium für die Anwendung von sequentiellen Files auf Disketten: Immer dann, wenn alle Daten im Stück in den RAM zu laden sind (und natürlich da auch hineinpassen). Nun ist es leider nicht so, daß man sagen kann: »Bei großen Files wähle ich also Random«.

Ein Random-File stellt man sich am besten wie ein Buch vor. Für jeden Eintrag (Record genannt) ist eine Seite reserviert. Der Zugriff erfolgt über die Seiten-(Record-)Nummer, wobei das System auf Anhieb die richtige Seite aufschlägt. Die Seitengröße gilt für das ganze Buch. Da aber die »Schriftgröße« konstant ist, müssen Sie die Seiten so groß halten, daß der längste Eintrag auf eine Seite paßt. Der Nachteil: Alle anderen Seiten sind mehr oder weniger leer. Random-Files kosten viel Platz. Außerdem entbindet die Random-Organisation den Programmierer nicht von einer unangenehmen Aufgabe. Er muß wissen, unter welcher Record-Nummer was abgelegt ist, beziehungsweise das System der Ablage im Programm definieren und einige Suchhilfen zur Verfügung stellen.

Keine echten Random-Files

Eine Diskette ist in konzentrische Spuren (Tracks) unterteilt und diese wiederum in Sektoren. Im Directory (Inhaltsverzeichnis) wird notiert, welche Tracks und Sektoren jeweils zu welchem File-Namen gehören. Meistens werden (um die Anzahl der Notizen kleiner zu halten) mehrere Sektoren zu einem Block zusammengefaßt. Diese sind der Reihe nach, beginnend auf dem ersten Track von 1 bis n durchnumeriert. Folglich führt die Blocknummer, dividiert durch die Anzahl Blöcke je Track, sehr einfach auf die Track-Nummer.

Im Directory werden zu jedem File die Block-Nummern notiert und zwar in der logischen Folge. Diese kann infolge von mehrmaligem Aufzeichnen, Löschen und Vergrößern bestehender Files alles andere als seriell sein. Möglich ist zum Beispiel:

— File 1 belegt die Blöcke 3, 5, 7
— File 2 belegt die Blöcke 1, 2, 4, 8

So gesehen wird also ein File auf der Diskette immer sequentiell aufgezeichnet, in einem Block nach dem nächsten, so dieser gerade frei ist. Der »große« Unterschied zu Random ist gar nicht so groß. Die Blöcke werden rein rechnerisch in Records eingeteilt, deren Größe können Sie wählen, meistens jedoch mit der Einschränkung ≤256 Bytes (der physikalischen Größe eines Sektors). Wenn Sie nun einen Record über seine Nummer ansprechen, dann fängt das System an zu rechnen: Record-Nummer mal Record-Länge dividiert durch Blockgröße ergibt die relative Block-Nummer. Von da aus läßt sich dann mit »einem Blick ins Directory« die absolute Block-Nummer ermitteln, über die Blockgröße kommt man so zum Sektor und von da über die Record-Größe auf die gesuchten Daten.

Es gibt noch einen zweiten parallelen Weg. Die Bytes eines Files werden von 0 bis n (n = File-Länge-1) durchnumeriert. Dazu wird beim Öffnen des Files der sogenannte Filepointer, auch RB (Relatives Byte) genannt auf Null gesetzt. Bei sequentiellen Files wird dieser Filepointer beim Lesen/Schreiben eines jeden Bytes um 1 erhöht. Bei Random-Files wird lediglich aus Record-Nummer mal Record-Länge das relative Byte (der Filepointer-Wert) errechnet. Das, dividiert durch die Blockgröße, ergibt wieder die Block-Nummer und weiter geht's, wie schon geschildert. Die Sache mit dem Filepointer hat einige Vorteile, zum Beispiel ist damit der Zugriff auf jedes Byte eines Files möglich, aber erst einmal zur Frage »echtes Random?«

Der Zugriff auf einen Record eines Random-Files läuft, wie schon angedeutet, über einige Umwege.

Der Zugriff beginnt beim Directory, von dort geht's zum sogenannten File-Entry oder über einige Sektoren mit der sogenannten Track/Sektor-Liste zum eigentlichen Sektor, der den Record hält. Dazwischen läuft noch einiges an Mathematik und Zugriff auf Tabellen. Bei einer echten Random-Organisation werden die Adressen aller Records auf einer Spur notiert, die sich das System natürlich merkt, wenn der File geöffnet wird. Das heißt nach spätestens einer Umdrehung der Disk ist die Adresse eines Records bekannt.

Dieser Komfort ist auf Personal Computern nicht üblich. Dort ist ein Random-File nichts weiter als ein speziell verwalteter sequentieller File. Diesen Nachteil werden wir noch zu unserem Vorteil nutzen, zuerst aber noch etwas Technik.

Der OPEN-Befehl ist aber leider unvermeidbar gefährlich. Ihm sollte möglichst bald ein CLOSE folgen. Schauen wir uns an, warum. Obwohl von der Diskette virtuell (scheinbar) immer nur ein Byte nach dem anderen gelesen wird, sieht es praktisch anders aus. Der FDC (Floppy Disk Controller) kennt nur den Befehl »ganzen Sektor lesen«. Dazu muß ihm das DOS (Disk Operating System) sagen, wo im RAM die zum Beispiel 256 Bytes abzulegen sind. Im Falle Schreiben will der FDC wissen, ab welcher RAM-Adresse die Daten zu finden sind. Diesen RAM-Bereich nennt man Sektor-Puffer. Tatsächlich arbeitet das DOS immer nur auf diesen Puffer. Ist er voll, wird er im Stück auf die Disk kopiert beziehungsweise im Falle »Lesen« woandershin (im RAM) übertragen. Doch der Sektor-Puffer allein reicht nicht aus. Zusätzlich wird je File ein zweiter Puffer angelegt, in dem zum File gehörige Directory-Informationen gehalten werden. Beide Puffer zusammen heißen DOS-Buffer.

Für jeden offenen File benötigt man einen DOS-Puffer, meist zwei und mehr, zum Beispiel wenn man einen File lesen und gleich auf einen anderen schreiben (kopieren) will. Der OPEN-Befehl hat im wesentlichen die Aufgabe, zu einem File-Namen einen solchen DOS-Puffer einzurichten.

In manchen Systemen wird dafür RAM reserviert, für wieviel Puffer kann der User sogar wählen. Dort belegt dann jedes aktive OPEN einen dieser Puffer.

Wird nun während einer Session etwas geändert (File wird länger), dann wird das nur im File-Puffer notiert, nicht auf der Disk. Erst ganz zum Schluß wird das Directory und die Block-Belegungstabelle aktualisiert, das bewirkt der Befehl CLOSE. Und noch eines: Der Sektor-Puffer wird immer erst auf die Disk geschrieben, wenn er voll ist. Da aber die Datenmenge höchst selten ein ganzzahliges Vielfaches von 256 ist, bleibt der Sektor-Puffer zum Schluß mehr oder weniger leer. Hier sorgt CLOSE für dreierlei.

Es wird eine EOF-Marke (End of File) nach dem letzten User-Byte in den Puffer geschrieben.

Es wird der ganze Puffer auf die Disk kopiert.

Es wird der Puffer wieder freigegeben (für das nächste OPEN).

Warum ist nun OPEN gefährlich? Stellen Sie sich vor, zwischen OPEN und CLOSE tritt eine Störung auf, wie zum Beispiel Netzausfall, eine Spannungsspitze, Fehlbedienung oder der simple Druck auf die BREAK- oder RESET-Taste! Dann heißt das, die letzten Daten sind nicht auf der Disk, dem File fehlt das EOF-Byte und das Directory stimmt nicht mit der Wirklichkeit überein. Die meisten Betriebssysteme geben sich dann gar keine Mühe zu retten, was noch zu retten ist, die melden schlicht Error und weigern sich, den File weiter zu bearbeiten.

Daraus folgt Regel 1: Einem OPEN sollte so schnell wie möglich ein CLOSE folgen.

Regel 2: Updating nie am Original. Das heißt wenn Sie einen File ändern, zum Beispiel erweitern wollen, legen Sie zuerst eine Kopie an.

Dazu muß der File nur gelesen werden, im Modus »Read only« ist aber ein File gegen die genannten Pannen recht unempfindlich.

Darum:
— File im Read-Mode öffnen
— auf einen im Write-Mode geöffneten File kopieren
— Read-File schließen
— den zu »appenden« Teil schreiben

Apple II / TRS-80/Genie

Schreiben von sequentielle Files auf die Disk

```
10 D$=CHR$(4)                                  
20 PRINT D$;"OPEN TESTFILE"      20 OPEN "O",1,"TESTFILE"
30 PRINT D$;"WRITE TESTFILE"                   
40 PRINT"einige Daten"           40 PRINT #1,"Einige Daten"
50 PRINT D$;CLOSE TESTFILE"      50 CLOSE 1
```

Lesen von der Disk

```
10 D$=CHR$(4)                                  
20 PRINT D$;"OPEN TESTFILE"      20 OPEN "I",1,"TESTFILE"
30 PRINT D$;"READ TESTFILE"                    
40 INPUT A$                      40 INPUT #1,A$
50 PRINT D$;CLOSE TESTFILE"      50 CLOSE 1
```

Bild 1. Lesen und Schreiben von sequentiellen Files

Apple II / TRS-80/Genie

Schreiben von Random-Files auf die Disk
(hier in den 3. Record)

```
10 D$=CHR$(4)                                  
20 PRINT D$;"OPEN TESTFILE L12"  20 OPEN "D",1,"TESTFILE"
30 PRINT D$;"WRITE TESTFILE R3"  30 FIELD 1, 12 AS A$
                                 35 LSET A$="Einige Daten"
40 PRINT"einige Daten"           40 PUT 1,3
50 PRINT D$;CLOSE TESTFILE"      50 CLOSE 1
```

Lesen von der Disk

```
10 D$=CHR$(4)                                  
20 PRINT D$;"OPEN TESTFILE L12"  20 OPEN "R",1,"TESTFILE"
30 PRINT D$;"READ TESTFILE R3"   30 FIELD 1, 12 AS A$
40 Input A$                      40 GET 1,3
50 PRINT D$;CLOSE TESTFILE"      50 CLOSE 1
```

Bild 2. Bei Random-Files sammelt der Apple Pluspunkte

```
10 D$=CHR$(13)+CHR$(4)                    0123456789
20 F$="TESTFILE"
30 PRINT D$"OPEN"F$;D$"DELETE"F$;         123456789
   D$"CLOSE"F$
40 PRINT D$"OPEN"F$;D$"WRITE"F$           23456789
50 PRINT "0123456789"
60 PRINT D$"CLOSE"F$                      3456789
70 REM
80 REM --------------------               456789
90 REM
100 PRINT D$"OPEN"F$                      56789
110 FOR X=0 TO 9
120 PRINT D$"READ"F$",B",X                6789
130 INPUT A$:PRINT A$
140 NEXT                                  789
150 PRINT D$"CLOSE"F$                     89
160 END
                                          9
```

Bild 3. So kann man beim Apple den Filepointer manipulieren

— alles testen
— Original löschen, neuen File auf Namen des Originals umbenennen.

In der Befehls-Syntax unterscheidet man zwei Gruppen. Dabei geht es unter anderem darum, wie man zwischen den DOS-Puffern, mit denen das System arbeitet und den File-Namen, die der Benutzer kennt, eine Verbindung herstellt. In den meisten Basic-Dialekten sind die DOS-Puffer von 1 bis n numeriert. Im OPEN-Befehl wird eine dieser Nummern dem File-Namen zugeordnet, zum Beispiel als »OPEN "0",1,"Daten"«. Hier wurde der File »Daten« geöffnet, der Transfer läuft über den Puffer Nummer 1, "0" heißt, es ist ein Output-File. Beim Apple II kennt der Anwender die Puffer-Nummer nicht, und das geht so:

```
10 D$=CHR$(4)
20 PRINT D$"OPEN Daten"
30 PRINTD$"WRITE Daten"
```

Zeile 10 bringt das Steuerzeichen »es folgt DOS-Befehl« in die Variable D$. In Zeile 20 die Anweisung, den File »Daten« zu öffnen, Zeile 30 schließlich legt die Richtung fest, hier Schreiben (auf Disk).

Jetzt wirken alle PRINT-Befehle anstatt auf den Schirm auf die Disk, auch LIST schreibt jetzt auf die Diskette. Ein CLOSE-Befehl hebt diese Wirkung wieder auf.

In der anderen Lösung muß man beim PRINT-Befehl die Buffer-Nummer mit angeben, zum Beispiel als »PRINT#1,"text"«.

In Bild 1 sind beide Verfahren gegenübergestellt, und zwar am Beispiel einfacher serieller Files. Wie Sie sehen, ist die Apple-Lösung recht umständlich. Ihr Vorteil liegt im einfachen Routing, so nennt man das Umleiten von Bildschirmausgaben auf andere Geräte, beziehungsweise das Holen von Daten von der Disk anstatt von der Tastatur.

In Bild 2 werden die Lösungen für Random-Files gegenübergestellt. Hier zeigt der Apple deutliche Vorteile, weil das Fielding und Converting (nicht gezeigt) entfallen.

ISAM heißt Index Sequentiel Access Method und ist tatsächlich die Kombination der Vorteile beider Standard-Verfahren, sozusagen ein Random-File mit variabler Record-Länge und noch einigen Vorteilen. Sie erinnern sich an den Filepointer? Dieser wird beim Lesen oder Schreiben jeweils um 1 Byte weitergeschaltet. Es gibt aber auch einen Befehl, den Filepointer auf ein bestimmtes Byte im File zu stellen. Liest man dann ab da, wird der Text bis zum nächsten Schlußzeichen erfaßt.

```
 1 PRINT "GEBEN SIE 3 TEXTE EIN":PRINT
 2 FOR I=1 TO 3
 3 PRINT I;". TEXT ";:INPUT A$(I)
 4 NEXT
10 D$=CHR$(13)+CHR$(4)
20 F$="TESTFILE"
30 PRINT D$"OPEN"F$;D$"DELETE"F$;D$"CLOSE"F$
35 PRINT D$"OPEN INDEX";D$"DELETE INDEX";D$"
   CLOSE INDEX"
40 PRINT D$"OPEN"F$;D$"WRITE"F$
50 FOR I=1 TO 3
60 PRINT A$(I)
62 A(I)=A(I-1)+LEN(A$(I))+1
65 NEXT
66 PRINT D$"CLOSE"F$
67 PRINT D$;"OPEN INDEX";D$"WRITE INDEX"
68 FOR I=0 TO 2:PRINT A(I):NEXT
69 PRINT D$"CLOSE INDEX"
70 REM
80 REM ----------------------
90 REM
91 CLEAR
92 D$=CHR$(13)+CHR$(4):F$="TESTFILE"
95 PRINT D$"OPEN INDEX";D$"READ INDEX"
96 FOR I=0 TO 2:INPUT A(I):NEXT
97 PRINT D$"CLOSE INDEX"
100 PRINT D$"OPEN"F$
110 FOR I=0 TO 2
120 PRINT D$"READ"F$",B",A(I)
130 INPUT A$:PRINT A$
140 NEXT
150 PRINT D$"CLOSE"F$
160 END
```

Bild 4. Ein ISAM-File mit drei Sätzen und einem Schlüssel

```
 2 FOR I=1 TO
 3 PRINT I;". TEXT ";:INPUT A$(I)
 4 NEXT
20 F$="TESTFILE:0"
30 OPEN "O",1,F$,"MI"
35 OPEN "O",2,"INDEX:0"
50 FOR I=1 TO 3
60 PUT 1,,,A$(I)
62 A(I)=LOC(1)!
65 NEXT
66 CLOSE 1
68 FOR I=1 TO 3:PRINT #2,A(I):NEXT
69 CLOSE 2
70 REM
80 REM ----------------------
90 REM
91 CLEAR
92 F$="TESTFILE:0"
95 OPEN"I",1,"INDEX"
96 FOR I=1 TO 3:INPUT #1,A(I):NEXT
97 CLOSE 1
100 OPEN "I",1,F$,"MI"
110 FOR I=1 TO 3
120 RBA=A(I-1)
130 GET 1,!RBA,,A$:PRINT A$
140 NEXT
150 CLOSE 1
160 END
```

Bild 5. NEWDOS und GDOS unterstützen ISAM-Files

Bild 3 demonstriert dies für den Apple. In den Zeilen 10 bis 60 wird auf einen sequentiellen File der String »0123456789« geschrieben.

In der Schleife ab Zeile 110 wird der Filepointer jeweils auf die relativen Bytes 0 bis 9 gestellt und ab da gelesen. Das Ergebnis zeigt das Bild unter dem Listing. Der Trick beim ISAM-File ist nun schon fast zu erraten. Wenn man auf einen sequentiellen File Sätze verschiedener Länge schreibt, und zu jedem Satzbeginn den Filepointer-Wert notiert, dann kann man auch direkt auf jeden Satz zugreifen. Praktisch werden die Filepointer-Werte in einem Array notiert, der dann separat auch auf der Disk gespeichert wird. Somit muß man gar nicht die Filepointer kennen, sondern kann über den n-ten Array-Index auf den n-ten Satz zugreifen.

Im Prinzip entspricht also der Index der Record-Nummer im Random-File. Einzusehen, daß man keinen Index-Array braucht, wenn alle Sätze gleich lang sind.

Dann kann man den Filepointer-Wert auch errechnen, und genau das tut das DOS, wenn Sie einen Random-File definiert haben, mehr nicht.

Bild 4 zeigt nun die vollständige Lösung für den Apple. Sie müssen wissen, daß der Filepointer (B-Parameter) ab Null zählt, folglich muß der erste Index auch Null sein. Das ist er, weil die Schleife ab Zeile 50 mit 1 startet. Den letzten Wert, den Filepointer nach dem letzten Satz, benötigen wir auch nicht, weshalb die Leseschleife ab Zeile 110 nur bis 2 läuft. Zeile 62 rechnet den Filepointer-Wert über die Länge der Sätze. Das automatisch hinzukommende Schlußzeichen je Satz ist mitzuzählen.

Bild 5 bringt ein ISAM-Beispiel für die Modelle und TRS-80 und Video-Genie unter NEWDOS beziehungsweise GDOS. Der Vorteil liegt hier darin, daß über die LOC-Funktion der Filepointer abgefragt werden kann, beim Apple und anderen muß man dessen Wert selbst rechnen, siehe Zeile 62 im Apple-Listing.

Die bisher vorgestellten ISAM-Files haben nur ein Index-Array, auch Schlüssel genannt. Schon mit einem Schlüssel ist allerhand möglich, zum Beispiel sortieren. Dazu muß man zwar den Daten-File noch lesen, in ihm aber kein einziges Byte bewegen. Sortiert wird nur der Index. Für diverse Sortierkriterien kann man sogleich die richtigen Index-Files bereit halten. Der zweite Weg besteht darin, den Index-File zu teilen. In einem hält man zum Beispiel für eine Kunden-Datei alle Pointer auf Großkunden, im nächsten Index-File alle Pointer auf Kunden für die Produktgruppe A. In Adreß-Dateien ist es sinnvoll, so viele Index-Files zu führen, wie im Menü Kriterien angeboten werden. Kommt dann eine Adresse hinzu, wird sie immer ans Ende des Daten-Files gehängt und der entsprechende Index-File um den Filepointer-Wert erweitert.

Schließlich kann man noch mit mehreren Schlüsseln (Index-Files) arbeiten. Schon bei der Erfassung läßt man zwei Pointer (oder mehr) mitlaufen.

Das war nur ein kleiner Ausschnitt von dem was ISAM-Files alles bieten. (P. Wollschläger/bo)

»Alle Menschen sind klug; die einen vorher, die anderen nachher.«

Chinesisches Sprichwort

»Die Welt ist überhaupt nur dadurch weitergekommen, daß irgend jemand die Courage gehabt hat, an Dinge zu rühren, von denen die Leute, in deren Interesse das lag, durch Jahrhunderte behauptet haben, daß man nicht an sie rühren darf.«

Arthur Schnitzler,
österreichischer Schriftsteller
(1862–1931)

»Es ist nett, wichtig zu sein. Aber es ist wichtiger, nett zu sein.«

Graffito an einer Mauer

Einlesen von Programmzeilen beim TRS-80

Manchmal ist es durchaus nötig, Programmzeilen erst während des Programmablaufs einzugeben. Dies ist zum Beispiel bei Plotprogrammen nützlich, da hier oft mehrere Funktionen nacheinander geplottet werden sollen. Ferner können Sie dadurch auch Eingaben dauerhaft in Data-Zeilen speichern. Deshalb soll an folgenden Programmzeilen demonstriert werden, wie einfach dies unter NEWDOS/80 geht.

```
 10 PRINT "Geben Sie bitte die Funktion ein"
 15 PRINT "Die unabhängige Variable ist X"
 20 INPUT "Y=";A$
 30 ZL$="1000 Y="+A$+":RETURN"+CHR$(13)
 40 OPEN "O",1,"FUNKTION/BAS", "FI"
 50 PUT 1,,,(LEN(ZL$))ZL$;
 60 CLOSE(1)
 65 CMD"F",DELETE 1000
 70 MERGE "FUNKTION/BAS"
 75 ON ERROR GOTO 200
 80 X=10
 90 GOSUB 1000
100 PRINT "Y(";X;")=";Y
110 END
200 IF ERL=1000 THEN PRINT "Fehler in der Funktionseingabe"
210 RESUME 10
```

Das ganze Geheimnis an der Sache ist, daß nicht nur der LOAD-Befehl, sondern auch der MERGE-Befehl auf ASCII-Files anwendbar ist. Deshalb müssen die einzulesenden Programmzeilen lediglich als ASCII-File abgelegt werden. Dies ist eine der leichtesten Übungen für NEWDOS, deshalb sei hier nur noch kurz das Rezept in Stichworten erläutert: Die Programmzeile mit Chr$(13) abschließen und ein PUT auf ein FI-File. Für die korrekte Syntax ist ausschließlich der Anwender des Programms verantwortlich. Deshalb gehört ein Abfangen von Fehlern in den eingelesenen Zeilen mit einem ON ERROR GOTO zur unbedingt notwendigen Austattung des Programms. Die Anweisung ON ERROR GOTO darf allerdings erst nach dem MERGE-Befehl stehen. (M. Ebinger/bo)

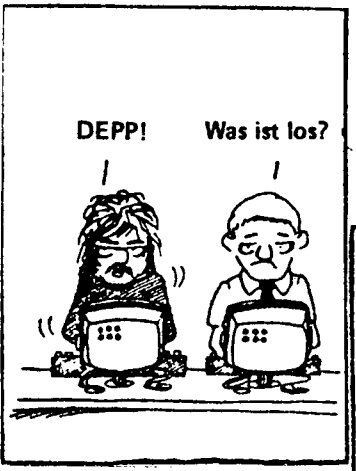
Peeks & Pok's -- Peeks & Pok's -- Peek

```
POKE  16413,0  schaltet den Bildschirm ab.
               Du hast keine Daten oder Eingaben
               mehr auf dem Bildschirm, obwohl alle
               Eingaben vom Programm akzeptiert werden.

POKE  16413,7  schaltet den Bildschirm wieder ein.
               Anwendungen dieses Pok's :
               bei z.B. Passwordeingaben
```

Ulrich Böckling

Peeks & Pok's -- Peeks & Pok's -- Peek



Einen Rechner im Herzen der Schweiz
packt beim Rechnen ganz plötzlich der Geiz.
Er addiert voller List
auch noch dort, wo nichts ist –
das hat grad in der Schweiz seinen Reiz.

Ein Kurarzt im schönen Bad Pfrund
prüft am Terminal jeden Befund.
Diagnose per Kabel
ist für ihn keine Fabel –
und ihn selbst macht das ganz schön gesund.

Ein End-User bei Bayrisch Zell
rechnet sicher und unheimlich schnell.
Im April wird er Vater,
das gibt sicher Theater –
so verrechnet sich jeder mal, gell?

Eine Lochkartendame in Herne
lochte stets ihre Karten recht gerne.
Eines Tages, oh Schreck,
war der Locher dann weg –
jetzt verarbeitet sie in der Ferne.

Ein Computel steht auch in Shanghai
und zählt stets binäl bis auf zwei.
Doch weil viele Millionen
von Chinesen da wohnen,
ist das Zählen schon fast Quälelei.

Ein Computerexperte aus Bruhl
macht jetzt Hochzeit, doch bleibt er ganz cool.
Statt von Liebe und Leid
spricht von Bit er und Byte –
und die Braut schluchzt: „Du hast kein Gefool!"

Mission Impossible

Ratschläge

Die Kommandos können durch die ersten drei Buchstaben abgekürzt werden.
SAVE GAME speichert den Spielstand auf Kassette.
HELP und SCORE sind keine Hilfen.
Der Befehl EXAMINE bringt dafür um so mehr. Er sollte möglichst oft im Bezug auf gefundene oder sehbare Gegenstände angewandt werden.

1. Bis zur Apparatur mit den mehrfarbigen Knöpfen und dem Finden des Saboteurs sollten Sie es alleine schaffen.
Bei den Knöpfen hilft nur ausprobieren. Achten Sie dabei auf den Bombendetektor. Dies gilt für jede Benutzung der Knöpfe.
2. Die Karten mit ihrem Bild verschaffen Ihnen jeweils Eintritt in bestimmte Räume.
3. Im Besucherraum sollten Sie auf das Fenster achten.
4. Zu öffnen geht es nicht. Also weiter mit Gewalt. Das Mittel dazu sehen Sie sehr früh.
5. Das Sicherheitssystem ähnelt dem der Tür. In welchem Zusammenhang tauchte "window" schon mal auf?
6. Jeder Schlüssel paßt in ein Schloß. Wo haben Sie schon welche gesehen?
7. Ein Mop mit komischen Geräuschen? Stellen Sie sich das Ding bildlich vor. Was könnte damit sein?
8. Immer noch nichts? SUCHEN hilft.
9. Eine Tür läßt sich nicht öffnen. Da gibt's nur eins: hart bleiben.
10. Filmkassette? Projektor? Vorführung? Viel Laufarbeit, aber es lohnt sich.
11. Aha, Schutzanzug ist nötig. War eigentlich klar. Ist ja auch ein Reaktor. Aber das Plastik?
12. Die Bombe tickt vor sich hin, ohne Sie zu beachten? Trennen Sie das Ding von seinem Nabel. Das Mittel ist in jeder Werkzeugkiste vorhanden. Und Sie haben es auch schon gesehen.
13. Was tun mit dem Ding? Was tun, wenn es brennt?
14. Sie löschen Feuer doch auch mit Wasser. Sie müssen es nur dahin bekommen, wo es hin soll.
15. Sagen Sie bloß nicht, sie haben 11. vergessen! Wenn der Krug nicht zum Brunnen geht, muß ...
Na also!

Achtung !!!

Nicht umblättern, da die

die Auflösung auf der Rückseite ist.

Versuch's doch erst selber!

Also bitte .... die Auflösung

Mission Impossible

Die Situation und der Sinn des Spiels lassen sich aus dem Anfangstext und der ersten Situation ersehen.

Die Räume
Jeder Raum erhält eine Nummer. Die möglichen Ausgänge werden den Nummern der Räume zugeordnet, zu denen sie führen.

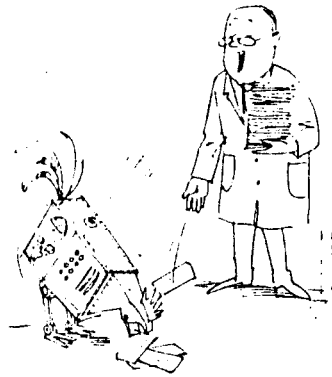
1. Briefing room; W=2
2. Large grey corridor; N=3, S=4, W=5, E=1, U=6, D=7
3. Twisting blue hallway; N=9, S=2, W=6
4. Grey room; N=2
5. Maintenance room 1; E=2
6. Twisting white hallway; N=10, E=3, D=2
7. Twisting yellow hallway; U=2, N=8
8. Yellow room; S=7, Tür=12
9. Blue room; S=3, Tür=15
10. White room; S=6, Tür=11
11. Visitors room; Tür=10
12. Yellow corridor; W=13, Tür=8
13. Maintenance room 2; E=12, U=14
14. Projectionist room; D=13
15. Anteroom; Tür1=9, Tür2=18, W=17, U=16
16. Viewing room; D=15
17. Storage room; E=15
18. Control room; E=19, D=20, Tür=15
19. Break room; W=18
20. Reactor core; U=18

Der Spielverlauf:
Sie beginnen immer in Raum 1. Nach INVENT bemerken Sie, daß Sie einen Bombendetektor tragen, der Sie warnt, wenn die Bombe scharf ist.
Neben sich finden Sie einen Recorder. Mit TAKE RECORDER und START RECORDER wird Ihnen die Situation noch einmal erklärt.
Begeben Sie sich in Raum 5. Dort finden Sie einen Plastikeimer, der später wichtig für Sie ist. TAKE PAIL.
Sodann gehen Sie in Raum 4, wo Sie sich mit SIT vor eine Apparatur setzen. PESS RED und PRESS WHITE ergeben eine Reaktion, die sich nach GET UP und GET PICTURE in Form einer Besucherkarte vorfinden.
Nun warten Sie, bis in der Ferne ein Fall zu hören ist. Durchsuchen Sie alle für Sie bis jetzt erreichbaren Räume, bis Sie den Saboteur finden. Durchsuchen Sie ihn mit FRISK HIM und nehmen Sie sodann seine Karte (bzw. PICTURE) und ihn selber.
In Raum 10 verschaffen Sie sich mit SHOW AUTHORIZATION Einlaß zu Raum 11. Mit SMASH WINDOW, WITH RECORDER. SHOW AUTHORIZATION, ENTER WINDOW, GET KEY, ENTER WINDOW erlangen Sie ein weiteres wichtiges Utensil. Durch PRESS WHITE verlassen Sie den Raum.

Vor der Apparatur in Raum 4 sitzend geben Sie UNLOCK YELLOW ein. Nun drücken Sie hintereinander die Knöpfe weiß, rot, gelb, weiß und holen sich ihr nächstes Bild ab, diesmal als Instandhalter.
Vor Raum 8 zeigen Sie wieder ihr Bild und begeben sich in Raum 13. Mit SEARCH MOP finden Sie einen weiteren Schlüssel. Die Zange nehmen Sie auch gleich mit.
In Raum 4 holen Sie sich mit rot, blau und weiß nach Aufschließen des blauen Knopfes die Karte "Sicherheitspersonal".
Nachdem Sie damit über die Räume 9 und 15 nach Raum 17 gelangt sind, ziehen Sie sich mit WEAR SUIT den Schutzanzug an und nehmen mit GET WATER Wasser mit (dazu brauchen Sie den Eimer).
Die Tür zu Raum 18 öffnen Sie mit PUSH HARD und nach ENTER DOOR finden Sie dort eine Filmkassette.
Sie können die Kassette mit INSERT FILM in den Projektor im Raum 14 einlegen und dann in Raum 11 mit PRESS GREEN ein kleines Filmchen sehen.
Zurück in Raum 18 stellen Sie den Eimer mit dem Wasser ab, denn Plastik verformt sich in Reaktornähe.
In Raum 20 ist endlich die Zeitbombe, die Sie mit CUT WIRE lösen und dann mitnehmen in Raum 18. Dort leeren Sie mit POUR PAIL den Eimer über die Bombe, die damit entschärft ist.

Gerald Schröder

1966: »put – put – put . . .«



2066: »input – input – input . . .«

Laufwerkmodifikation

Was macht man, wenn man 40-Spur-Disketten auf 80-Spur-Laufwerken laufen lassen möchte? Ganz einfach, man stellt den PDRIVE passend ein.

Was macht man, wenn man selbstbootende 40-Spur-Disketten auf einem 80-Spur-Laufwerk booten möchte? Auch ganz einfach, bei einem neuen TEAC 55F 80 -Spur-Laufwerk überbrückt man die beiden Lötpunkte, die für den Widerstand R 14 vorgesehen sind mit einem 10-Ohm-Widerstand und schon hat man ein 40-Spur -Laufwerk.
Ich habe den Widerstand mit einem Minischalter umschaltbar gemacht und kann so hardwaremäßig von 80-Spur-Drive auf 40-Spur-Drive umschalten. Klappt prima.

Ulrich Böckling

Um eine 40-Track-Disk auf einem 80-Track-Laufwerk zu booten gibt es auch eine Softwaremöglichkeit. Man muß nur zuerst ein paar Bytes im Bootsektor der 40-Track-Disk umzappen. Danach kann man die Diskette auf einem 80-Track -Laufwerk booten.
Mit den Zaps wird erreicht, daß das 80-Track-Laufwerk zwei Steppschritte macht und somit jeden 2. Track anspricht. Wie bei PDRIVE die Einstellung AL.

Um die gleiche Diskette auf 40-Track-Laufwerken zu booten sollte die Änderung wieder rückgängig gemacht werden. Ist ja klar.

Die zu ändernden Zeilen sind wie folgt :

Bootsektor
Spur 0, Sektor 0

Bootsektor alt

```
A0  CB7E 28E6 7E36 D0C1 D1E6 FC20 0C1C 7BD6 .B(.B6.....ä.
B0  0A20 0314 1E00 D97E C9CD D742 360B 1098 .....ß...B6...
C0  21DD 427E FE03 28FB 23CD 3300 18F5 CDD7 !.BB..(.#.3.....
D0  42CB 4620 FC7E C93E 063D 20FD C91C 1F45 B.F..B.>.=.....E
E0  5252 4F52 031C 1F4E 4F20 5359 5303 18F1 RROR...NO.SYS...
F0  213D 6E18 D021 596E 18CB 2185 6E80 0704 !=n..!Yn..!.n....
```

Bootsektor geändert

```
A0  CB7E 28E6 7E36 D0C1 D1E6 FC20 0C1C 7BD6 .B(.B6.....ä.
B0  0A20 03CD E542 D97E C9CD D742 360B 1098 .....B.B...B6...
C0  21DD 427E FE03 28FB 23CD 3300 18F5 CDD7 !.BB..(.#.3.....
D0  42CB 4620 FC7E C93E 063D 20FD C91C 1F45 B.F..B.>.=.....E
E0  5203 4E53 0314 1E00 36D0 3643 CDCE 4236 R.NS....6.6C..B6
F0  D036 43CD CE42 36D0 CDCE 427A 32ED 37C9 .6C..B6...Bz2.7.
```

Jens Neueder

Auf Seite 67 (Info 6) schreibt Walter, daß man Edtasm von Debug aus aufrufen könne. Nachteil sei nur, daß der Cursor nicht sichtbar sei. Diesen Nachteil kann man im NEWDOS80 mit Zap # 22 beheben:

Edtasm/CMD ,05 ,12 ändern von CD 39 59 0E
in CD 00 57 0E

Edtasm/CMD ,03 ,1A von 40 7D E6 3F C0 11 C0 FF 19 C9
in 40 CD 39 59 3E 0E C3 39 59 C9

Noch ein anderes nützliches Zap, wenn im Basic die Abkürzungen L ,E ,D ,A für LIST ,EDIT ,DELETE ,AUTO im Direktmodus nicht immer erkannt werden:

SYS18/SYS ,02 ,31 ändern von 2C D7 28 06 FE
in 2C C3 59 55 FE

SYS18/SYS ,03 ,68 von 1B 00 00 00 00 00 00 00 00 00 00
in 1B D7 FE 0D CA 2F 54 C3 29 54 00

Herbert Alber

Ein weiteres Problem vom Clubtreffen will ich gleich noch an dieser Stelle klären:
Viele von Euch haben das Schachprogramm SARGON3. Aber es läuft nicht unter Disk-Betrieb. Dies ist nämlich ein Kassettenprogramm, das aber unter Newdos mit folgendem Trick zu starten ist:
1.: Load SARGON3/CMD
2.: BASIC2
3.: SYSTEM
4.:/48128
So funktioniert es bei mir einwandfrei. Der Grund für das seltsame Verhalten: Diese Programm ist ursprünglich in einem nicht TRS-Kassettenformat gespeichert und dehalb versagt auch LMOFFSET.

W.Zwickel

Pi ganz einfach

In allen Basic-Lehrbüchern, die ich kenne – selber besitze ich gegen zwei Dutzend – und in allen Programmen, die ich bisher gesehen habe, wird die Zahl Pi mit ihrem Zahlenwert definiert:

...Pi = 3.141592..
Es wäre sinnvoller, die Zeile stattdessen zu schreiben:
...Pi = 4*ATN(1)

Denn erstens irrt man sich dabei nicht, wie beim Eintippen von Ziffern, zweitens geht es leichter und drittens wird dabei die Genauigkeit des jeweiligen Rechners ausgenützt.

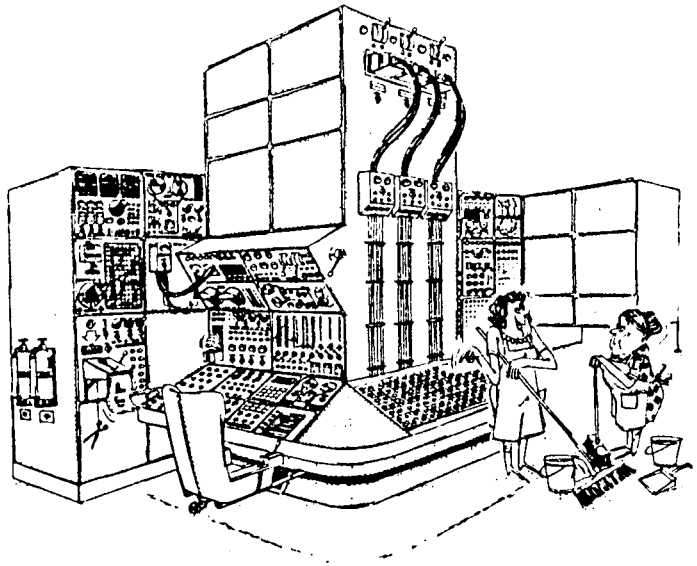
Heinrich Kraft, Kiel

Ausgabeumschaltung in BASIC

Im letzten Clubinfo war ein Beitrag vom Josef Konrad abgedruckt, in dem er beschrieb, wie man Ausgaben, die normalerweise auf dem Bildschirm erscheinen, auf den Drucker umleitet. Die in diesem Beitrag aufgezeigte Methode hat den Nachteil, daß nur die Texte umgeleitet werden, die durch den PRINT-Befehl ausgegeben werden. Texte, die anders auf den Bildschirm gebracht werden (z.B. INPUT"Gib was ein";A$), werden davon nicht berührt.
Abhilfe schafft hier die Verwendung des ROUTE-Kommandos des NEWDOS80. Wie dies zu bewerkstelligen ist, habe ich unten kurz aufgezeigt. Benutzt man diese Methode, wird alles, was normalerweise auf dem Bildschrim angezeigt wird, auf den Drucker umgeleitet. Wichtig ist dabei nur, daß man nach Beendigung des Programms die ROUTE-Anweisung durch ROUTE,CLEAR (bzw. CMD"ROUTE,CLEAR) wieder rückgängig macht.

```
10 CLEAR 1000 : CLS
20 PRINT @ 0,"Ausgabe auf (D)rucker oder (B)ildschirm?"
30 AINKEY: IF A"D" OR A"d" THEN 50
40 IF A"B" OR A"b" THEN 60 ELSE GOTO 30
50 CMD"ROUTE,DO,PR" : GOTO70
60 CMD"ROUTE,CLEAR"
70 FOR X=32 TO 191 : PRINT CHRX);" "; : NEXT : RUN
```

Hartmut OBERMANN



»Na, wie wär's – einfach auf ein paar Knöpfe drücken und dann nix wie weg!«

Im Clubinfo Nr. 6 veröffentlichte unser Assembler-Freak Arnulf Sopp einen Beitrag mit dem Titel "Nochmals: Joystick Anschluß". Die darin aufgezeigte Möglichkeit einen Joystick an ein VideoGenie anzuschließen oder die Tastatur um weitere Tasten zu erweitern ist durchaus praktikabel und bringt eine Menge zusätzliche Möglichkeiten ohne viel zu kosten.

Aber es ist Vorsicht geboten bei dem Versuch, die Vorgehensweise, die der Arnulf beschreibt, auf den TRS80 zu übertragen. Prinzipiell ist zwar die Tastatur gleichartig aufgebaut, schaltungstechnisch unterscheiden sich die Computer stellenweise jedoch erheblich.

Eine dieser Stellen ist die Tastatur der beiden kompatiblen Computer. Während die Tastaturplatine des VideoGenie nur die Tasten und ein paar Widerstände beherbergt, sitzen auf der TRS80-Tastatur zusätzlich vier Integrierte Schaltkreise. Diese IC's besitzt der TRS80 nicht etwa mehr als das VideoGenie, sie sind nur beim Nachbau (und nichts anderes ist das VideoGenie) von der Tastaturplatine auf das CPU-Board verlagert worden. Aus dieser Tatsache ergeben sich einige Umstände, wenn man den Joystickanschluß trotzdem so realisieren will, wie der Arnulf ihn beschreibt oder wenn man sich ein paar zusätzliche Funktionstasten einbauen will.

Am besten beschafft man sich eine Einbaubuchse mit 16 Anschlüßen, auf die man die Tastaturspalten- und -reihen-Leitungen legen kann. Die Spaltenleitungen sind relativ einfach zu finden, man kann sie direkt an den 4.7kΩ abnehmen. Dabei muß man nur darauf achten, daß man die richtige Seite des Widerstands erwischt. Die Seite von der aus eine Leiterbahn zu irgendeiner Taste führt ist die richtige! Die Reihenleitungen sind auch nicht viel schwieriger ausfindig zu machen. Man verwendet am besten die Anschlüße der IC's, die folgendermaßen zugeordnet sind:

Spalte 1 Widerstand R8	Reihe 1 IC 1 Pin 8
Spalte 2 Widerstand R5	Reihe 2 IC 1 Pin 2
Spalte 3 Widerstand R3	Reihe 3 IC 1 Pin 10
Spalte 4 Widerstand R2	Reihe 4 IC 2 Pin 2
Spalte 5 Widerstand R7	Reihe 5 IC 1 Pin 6
Spalte 6 Widerstand R1	Reihe 6 IC 1 Pin 4
Spalte 7 Widerstand R4	Reihe 7 IC 1 Pin 12
Spalte 8 Widerstand R6	Reihe 8 IC 2 Pin 4

Diese sechzehn Punkte verbindet man mit den Anschlüßen der Buchse und kann daran dann, wie vom Arnulf beschrieben, die Tastaturerweiterungen anschließen.

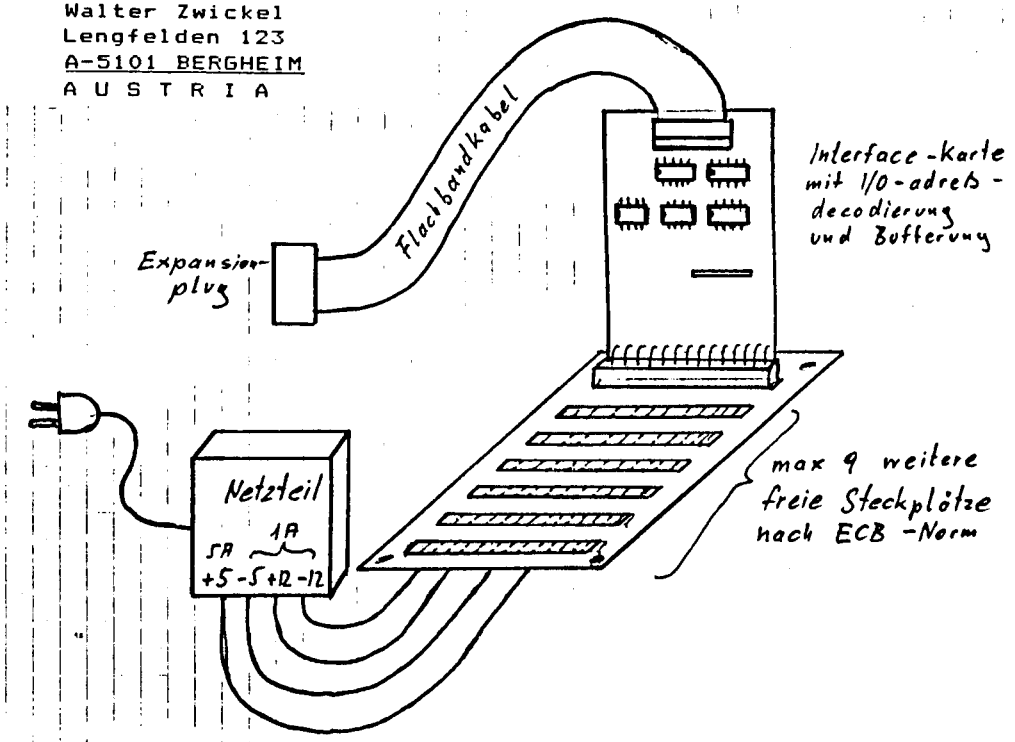
Auch ich wünsche euch beim Basteln viel Spaß und gutes Gelingen!!!

Hartmut Obermann

Hardware - Arbeitskreis

E C B - B U S System

Walter Zwickel
Lengfelden 123
A-5101 BERGHEIM
A U S T R I A



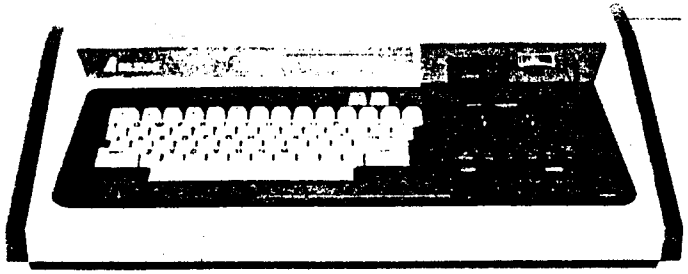
Liebe Clubfreunde

Wie die meisten von Euch sicherlich schon wisen, haben wir beim Clubtreffen einen Hardware-Arbeitskreis gebildet. Als erstes Projekt wollen wir ein universelles ECB-Bus System bauen. Das soll vorerst von den Mitgliedern des Arbeitskreises entwickelt werden. Wobei jeder Mitstreiter einen Teil der aufzubringenden Arbeit leistet. Wenn das Projekt fertig ist, sollen fertige Platinen und eine Bauanleitung vorliegen, die es auch dem elektronisch nicht so vorbelasteten Clubmitglied erlauben das ECB-Bus System aufzubauen.
Meine Planungen sind inzwischen soweit fortgeschritten, daß ich unbedingt auch die Meinungen der übrigen Hardware-Fans brauche. Dazu will ich das System kurz beschreiben:
Die Basisplatine soll eine ECB-Platine von der Zeitschrift c't sein. Diese hat 10 Steckplätze mit sogenannten VG-Steckern mit je 96 Kontakten in 3 Reihen angeordnet. Diese Platine kostet fertig ca. 50.- DM . Das ECB-System braucht nur 2 dieser Reihen, sodaß noch 32 Leitungen für TRS-spezifische Dinge zur Verfügung stehen. Der Anschluß an den TRS/VG/Komtek soll mit einer von mir zu entwickelnden Karte geschehen. Wobei Pufferung aller Signale vom und zum Computer, Adreßdecodierung der wesentlichen Portadressen sowie Terminierung der vom Computer kommenden Leitungen die Hauptaufgaben sind. Die Terminierung macht mir noch etwas Sorgen. Es stehen 3 Varianten zur Auswahl:
1. Aktive Terminierung (m.E. zu aufwendig, da eine weitere Karte notwendig würde)
2. Niederohmige Terminierung mit 220 und 330 Ohm. Ist leicht machbar, aber verbraucht ca 400 mA. Kann auch Probleme machen, wenn im Computer Treiber IC's der LS-Type verwendet werden.
3. Hochohmige Terminierung mit 2.2 und 3.3 Kohm. Erfordert ein Verbindungskabel zum Computer mit max. 60 cm Länge.
Ich habe vor die Karte so zu gestalten, daß man wahlweise die Variante 2 oder 3 bestücken kann.
Das zweite Problem ist die Steuerung des Datenbus-Puffers. Der muß ja auch auf read geschalten werden und darf keinesfalls versehentlich adressiert werden, sonst knüppelt er ja die Signale am Datenbus zusammen. Nur mit I/O request klappt das nur, wenn man sich auf die I/O Adressen beschränken würde. Aber das wollen wir ja nicht. So werde ich wahrscheinlich auf der Interface-card nur die I/O Adressen decodieren und falls jemand eine Karte mit Memory-Adressen baut, muß er eben selber ein CHIP-SELECT Signal erzeugen und auch zur Treiberkarte senden. Leitungen sind auf der 96-poligen Buchse ja genug frei.
Teilt mir bitte mit, ob Ihr damit einverstanden seid, oder ob jemand eine bessere Idee für obige Probleme hat. Ich habe meine ECB-Platine schon bestellt und habe auf jeden Fall beschlossen, eine Interface-Karte in Fädeltechnik zu bauen und alles auf Herz und Nieren zu testen, bevor es an eine Platine geht. Deshalb bitte ich auch um Geduld, daß es sicher noch ein paar Monate dauern wird, bis die fertigen Platinen vorliegen.

Herzliche Grüße aus Salzburg

Lieber INFO-Leser
Wenn Du in der INFO einen Druckfehler findest,
dann bedenke bitte, daß dieser beabsichtigt ist.
Die INFO bringt für jeden etwas, und es gibt
immer Leser, die nach Fehlern suchen.

Ein sicheres Plätzchen



Maschinenroutinen geschützt im GENIE untergebracht

Helmut Bernhardt

Die Sicherung von Maschinenprogrammen im oberen Adreßbereich durch Befehle wie HIMEM oder MEMSIZE gewährt lediglich Schutz gegen Überschreiben durch BASIC-Programme. Der Computer legt Maschinenprogramme beim Laden immer in den Speicherbereich, für den sie vorgesehen sind. Die Pointer werden dabei nicht berücksichtigt, so daß es normalerweise keinen Speicherbereich gibt, der wirklich 'sicher' ist. Mit einer kleinen Zusatzschaltung kann man jedoch ein 'sicheres Plätzchen' schaffen.

Auf der Suche nach diesem Platz im Adreßraum des TRS-80 oder GENIE I, II findet man schnell einige 'Lücken'. So beginnt der Adreßbereich der Tastatur bei 3800h und endet bei 38FFh. Da aber die Dekodierung des Freigabesignals für die Tastatur-Lesetreiber unvollständig ausgeführt ist, wird auch der Bereich von 3900h bis 3BFFh blockiert.

Das muß man nicht so hinnehmen. Durch geringfügige Eingriffe in den Rechner läßt sich die Tastatur vollständig dekodieren und gleichzeitig ein Freigabesignal für einen zusätzlichen Speicher-Baustein im Bereich 3900h bis 3BFFh gewinnen. Die dazu notwendige Hilfsschaltung zeigt Bild 1. Der zusätzliche Speicher besteht aus zwei ICs 2114 (1 K x 4 Bit); die weitere Dekodierung geschieht mit zwei TTL-ICs.

In den so gewonnenen 768 Byte RAM sind eigene Routinen ziemlich sicher, da man normalerweise nicht auf diesen Speicherbereich zugreifen kann. Programme mit eigener Treiberroutine könnten allerdings versuchen, die Tastatur im Bereich 3900h bis 3BFFh auszulesen. Solche Routinen würden laufend 'Eingaben' aus dem RAM erhalten, was zu Fehlfunktionen führt.

In solchen (seltenen) Fällen hilft dann nur noch der Einbau eines Zweifach-Umschalters, der entweder das unvollständig dekodierte oder das neu gewonnene Freigabesignal an die Tastatur-Lesetreiber legt und die Freigabe der RAMs bei Benutzung des 'alten' Freigabesignals für die Tastatur abschaltet. Programme, die diesen Schalter benötigen, lassen sich dann nicht mit dem zusätzlichen RAM verwenden. Das Betriebssystem NEWDOS-80 gehört glücklicherweise nicht zu dieser Gruppe von Software.

Eingriffe

Die vier ICs der Hilfsschaltung kann man auf einem Streifen Lochraster-Platine unterbringen. Die Versorgungsspannung (+5 V) ist jeweils an Pin 14 der TTL-ICs und an Pin 18 der RAM-Bausteine anzuschließen. Der Masseanschluß ist bei den RAMs Pin 9, bei den TTL-Bausteinen Pin 7. Zwischen Versorgungsspannung und Masse sollte man einen Kondensator (100 nF) schalten.

Beim Einbau der Karte in GENIE-Rechner kann man folgendermaßen vorgehen:

Zuerst ist das Gehäuse des Computers zu öffnen. Nachdem man die acht tiefer versenkten Schrauben in der Bodenwanne des Gerätes herausgedreht hat, läßt sich die obere Gehäuseschale abheben. Schraubt man noch die Tastatur ab, liegt links das CPU-Board, in der Mitte das Interface-Board und rechts das Netzteil sowie der Kassettenrecorder.

Die durchzutrennende Leitung (siehe Bild 1) ist die Leiterbahn, die vom hintersten IC (Z35, 74LS32) auf dem CPU-Board in der rechten Spalte nach 'hinten' verläuft (neben der 5-V- und Masse-Leitung). An die beiden Seiten der Trennstelle kann man den Ein- und Ausgang des 'OR'-Gatters der Hilfsschaltung anschließen. Das IC 'Z12' findet man an zweiter Stelle (von vorne) in der linken Spalte auf dem CPU-Board. Der Baustein 'Z15' (74LS32) liegt in der zweiten Spalte (von links gezählt) als zweites IC.

Die Numerierung der Verbindungsleitungen zum Interface-Board, an denen die meisten Signale für die Zusatz-Schaltung abzunehmen sind, erfolgt von vorne nach hinten (Nummern 1 bis 32).

Beim Anschließen der Erweiterung sollte man unbedingt darauf achten, daß die Datenleitungen nicht an den ROMs, sondern an den Lötpunkten der Stecker für die Verbindungsleitung zum Interface-Board abgenommen werden. Die an den ROM-Chips anliegenden Datenleitungen sind wegen der Freigabeschaltung der Treiber nur in Leserichtung verwendbar. An der Steckverbindung lassen sich außer $\overline{CS}$ und $\overline{MWR}$ alle für die RAMs nötigen Signale abnehmen. □

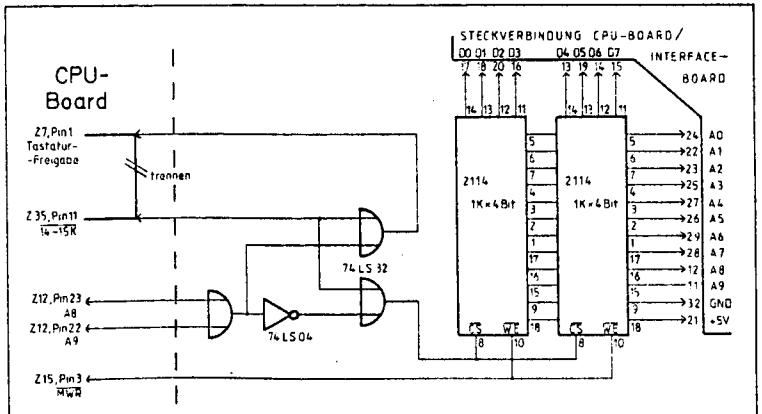


Bild 1. Die Zusatzschaltung für ein sicheres Plätzchen

SuperTape für TRS-80

Andreas Burgwitz, Andreas Stiller

Endlich schließt sich auch der TRS-80 der Gemeinde der SuperTape-Rechner an. Zuvor galt es allerdings eine Hürde zu nehmen, die die Entwickler des mittlerweile recht betagten Rechners errichteten: Vom SuperTape-Eingangssignal läßt die Schaltung des Kassettenports nichts über, was mittels Software ausgewertet werden könnte. Die Lösung dieses Problems besteht aus einer kleinen Zusatzplatine, die dem TRS-80 SuperTape-Betrieb mit 7200 Baud ermöglicht.

Gerade die Anwender des TRS-80 werden ein schnelles und vor allem störsicheres Kassettenverfahren begrüßen; sind sie doch vom Standard-Verfahren des Rechners Schlimmes gewohnt. Und gerade dieses Verfahren bescherte die Probleme, die den Einsatz einer Zusatzschaltung nötig machen. Aber wie fast alles, hat auch dies seine guten Seiten: Mit dem so notwendig gewordenen zusätzlichen Interface kann der TRS-80 SuperTape-Kassetten mit 7200 Baud beschreiben und lesen. Zusätzlich bietet die Schaltung je ein frei verwendbares 'Ein-/Ausgabebit'.

Die SuperTape-Software entspricht im wesentlichen dem in Heft 1/85 vorgestellten Programm für das Colour Genie, und damit auch dem SuperTape für den ZX-Spectrum. Allerdings sind die SuperTape-Ein-/Ausgaberoutinen an den Tandy angepaßt. Da der TRS-80 auch mit 7200 Baud lesen und schreiben soll, mußte die Laderoutine völlig neu erstellt werden.

Die TRS-80 SuperTape-Lösung kann man auf Geräten mit und ohne Diskettenlaufwerk betreiben. Sinnvollerweise sollte der Tandy aber mit minimal 16 KByte RAM ausgebaut sein. Das abgedruckte Programm liegt im Speicher ab der Adresse B900h. Natürlich kann man das Programm auch an die obere Grenze des Speichers legen, wo es aber oftmals mit einem Monitor oder anderen Utilities kollidiert.

Vor dem Start von SuperTape sollte man diesen Speicherbereich gegen Überschreiben durch BASIC-Programme schützen. Dies geschieht durch die Beantwortung der Frage 'MEM SIZE ?' mit 47300. Allerdings sichert man SuperTape damit nicht gegen Maschinenprogramme, die im Speicher ab B900h liegen.

Input

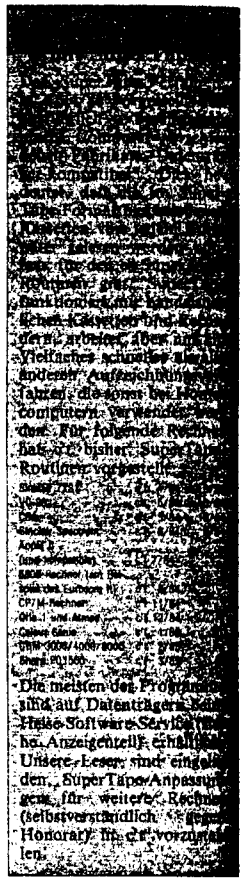
Die Eingabe des Programms in den Rechner sollte nicht ohne die Hilfe eines Monitor-Programms geschehen, das das Abspeichern der eingegebenen Routinen als 'SYSTEM-File' erlaubt. Mehr Tipparbeit, aber wesentlich weniger Ärger bei Änderungen ergibt die Eingabe mit Hilfe eines Assemblers. Verfügt man über keins dieser Hilfsprogramme, sollte man SuperTape vom c't-Software-Service als SYSTEM-File beziehen.

Bits schieben

Wie das SuperTape-Verfahren funktioniert, ist in mehreren vorausgegangenen c't-Ausgaben ausführlich erläutert worden. Die SuperTape-Programme aller Z80-Rechner orientieren sich dabei an den Routinen für den Spectrum aus c't 6/84.

Für 7200 Baud wird allerdings dabei eine einigermaßen 'flotte' CPU vorausgesetzt, die der Tandy mit seinen 1,774 MHz Taktfrequenz offensichtlich nicht aufweisen kann. Vor allen Dingen bei der Laderoutine wird es zeitlich etwas zu 'eng'.

Die optimale Prüfzeit liegt für 7200 Baud bei etwa 2/3 einer ganzen Periode für die Übertragung einer Null: also 93 Mikrosekunden. Davon ist sogar noch die mittlere Flankenerkennungszeit abzuziehen, so daß dem Tandy also ungefähr 150 bis 160 Takte zur Verfügung stehen. Außerdem sollte man nach oben und nach unten noch etwas Spiel haben, um rein empirisch die beste Prüfzeitkonstante ermitteln zu können.



Die meisten der Programme sind auf Datenträger im Heise-Software-Service (s. Anzeigenteil) erhältlich. Unsere Leser sind eingeladen, SuperTape-Anpassungen für weitere Rechner (selbstverständlich gegen Honorar) für c't vorzuschlagen.

Die Spectrum-Routine benötigt jedoch bereits 175 Takte, selbst wenn gar kein Sprung mehr in eine Warteroutine erfolgt. Daß so ein Sprung nicht ausgeführt werden soll, müßte aber das Programm auch noch feststellen, da ja für 3600 Baud auf jeden Fall eine zusätzliche Warteschleife nötig ist. Für diese Abfrage wäre erst recht keine Zeit mehr vorhanden, so daß die gesamte Laderoutine einschließlich Synchronisation zweimal geschrieben werden müßte, einmal für 7200 Baud und einmal für 3600 Baud.

Das Spectrum-Programm birgt auch noch eine Gefahr für andere Rechner: Beim Einlesen des Ports erstreckt sich die Abfrage über den gesamten Datenbus. Eine eventuell 'floatende' Datenleitung verfälscht das

Ergebnis völlig. Eine zusätzliche Ausmaskierung des einzulesenden Bits ist also vonnöten.

Über alle Register

Um 7200 Baud auch auf dem Tandy — sogar mit einiger Reserve — lauffähig zu bekommen, muß man fast alle Register der Z80-CPU 'ziehen'. Lediglich das Interrupt- und Refresh-, das IY- und das DE'-Register bleiben unberücksichtigt.

Das neue Interface legt die Signale vom Kassettenrecorder auf die Datenleitung D7. Das hat den Vorteil, daß hierbei bereits mit dem IN-Befehl das Bit abgefragt werden kann. Allerdings muß dafür das Register C die Portadresse enthalten, also IN A,(C). Die Abfrage JP M beziehungsweise JP P wertet dann unmittelbar das eingelesene Bit auf D7 aus. Damit läßt sich die Flankenabfrage innerhalb von nur 22 Takten durchführen, was der Lesesicherheit sehr zugute kommt.

Der eingelesene Wert wird als Carry-Flag in AF' zwischengespeichert. Falls der neu eingelesene Wert mit dem alten übereinstimmt, erkennt das Programm eine Null; bei Ungleichheit eine Eins. Bei einer Eins wird das Prüfsummenregister HL inkrementiert. Das entsprechend gesetzte Carry-Flag gelangt dann in bewährter Manier per Rotation ins E-Register, das so Bit für Bit bis zu dem gewünschten Byte aufgefüllt wird. Das Programm erkennt die 'Fülle' an dem Endflag, das am Anfang (Adresse BCC9h) gesetzt wird und dann nach acht Rotationen ins Carry gelangt.

Beim bitweisen Einlesen innerhalb der ersten Synchronisationsstufe findet — bedingt durch den Befehl SET E,0 — nur eine Rotation statt.

Der indirekte Sprung über das IX-Register ermöglicht eine einfache und schnelle Verzweigung (8 Takte) in eine Warteschleife, die der gewählten Baudrate entspricht. Bei 7200 Baud läßt sich so die Wartezeit im 'Viertakterhythmus' durch Einsprung in die NOP-Reihe von LDW36 bis LDSAM verändern. Der Benutzer von SuperTape sollte sowohl diese Einsprungadresse für 7200 Baud (in Adresse BC25h) als auch die Wartekonstante LCON36 für 3600 Baud etwas variieren, um die für seinen Recorder besten Werte zu finden.

Immerhin 22 Takte ließen sich gegenüber den Spectrum-Routinen dadurch sparen, daß für die Blocklade- und Vergleichsoperationen die gestrichenen Register herangezogen wurden. Die zeitkritische Situation am Ende der Synchronisation konnte entschärft werden, da die Routine gleich in die LDBYT-Routine 'überläuft' und somit bereits mit einem geladenen Byte zurückkehrt. Dieses eine Byte ist allerdings extra zu behandeln, bevor die eigentliche Lade- beziehungsweise Verify-Schleife beginnt.

Summa summarum kommt die Laderoutine einschließlich des JP (IX)-Befehls mit 139 Takten 'Eigenzeit' aus.

Gut ge'toggle't

Beim SAVE liegen die Dinge wesentlich einfacher. Da das Interface auch noch die Arbeit des 'toggelns' übernimmt, also bei jedem Aufruf den Portzustand ändert, wird das sonst nötige Toggle-Register frei. Dieses findet im Programm Verwendung als Flagregister, das das Baudratenflag schnell abrufbar speichert. Bei 3600 Baud (Bit 7 von D = 0) wird dann eine zusätzliche Wartezeit von 124 Takten eingeschoben.

Der Rahmen

Den Rahmen um die eigentlichen SuperTape-Routinen bildet ein menügesteuerter Programmteil, der die Anwendung von SuperTape komfortabel macht. Nach dem Laden von SuperTape meldet sich der TRS-80 mit:

```
SUPERTAPE FUER TRS-80, MODELL 1
(S)AVE (Q)UICK (L)OAD (V)ERIFY (E)XIT
COMMAND:
```

Durch die Eingabe eines in Klammern stehenden Buchstabens verzweigt das Programm zu der entsprechenden Routine und fordert die eventuell noch benötigten Angaben an. Gibt man zum Beispiel 'S' für Speichern mit 3600 Baud (oder Q — Speichern mit 7200 Baud) ein, fragt das Programm nach dem Namen des abzuspeichernden Programms:

```
NAME:        .TYP
```

Allerdings sollte man bei 7200 Baud einen besseren Recorder verwenden als das Tandy-Gerät.

Nach der Eingabe des Namens gelangt man durch Betätigen der 'RETURN'-Taste zur TYP-Abfrage, zum Beispiel 'BAS' für BASIC-Programme oder 'HEX' für Speicherinhalte. Beim Typ 'HEX' fragt das Programm nach der Start- und Endadresse des Speicherbereichs. Diese Frage entfällt beim Typ 'BAS'; das Programm schreibt ein BASIC-Programm vom Anfang bis zum Ende (im Rechner durch Zeiger markiert) auf Band.

Bei der folgenden Frage 'TAPE READY ?' hat man noch die Möglichkeit, sofort wieder zum Menü zurückzukehren, indem man diese Frage mit 'N' beantwortet. Andernfalls schreibt das Programm die Daten auf Band.

Alle Funktionen, die man schon einmal ausgeführt hat, kann man im Menü durch die Eingabe von 'Shift Buchstabe' erneut aufrufen, wobei die zuvor eingegebenen Parameter übernommen und angezeigt werden.

Soll die Aufzeichnung sogleich überprüft werden, genügt die Eingabe von 'v' (Shift), da die Parameter ja dieselben wie beim Abspeichern sind. Nach der Frage 'TAPE READY ?' erscheint in der untersten Bildschirmzeile ein Sternchen, das zu 'flackern' beginnt, sobald Signale vom Band kommen. Während das Sternchen 'blinkt', kann man den Ladevorgang jederzeit durch die 'BREAK'-Taste abbrechen und zurück zum Menü gelangen. Als Abschluß der Verify-Routine listet das Programm die Daten des 'geladenen' Programms und gibt eine Lade-/Prüfmeldung neben dem Sternchen aus:

- #* Fehlerfrei
- B* Blockfehler
- >* Fehler bei Verify
- F* Prüfsummenfehler

Beim Load-Befehl 'L' fragt das Programm nach der Eingabe des Namens noch zusätzlich nach einer neuen Startadresse. Somit kann man ein Programm an beliebiger Stelle im Speicher ablegen. Bei der Namenseingabe erlaubt das Programm die Verwendung von 'Jokern'; das Zeichen '*' steht für einen beliebig langen Namen mit beliebigen Buchstaben. Die einzige Ausnahme hiervon bilden BASIC-Programme: Sie sollten immer mit der Typ-Angabe 'BAS' geladen werden, da nur dann die Zeiger für den späteren Start des BASIC-Programms richtig gesetzt werden. Die Zeiger 'stehen' zwar richtig, wenn man ein mit dem DISK-BASIC erstelltes Programm wieder in diesem BASIC starten beziehungsweise listen will; sie stimmen aber nicht, wenn ein ROM-BASIC-Programm unter DISK-BASIC laufen soll (oder umgekehrt).

Nach dem erfolgreichen Laden eines BASIC-Programms verzweigt das SuperTape-Programm in das BASIC: Hat man SuperTape aus dem ROM-BASIC gestartet, springt das Programm auch in dieses BASIC. Arbeitet man mit Disketten und hat SuperTape nicht aus dem DISK-BASIC aufgerufen, erscheint kurz die Fehlermeldung 'KEIN BASIC' und anschließend das Menü. Soll ein BASIC-Programm nach dem Laden ausgeführt werden, ist folgendermaßen vorzugehen:

Vom DOS aus mit dem Befehl 'LOAD SUTRS/CMD' SuperTape laden.

Anschließend muß man BASIC aufrufen.

Im BASIC den Befehl 'SYSTEM' eingeben und auf die Meldung '?*' mit '/47360' antworten — SuperTape wird gestartet.

Einen direkten Rücksprung ins BASIC oder DOS kann man durch die Eingabe von 'E' im SuperTape-Programm erreichen. Arbeitet man mit Disketten und hatte vor dem Aufruf von SuperTape BASIC geladen, führt 'E' direkt ins DISK-BASIC. Ohne geladenes BASIC gelangt man durch die Eingabe von 'E' in die Befehlsebene vom DOS, beim Arbeiten im ROM-BASIC erfolgt ein BASIC-Warmstart.

Die Unterscheidung zwischen diesen drei 'Betriebssystemen' erfolgt selbsttätig durch das Programm. Dazu wertet es ein Byte aus dem 'Keyboard DCB' (Zeiger auf die Tastatur-Treiberroutine) aus.

Beim TRS-80, Modell 1 mit dem 'Level 2 ROM-BASIC' steht in dieser Speicherzelle (4017h) der Wert 03 beim aktivierten ROM-BASIC, beim Betrieb des TRS-80 unter NEWDOS 40 erwartet das Programm das Byte 43h in der Speicherzelle. Ein Sprung in das DISK-BASIC erfolgt dann, wenn die zuvor genannten Werte nicht erkannt werden konnten.

Interface

Die Adreßdekodierung des Interfaces besteht aus IC1 und IC2. Die ausgewerteten Adressen schalten entweder ein Flipflop (IC3a) oder erlauben Ein-/Ausgaben über den Puffer IC4. Das Flipflop IC3a schaltet bei jedem Zugriff auf den Port 7Fh (127d) um, unabhängig von den ausgegebenen Daten. Über die drei Inverter (IC5a...IC5c) und das Filter (R2, R3, C1, C2) gelangt das Ausgangssignal des Flipflops an den Kassettenrecorder.

Die Datenleitungen D0, D1, D6 und D7 des TRS-80 liegen über den Puffer IC4 an IC3a und IC3b. Der logische Pegel der Datenleitung D0 gelangt über die Brücke J1 an den 'Clear'-Eingang von IC3a. Legt man diesen Eingang kurzfristig auf logisch 0, schaltet der Ausgang von IC3a auf logisch 1. Diese Funktion benötigt man aber nur dann, wenn mehr als zwei Rechner mit einer 'SuperTape-Ringleitung' verbunden werden sollen. Im Normalfall kann man die Brücke 1 offenlassen und an der Leiste X3 den Impuls von D0 abgreifen.

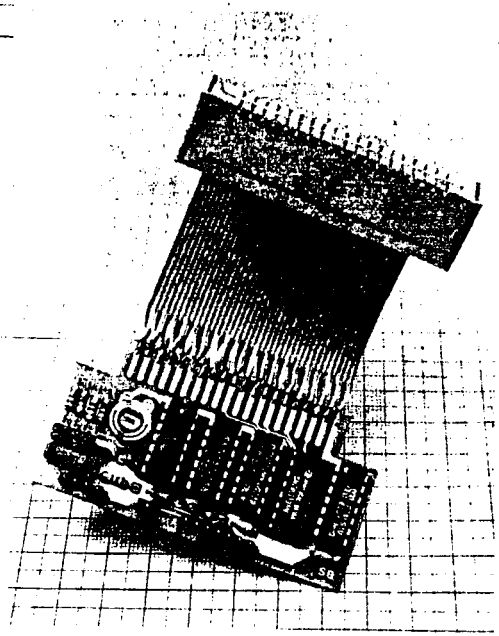
Mit dem Datenbit D1 kann man das Flipflop IC3b steuern. So setzt zum Beispiel der BASIC-Befehl 'OUT 126,2' den Ausgang von IC3b auf logisch 1. Schließt man an diesen Ausgang eine Treiberstufe und ein Relais an, kann man den Kassettenrecorder programmgesteuert ein- und ausschalten.

Das Signal vom Kassettenrecorder gelangt über die Triggerschaltung (IC5d, IC5e) an die Datenleitung D7. Der Pegel dieser Leitung entspricht nach einer Portabfrage dem Eingangssignal am Kassetten-Eingang. Die Leitung D6 liegt über den Puffer an der Anschlußleiste X3, Pin 2. Hier kann man ein Eingangssignal mit TTL-Pegel in den Rechner einlesen.

Aufbau

Da die Leiterplatte sehr eng bestückt wird, sollte man beim Aufbau besonders sorgfältig vorgehen. Der Anschluß des Interfaces an den Rechner geschieht über die Steckerleiste X4. Die Platine ist so ausgelegt, daß man eine 40polige Steckerleiste direkt anlöten kann. Soll das Interface an den 'Interface-Bus' (J3) des Tandy-Expansion-Interface angeschlossen werden, hat diese Lösung allerdings den Nachteil, daß die Platine 'in der Luft schwebt'. Beim direkten Anschluß an den TRS-80-Bus kann man der Leiterplatte durch zwei Distanzrollen festen Halt geben.

Die wohl bessere Lösung besteht darin, die Platine über ein Stück Flachbandkabel und einen 'Preßstecker' mit dem Rechner zu verbinden. Allerdings muß das Kabel so kurz wie möglich sein, da sonst der Tandy recht 'eigenwillig' wird.



Das SuperTape-Interface für den TRS-80.

Test

Nach dem Bestücken sollte man die Platine sehr sorgfältig auf Kurzschlüsse und Unterbrechungen untersuchen. Anschließend kann man die Karte auf den Bus des Rechners oder des Expansion-Boards stecken

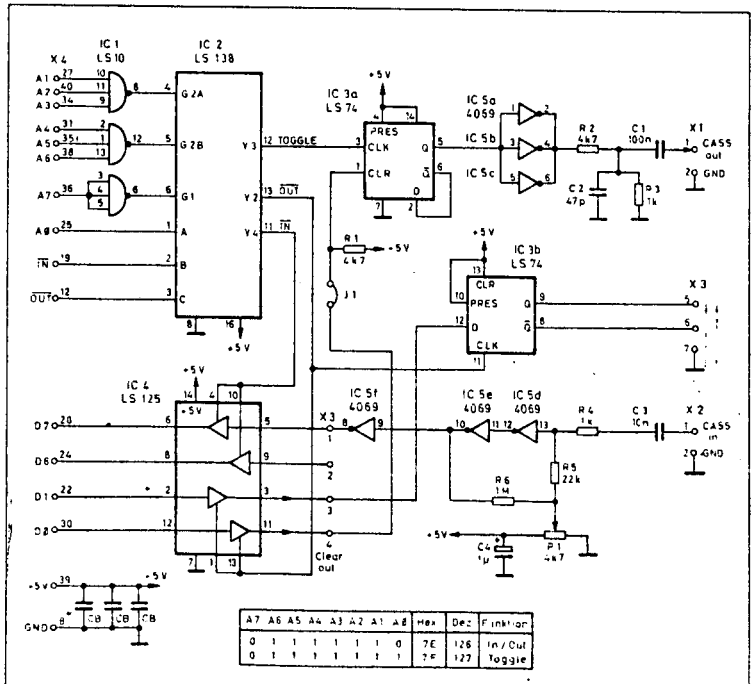


Bild 1. Schaltplan des TRS-80-Interfaces.

und den Computer einschalten: Das System sollte sich wie gewohnt verhalten. Andernfalls ist der Computer sofort abzuschalten und das Interface erneut zu überprüfen.

Tritt dieser Fall nicht ein, können die weiteren Tests des Interfaces mit BASIC-Befehlen und einem Voltmeter durchgeführt werden. Dazu sollte man die Brücke J1 schließen und ein Voltmeter an Pin 2, 4 oder 6 von IC5 anschließen. Nach jeder Ausführung des BASIC-Befehls 'OUT 127,0' muß der Pegel an IC5 wechseln; von circa 0 Volt auf etwa 5 Volt und umgekehrt. Für den nächsten Test sollte die Spannung an IC5 etwa 0 Volt betragen. Mit der Ausführung der Anweisung 'OUT 126,0' muß an IC5 der Pegel auf rund 5 Volt wechseln (Reset-Funktion).

Anschließend ist das Voltmeter an Pin 5 von der Steckerleiste X3 anzuschließen. Der Befehl 'OUT 126,2' bewirkt, daß der Ausgang von IC3b auf logisch 1 (etwa 5 Volt) springt. Mit der Anweisung 'OUT 126,0' ergibt sich eine Spannung von rund 0 Volt an dem Ausgang dieses ICs.

Für die Überprüfung des Kassetten-Eingangs gibt man folgendes Programm ein:

```
10 A = INP(126)
20 PRINT A;
30 GOTO 10
```

Nach dem Start des Programms wird ein Wert angezeigt, der sich ändern muß, wenn man P1 an den rechten oder linken Anschlag dreht. Die richtige Einstellung von P1 liegt an dem Punkt, wo die Anzeige zwischen den beiden Werten springt. □

Stückliste

Widerstände

R1,2	4k7
R3,4	1k
R5	22k
R6	1M
P1	4k7 Trimmer, liegend

Kondensatoren

C1	100n
C2	47p
C3	10n
C4	1µ, Tantal
CB	3 Stützkondensatoren, 100n, ker.

Halbleiter

IC1	74LS10
IC2	74LS138
IC3	74LS74
IC4	74LS125
IC5	CD 4069

Sonstiges

X1, X2, J1	Pfostenleiste, 2polig
X3	Pfostenleiste, 7polig
X4	40poliger 'Card-Edge'-Stecker, 2reihig, Raster 2,54 mm

Brückenstecker für Pfostenleiste; IC-Fassungen: 4 x 14polig, 1 x 16polig; Platine 'TRS-80 SuperTape'.

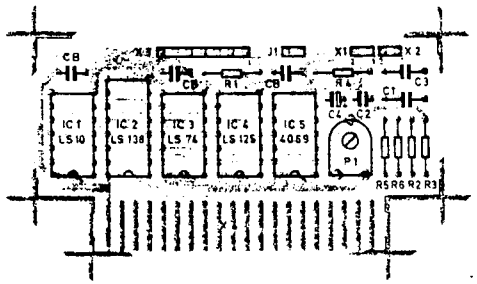


Bild 2. Der Bestückungsplan. Da die ICs sehr dicht beieinander liegen, müssen 'anreihbare' Fassungen verwendet werden.

```
8980                    ORG     8898BH
0833            DISPL   EQU     0833H
0449            RDTAS   EQU     0449H
0044            READY   EQU     0044H
01C9            CLS     EQU     01C9H
401A            CURSYM  EQU     401AH
40A4            HEADP   EQU     40A4H
40F9            TAILP   EQU     40F9H
40F8            WARNF   EQU     40F8H
40FD            WARND   EQU     40FDH
3FF8            BILDP1  EQU     03FF8H
3FF1            BILDP2  EQU     03FF1H
3840            ZTAST   EQU     03840H
0088            LCON36  EQU     88H
007E            PORTIN  EQU     07EH
007F            PORTOU  EQU     07FH
40CC            ROMBAS  EQU     040CCH
40CC            DOSBAS  EQU     040CCH
402D            DOS     EQU     402DH
FFFF            STACK   EQU     0FFFFH
4017            ECHEC   EQU     04017H
                ;
                ;
                ;*****************************************************
                ;SUPERTAPE TANDY TRS-80, MODELL I
                ;MENUEAUSWAHL
                ;
                ;*****************************************************
                ;
                ;
8980 31FFFF     START:  LD      SP,STACK
8983 CD2B69     START1: CALL    CLSRE           ;CLEAR BILDSCHIRM
8986 2150BD     START2: LD      HL,TEXT1        ;TEXT
8989 CDEFB9             CALL    PRTEXT          ;ZUM
898C CD4980     TAST:   CALL    RDTAS           ;BILDSCHIRM
898F 4F                 LD      C,A
8990 E65F               AND     5FH
8992 FE53               CP      053H
8994 2808               JR      Z,SAVE          ;'S'
8996 FE51               CP      51H             ;'Q'
8998 2864               JR      Z,SAVE
899A FE4C               CP      04CH
899C CA98BA             JP      Z,LOAD          ;'L'
899F FE56               CP      056H
89A1 CAF9B9             JP      Z,VERI          ;'V'
89A4 FE45               CP      045H            ;'E'
89A6 CA19BD             JP      Z,BWARM         ;ZUM BASIC ODER DOS
89A9 18E1               JR      TAST            ;SONST ZUR
                ;                               ;TASTATURABFRAGE
                ;*****************************************************
                ;BILDSCHIRM LOESCHEN UND CURSOR SETZEN
                ;*****************************************************
89AB CDC981     CLSRE:  CALL    CLS             ;CLEAR  SCREEN
89AE 3E67               LD      A,067H          ;NEUES CURSOR-
89B0 321A40             LD      (CURSYM),A      ;SYMBOL
89B3 3E0E               LD      A,00EH          ;CURSOR
89B5 CD3300             CALL    DISPL           ;EIN
89B8 C9                 RET
                ;
                ;*****************************************************
                ;TASTATURABFRAGE FUER LD NAME/TYP
                ;*****************************************************
89B9 7E         RONIT:  LD      A,(HL)
89BA CB49               BIT     1,C             ;GROSS ODER KLEIN?
89BC 200C               JR      NZ,RDCONT
89BE CD4980             CALL    RDTAS           ;TASTATURABFRAGE
89C1 FE0D               CP      00DH            ;RETURN ?
89C3 280C               JR      Z,RETUR         ;WENN JA SPRUNG
89C5 FE08               CP      008H            ;CURSOR -
89C7 2812               JR      Z,CURS          ;WENN JA SPRUNG
89C9 77                 LD      (HL),A          ;ZEICHEN ABLEGEN
89CA CD3300     RDCONT: CALL    DISPL           ;UND ZUM BILD
89CD 23                 INC     HL
89CE 10E9               DJNZ    RONIT           ;WEITER BIS B=0
89D0 C9                 RET
                ;*****************************************************
                ; RETURN WURDE GEDRUECKT
                ;*****************************************************
89D1 3E20       RETUR:  LD      A,020H          ;FUELLE REST MIT
89D3 77                 LD      (HL),A          ;SPACE AUF
89D4 CD3300             CALL    DISPL
89D7 23                 INC     HL
89D8 10F7               DJNZ    RETUR
89DA C9                 RET
                ;*****************************************************
                ; CURSOR ZURUECKSETZEN
                ;*****************************************************
89DB 2B         CURS:   DEC     HL              ;
89DC 2B                 DEC     HL              ;CURSOR
89DD 04                 INC     B               ;EIN ZEICHEN
89DE 04                 INC     B               ;ZURUECKSETZEN
89DF 3E18               LD      A,018H
89E1 18E7               JR      RDCONT
                ;*****************************************************
                ;LADE HEXADRESSE NACH HL
                ;*****************************************************
89E3 CD98BA     HEXADR: CALL    ASIBYT          ;LAEDT 4STELLIGE
89E6 67                 LD      H,A             ;HEXADRESSE UEBER
89E7 CD98BA             CALL    ASIBYT          ;ASCII-HEX-HANDLER
89EA 6F                 LD      L,A             ;VOM TASTATUR
89EB C9                 RET                     ;NACH HL
                ;*****************************************************
                ;LADE NAME UND TYP
                ;*****************************************************
89EC 212B8D     NAMTY:  LD      HL,SVNAME       ;NAME UND TYP
89EF 0E0C               LD      B,00CH          ;WERDEN IN DEN
89F1 CD3980             CALL    RONIT           ;SV-PUFFER
```

```
89F4 3E2E               LD      A,02EH          ;GELADEN
89F6 77                 LD      (HL),A
89F7 23                 INC     HL
89F8 0603               LD      B,003H
89FA CD3989             CALL    RONIT
89FD C9                 RET
                ;*****************************************************
                ;SAVE
                ;*****************************************************
89FE F5         SAVE:   PUSH    AF              ;RETTE BEFEHL
89FF 2182BD             LD      HL,TEXT2        ;TEXT
8A02 CDEFB9             CALL    PRTEXT          ;ZUM BILD
8A05 CD4CB9             CALL    NAITY           ;LADE BLOCKPARAMETER
8A08 3A38BD             LD      A,(SVTYP)
8A0B FE42               CP      042H            ;WENN TYP=B(BASIC)
8A0D 2832               JR      Z,TIBAS         ;HOLE BASIC-POINTER
8A0F CB49               BIT     1,C             ;ALTE WERTE?
8A11 203E               JR      NZ,SAVEA
8A13 21C9BD             LD      HL,TEXT3        ;
8A16 CDEFB9             CALL    PRTEXT          ;LADE STARTADRESSE
8A19 CD63B9             CALL    HEXADR          ;IN SV-PUFFER
8A1C 223CBD             LD      (SVSTA),HL      ;
8A1F E5                 PUSH    HL              ;
8A20 3E20               LD      A,020H          ;EIN SPACE ZUM
8A22 CD3380             CALL    DISPL           ;BILD
8A25 CD63B9             CALL    HEXADR          ;LADE ENDADRESSE
8A28 C1                 POP     BC              ;UND
8A29 ED42       DTLEN:  SBC     HL,BC           ;ERRECHNE LAENGE
8A2B 223EBD             LD      (SVLEN),HL      ;FUER PUFFER
8A2E 3E00       STFLA:  LD      A,000H          ;
8A30 213BBD             LD      HL,SVFLAG       ;BAUDRATE SETZEN
8A33 77                 LD      (HL),A          ;FLAG 00 SETZEN
8A34 F1                 POP     AF              ;HOLE BEFEHL
8A35 FE51               CP      51H             ;
8A37 2002               JR      NZ,NOQ          ;NEIN: BAUDRATE 0
8A39 3688               LD      (HL),088H       ;SONST 88H
8A3B CD0B89     NOQ     CALL    TAPERY          ;TAPE READY?
8A3E C345BB             JP      SUSUP           ;DANN SAVE!#
                ;*****************************************************
                ; TYP BASIC
                ;*****************************************************
8A41 2AA440     TIBAS:  LD      HL,(HEADP)      ;FALLS TYP =B
8A44 223CBD             LD      (SVSTA),HL      ;WERDEN ADRESSEN
8A47 E5                 PUSH    HL              ;FUER START/LAENGE
8A48 ED4BF940           LD      BC,(TAILP)      ;VOM HEADY/TAILPOINTER
8A4C C5                 PUSH    BC              ;ABGELEITET UND
8A4D E1                 POP     HL              ;AUTOMATISCH IN DEN
8A4E C1                 POP     BC              ;SV-PUFFER GESETZT
8A4F 18D8               JR      DTLEN
                ;
                ;*****************************************************
                ;TAPE READY ?
                ;*****************************************************
8A51 21D6BD     TAPERY: LD      HL,TEXT4        ;HOLE TEXTADRESSE
8A54 CDEFB9             CALL    PRTEXT          ;TEXT ZUM BILD
8A57 CD4980             CALL    RDTAS           ;TASTATURABFRAGE
8A5A FE4E               CP      04EH            ;'N' NEIN: BREAK
8A5C CA08B9             JP      Z,START         ;DANN ZUM MENUE
8A5F 3E0F               LD      A,00FH          ;SONST RET
8A61 CD3300             CALL    DISPL
8A64 C9                 RET
                ;*****************************************************
                ;MENUEGENERATOR (HL) ZUM BILD
                ;*****************************************************
8A65 7E         PRTEXT: LD      A,(HL)          ;PRINT WAS IN (HL)
8A66 FE23               CP      023H            ;STEHT ZUM BILD
8A68 C8                 RET     Z               ;BIS ENDMARKE
8A69 CD3300             CALL    DISPL           ;ERPEICHT
8A6C 23                 INC     HL
8A6D 18F6               JR      PRTEXT
                ;*****************************************************
                ;VERIFY
                ;*****************************************************
8A6F 21E6BD     VERI:   LD      HL,TEXT5        ;TEXT
8A72 CDEFB9             CALL    PRTEXT          ;ZUM BILD
8A75 CD6CB9             CALL    NAITY           ;HOLE BLOCKPARAMETER
8A78 3E02       VECONT: LD      A,002H          ;SETZE 02 IN
8A7A 322A8D             LD      (ZFLAG),A       ;ZFLAG
8A7D 1833               JR      TPRDY           ;TAPE READY
                ;
                ;*****************************************************
                ;LADEAUFRUF LOAD
                ;*****************************************************
8A7F 21FEBD     LOAD:   LD      HL,TEXT6        ;TEST 'LOAD' USW.
8A82 CDEFB9             CALL    PRTEXT          ;ZUM BILD
8A85 CD6CB9             CALL    NAITY           ;NAME + TYP EINGEBEN
8A88 3A38BD             LD      A,(SVTYP)
8A8B FE42               CP      42H             ; B'?
8A8D 2808               JR      NZ,LOHEX
8A8F 2AA440             LD      HL,(HEADP)
8A92 223CBD             LD      (SVSTA),HL
8A95 212A8D             LD      HL,ZFLAG
8A98 CBD6               SET     2,(HL)
8A9A 1816               JR      TPRDY
                ;
8A9C 21158E     LOHEX:  LD      HL,TEXT7        ;TEXT
8A9F CDEFB9             CALL    PRTEXT          ; NEUE STARTADRESSE'
8AA2 CB49               BIT     1,C             ;ALTE WERTE'
8AA4 2032               JR      NZ,LOADA
8AA6 CD4980             CALL    RDTAS           ;TASTATURABFRAGE
8AA9 FE4A               CP      04AH            ;'J'?
8AAB 2818               JR      Z,LHEX          ;DANN ZUR EINGABE
8AAD 3E00               LD      A,000H          ;ZFLAG AUF NULL
8AAF 322A8D     LDRDY:  LD      (ZFLAG),A       ;
```

```
8AB2 CD0B89     TPRDY:  CALL    TAPERY          ; TAPE READY
8AB5 3E2A               LD      A,02AH          ;STERNCHEN
8AB7 32F13F             LD      (BILDP2),A      ;SETZEN
8ABA CD0B88             CALL    LDSUP           ;LADEN
8ABD 3229BD             LD      (ERROR),A       ;FEHLERMELDUNG RETTEN
8AC0 CDE28A             CALL    LILPU           ;LD-PUFFER AUSGEBEN
8AC3 1828               JR      LDCHK           ;ZUM LADECHECK
                ;----------NEUE STARTADRESSE EINGEBEN
8AC5 2126BE     NADR:   LD      HL,TEXT8        ;TEXT 'ADRESSE'
8AC8 CDEFB9             CALL    PRTEXT          ;ZUM BILD
8ACB CD63B9             CALL    HEXADR          ;LADE ADRESSE
8ACE 223CBD             LD      (SVSTA),HL      ;IN SV-PUFFER
8AD1 3E04               LD      A,004H          ;SETZE FLAG 04
8AD3 322ABD             LD      (ZFLAG),A       ;IN ZFLAG
8AD6 180A               JR      TPRDY           ;UND WEITER
                ;----------LADEAUFRUF "L" OHNE BLOCKEINGABE
8AD8 3A2ABD     LOADA:  LD      A,(ZFLAG)
8ADB CB57               BIT     2,A
8ADD 288A               JR      Z,LADR
8ADF 213CBD             LD      HL,SVSTA+1
8AE2 0601               LD      B,001
8AE4 CD0488             CALL    LILPZ
8AE7 18C9               JR      TPRDY
8AE9 3E2A       LADR:   LD      A,02AH
8AEB CD3380             CALL    DISPL
8AEE 1880               JR      LDRDY
                ;*****************************************************
                ;LDCHECK:AUSWERTUNG LOAD/VERIFY
                ;*****************************************************
8AF0 3A29BD     LDCHK:  LD      A,(ERROR)       ;FEHLERMELDUNG
8AF3 32F03F             LD      (BILDP1),A      ;ABLEGEN
8AF6 FE42               CP      042H            ; B' ? =BLOCKFEHLER
8AF8 2808               JR      Z,TPRDY         ;ZURUECK ZUR AUFNAHME
8AFA FE23               CP      023H            ;'#' LADE ZFLAG
8AFC 2803               JR      Z,LOOK
8AFE C308B9     RUECK:  JP      START2          ;ZUM MENUE
                ;
8B01 3A2ABD     LOOK:   LD      A,(ZFLAG)       ;LADE ZFLAG
8B04 FE02               CP      002H            ;VERIFY?
8B06 28F6               JR      Z,RUECK         ;DANN ZUM MENUE
8B08 3A51BD             LD      A,(LDTYP)       ;TYP LADEN
8B0B FE42               CP      042H            ;B(BASIC) ?
8B0D 28EF               JR      NZ,RUECK        ;JA ? DANN WEITER
8B0F 18B8               JR      STBAS           ;UND BASIC-POINTER
                ;                               ;SETZEN
                ;*****************************************************
                ;ASCII/HEX WANDLER
                ;*****************************************************
8B11 CD98BA     ASIBYT: CALL    ASINIB          ;
8B14 87                 RLCA                    ;ROUTINE WANDELT
8B15 87                 RLCA                    ;ZWEI ASCII-ZEICHEN
8B16 87                 RLCA                    ;IN EINE ZWEISTELLIGE
8B17 87                 RLCA                    ;HEX-ZAHL
8B18 4F                 LD      C,A             ;
8B19 CD98BA             CALL    ASINIB          ;
8B1C B1                 OR      C               ;ERGEBNIS IN A
8B1D C9                 RET
                ;
8B1E CD4980     ASINIB: CALL    RDTAS           ;EIN ZEICHEN WIRD
8B21 F5                 PUSH    AF              ;VON DER TASTATUR
8B22 CD3380             CALL    DISPL           ;GELESEN UND ALS
8B25 F1                 POP     AF              ;EIN HEX-NIBBLE
8B26 D630               SUB     030H            ;INTERPRETIERT
8B28 D8                 RET     C
8B29 C6E9               ADD     A,0E9H
8B2B D8                 RET     C
8B2C C606               ADD     A,006H
8B2E F28E8A             JP      P,ASICON
8B31 C607               ADD     A,007H
8B33 D8                 RET     C
8B34 C630       ASICON: ADD     A,030H
8B36 87                 OR      A
8B37 C9                 RET
                ;
8B38 7E         BYTIAS: LD      A,(HL)          ;
8B39 F5                 PUSH    AF              ;ROUTINE WANDELT
8B3A D9                 EXX                     ;ZWEISTELLIGE
8B3B 0F                 RRCA                    ;HEXZAHL
8B3C 0F                 RRCA                    ;IN ZWEI ASCII-
8B3D 0F                 RRCA                    ;ZEICHEN UND
8B3E 0F                 RRCA                    ;GIBT DIESE
8B3F E60F               AND     00FH            ;ZUM BILD
8B41 CD0B8A             CALL    NIBIAS          ;AUS
8B44 F1                 POP     AF
8B45 E60F               AND     00FH
8B47 CD0B8A             CALL    NIBIAS
8B4A D9                 EXX                     ;
8B4B C9                 RET                     ;
                ;
8B4C C690       NIBIAS: ADD     A,090H          ;BERECHNET ASCII
8B4E 27                 DAA                     ;AUS NIBBLE UND
8B4F CE40               ADC     A,040H          ;BRINGT DAS ZEICHEN
8B51 27                 DAA                     ;AUF DEN BILDSCHIRM
8B52 F5                 PUSH    AF
8B53 CD3380             CALL    DISPL
8B56 F1                 POP     AF
8B57 C9                 RET
                ;*****************************************************
                ;LIST PUFFER
                ;*****************************************************
8B58 3E80       LILPU:  LD      A,080H          ;CP
8B5A CD3380             CALL    DISPL           ;ZUM BILD
8B5D 2144BD             LD      HL,LDNAME       ;LADE ADR AUS LD-PUFFER
8B60 0610               LD      B,010H          ;LAENGE 10H FUER
8B62 7E         LILP1:  LD      A,(HL)          ;NAME UND TYP
8B63 CD3380             CALL    DISPL           ;ZUM BILD
8B66 23                 INC     HL              ;WEITER
8B67 10F9               DJNZ    LILP1           ;BIS B=0
```

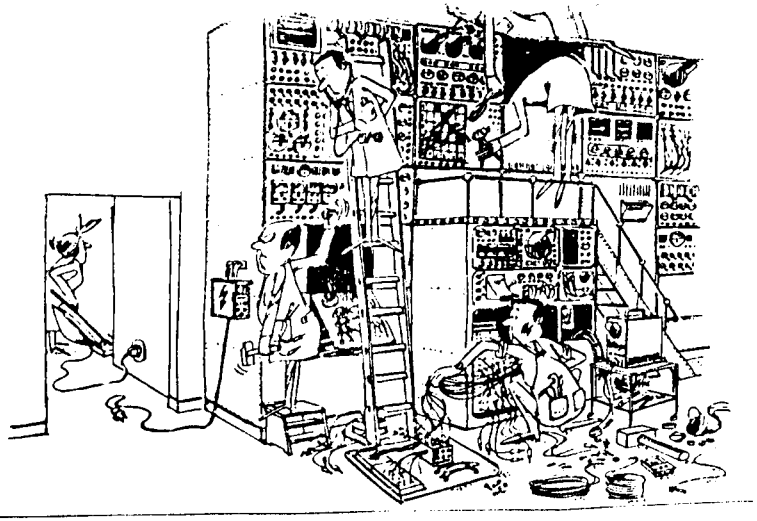
```
8AF3 2A558D         LD      HL,(LDSTA)      ;LADE STARTADR
8AF6 ED4B578D       LD      BC,(LDLEN)      ;UND LAENGE
8AFA ED4A           ADC     HL,BC           ;ADDIEREN
8AFC 22598D         LD      (LDFREI),HL     ;END-ADR ABLEGEN
8AFF 2A5A8D         LD      HL,LDSTH        ;ZEIGE AUF ADRESSEN
8B02 0603           LD      B,03H           ;
8B04 3E28   LILP2:  LD      A,028H          ;AUSGABESCHLEIFE
8B06 CD3300         CALL    DISPL           ;FUER STARTADRESSE
8B09 CDC28A         CALL    BYTIAS          ;LAENGE
8B0C 2B             DEC     HL              ;UND ENDADRESSE
8B0D CDC28A         CALL    BYTIAS          ;
8B10 23             INC     HL              ;
8B11 23             INC     HL              ;
8B12 23             INC     HL              ;
8B13 10EF           DJNZ    LILP2           ;
8B15 C9             RET                     ;
                    ;*****************************************************
                    ; SET BASIC-POINTER
                    ;*****************************************************
8B16 3A1740 STBAS:  LD      A,(ECHEC)       ;TEST OB BASIC
8B19 FE43           CP      43H             ;GELADEN IST
8B1B 2817           JR      Z,NOBAS         ;Z=NEIN
8B1D 2131BE         LD      HL,TEXT9        ;TEXT 'START BASIC'
8B20 CDEFB9         CALL    PRTEXT          ;AUSGEBEN
8B23 012900         LD      BC,WARM         ;
8B26 C3AE19         JP      19AEH           ;
8B29 2A558D WARM:   LD      HL,(LDSTA)      ;
8B2C ED5B578D       LD      DE,(LDLEN)      ;
8B30 19             ADD     HL,DE           ;
8B31 C3772C         JP      2C77H           ;
                    ;
                    ;KEIN BASIC GELADEN
8B34 213F8E NOBAS:  LD      HL,TEXT10       ;TEXT 'KEIN BASIC'
8B37 CDEFB9         CALL    PRTEXT          ;AUSGEBEN
8B3A 010000         LD      BC,0000         ;
8B3D 0B     NOBALO: DEC     BC              ;
8B3E 78             LD      A,B             ;
8B3F B1             OR      C               ;
8B40 20FB           JR      NZ,NOBALO       ;
8B42 C300B9         JP      START           ;
                    ;*****************************************************
                    ;SVSUP
                    ;*****************************************************
8B45 F3     SVSUP:  DI                      ;INTERRUPTS ABSCHALTEN
8B46 212B8D         LD      HL,SVNAME       ;SENDET PARAMETERBLOCK
8B49 011900         LD      BC,00019H       ;
8B4C 3E2A           LD      A,02AH          ;
8B4E 1600           LD      D,0             ;FUER 3600 BAUD
8B50 CD6A0B         CALL    SVBLO           ;SENDET DATENBLOCK
8B53 2A3C8D         LD      HL,(SVSTA)      ;
8B56 ED4B3E8D       LD      BC,(SVLEN)      ;
8B5A 3A388D         LD      A,(SVFLAG)      ;HOLE BAUDRATE
8B5D 57             LD      D,A             ;NACH  D
8B5E 3EC5           LD      A,0C5H          ;
8B60 CD6A0B         CALL    SVBLO           ;
8B63 C3B3B9         JP      START2          ;
                    ;*****************************************************
                    ;SVBLO
                    ;*****************************************************
8B66 F5     SVBLO:  PUSH    AF              ;RETTE ZUM STACK
8B67 0B             DEC     BC              ;DECR. BLOCKLNGE
8B68 C5             PUSH    BC              ;UND RETTE ZUM STACK
8B69 0648           LD      B,048H          ;SENDE 64 SYNC-ZEICHEN
8B6B 1E16   SVL01:  LD      E,016H          ;LADE  E MIT SYNC-BYTE
8B6D A7             DEFB    0A7H            ;WARTE 13 TAKTE
8B6E A7             DEFB    0A7H            ;
8B6F D8             DEFB    0D8H            ;
8B70 CD A2 8B       CALL    SVBIT           ;
8B73 10F6           DJNZ    SVL01           ;
8B75 C1             POP     BC              ;HOLE LAENGE VOM STACK
8B76 F1             POP     AF              ;HOLE SYNC-ABSCHLUSS
8B77 5F             LD      E,A             ;SENDE SYNC-ABSCHLUSS
8B78 CDA28B         CALL    SVBYT           ;
8B7B 00210000       LD      IX,00000H       ;INIT PRUEFSUMME
8B7F A6             DEFB    0A6H            ;WARTE 12 TAKTE
8B80 D8             DEFB    0D8H            ;
8B81 5E     SVL02:  LD      E,(HL)          ;SENDE BYTE
8B82 CDA28B         CALL    SVBYT           ;
8B85 ED41           CPI                     ;INC HL,DEC BC
8B87 EA8198         JP      PE,SVL02        ;SCHLEIFE BIS BC=0
8B8A 5E             LD      E,(HL)          ;SENDE LETZTES BYTE
8B8B CDA28B         CALL    SVBYT           ;
8B8E C0             DEFB    0C0H            ;WARTE 5 TAKTE
8B8F DDE5           PUSH    IX              ;SENDE PRUEFSUMME
8B91 C1             POP     BC              ;
8B92 59             LD      E,C             ;
8B93 CDA28B         CALL    SVBYT           ;
8B96 E5             DEFB    0E5H            ;WARTE 29 TAKTE
8B97 E1             DEFB    0E1H            ;
8B98 A7             DEFB    0A7H            ;
8B99 A7             DEFB    0A7H            ;
8B9A 78             LD      E,B             ;
8B9B CDA28B         CALL    SVBYT           ;
8B9E E3             DEFB    0E3H            ;WARTE 56 TAKTE UND
8B9F E3             DEFB    0E3H            ;SENDE SCHLUSSBYTE
8BA0 A6             DEFB    0A6H            ;
8BA1 D8             DEFB    0D8H            ;
                    ;*****************************************************
                    ;SVBYTE: SAVE B.TE
                    ;*****************************************************
8BA2 37     SVBYT:  SCF                     ;SETZE ENDFLAG
8BA3 CB18   SVBIT:  RR      E               ;AUSGABEBIT IN CARRY
8BA5 3802           JR      C,SVEINS        ;BIT =1? SPRUNG
8BA7 D37F           OUT     (07FH),A        ;UND AUSGEBEN
8BA9 A6             DEFB    0A6H            ;WARTE  14 TAKTE
8BAA A6             DEFB    0A6H            ;
8BAB CB7A   SVCONT: BIT     7,D             ;
```

```
8BAD CCD3BB         CALL    Z,SVWAIT        ;WARTE FUER 3600 BAUD
8BB0 E3             DEFB    0E3H            ;WARTE WEITERE 86 TAKTE
8BB1 E3             DEFB    0E3H            ;
8BB2 E3             DEFB    0E3H            ;
8BB3 E3             DEFB    0E3H            ;
8BB4 A7             DEFB    0A7H            ;
8BB5 D37F           OUT     (07FH),A        ;AUSGABE ZUM PORT
8BB7 CB7A           BIT     7,D             ;
8BB9 CCD3BB         CALL    Z,SVWAIT        ;WARTE FUER 3600BD
8BBC D8             DEFB    0D8H            ;WARTE 14 TAKTE
8BBD D8             DEFB    0D8H            ;
8BBE 1D             DEC     E               ;8 BITS AUSGEGEBEN?
8BBF C8             RET     Z               ;RETURN FALLS JA
8BC0 1C             INC     E               ;SONST RESTAURIERE E
8BC1 3A4030         LD      A,(ZTAST)       ;WARTE 48 TAKTE
8BC4 FE04           CP      04H             ;UND TESTE BREAK
8BC6 CA00B9         JP      Z,START         ;
8BC9 C8             DEFB    0C8H            ;
8BCA C8             DEFB    0C8H            ;
8BCB AF             XOR     A               ;LOESCHE CARRY
8BCC 18D5           JR      SVBIT           ;WEITER
                    ;
8BCE 0023   SVEINS: INC     IX              ;INKR. PRUEFSUMME
8BD0 C3ABBB         JP      SVCONT          ;
                    ;*****************************************************
                    ; SVWAIT WARTET 117 TAKTE FUER 3600 BAUD
                    ;*****************************************************
8BD3 3E06   SVWAIT: LD      A,006           ;WARTE 99 TAKTE
8BD5 3D     WTCON:  DEC     A               ;
8BD6 20FD           JR      NZ,WTCON        ;
8BD8 A7             DEFB    0A7H            ;WEITERE 9 TAKTE
8BD9 D8             DEFB    0D8H            ;
8BDA C9             RET                     ;
                    ;*****************************************************
                    ;LDSUP:LADEROUTINE
                    ;*****************************************************
8BDB 215C8D LDSUP:  LD      HL,LDEND        ;
8BDE 3619           LD      (HL),019H       ;
8BE0 3E20           LD      A,020H          ;
8BE2 77     CLCON:  LD      (HL),A          ;LOESCHE LD-PUFFER
8BE3 2B             DEC     HL              ;
8BE4 10FC           DJNZ    CLCON           ;
8BE6 23             INC     HL              ;HL=LDNAM
8BE7 011900         LD      BC,00019H       ;BLOCKLAENGE 19H
8BEA 3E2A           LD      A,02AH          ;
8BEC DD21118C       LD      IX,LDWAIT       ;BAUDRATE FUER 3600BD
8BF0 CD38BC         CALL    LDBLOC          ;LADE BLOCKPARAMETER
8BF3 C8             RET     NZ              ;RET BEI FEHLER
8BF4 1144BD         LD      DE,LDNAME       ;VERGLEICHE NAMEN
8BF7 212B8D         LD      HL,SVNAME       ;
8BFA 0610           LD      B,010H          ;
8BFC 7E     CPCON:  LD      A,(HL)          ;
8BFD FE2A           CP      02AH            ;FALLS * SKIP REST
8BFF 280A           JR      Z,SKNAM         ;
8C01 FE3F           CP      03FH            ;FALLS ? SKIP CHAR
8C03 2805           JR      Z,SKCHA         ;
8C05 1A             LD      A,(DE)          ;
8C06 BE             CP      (HL)            ;
8C07 3E42           LD      A,042H          ;
8C09 C0             RET     NZ              ;FALSCH? DANN RET
8C0A 23     SKCHA:  INC     HL              ;
8C0B 13     SKNAM:  INC     DE              ;
8C0C 3E04           LD      A,004H          ;AB 04 MIT
8C0E B8             CP      B               ;SV-TYP VERGLEICHEN
8C0F 2003           JR      NZ,SVCON        ;
8C11 013B8D         LD      BC,SVTYP        ;
8C14 10E6   SKCON:  DJNZ    CPCON           ;
8C16 2A558D         LD      HL,(LDSTA)      ;PARAMETER FUER DATEN-
8C19 ED4B578D       LD      BC,(LDLEN)      ;BLOCK SETZEN
8C1D 3A548D         LD      A,(LDFLAG)      ;
8C20 17             RLA                     ;
8C21 3004           JR      NC,LXALT        ;
8C23 0021D08C       LD      IX,LDW24        ;BAUDRATE 7200BD
8C27 3A2A8D LXALT   LD      A,(ZFLAG)       ;
8C2A CB57           BIT     2,A             ;NEUE STARTADRESSE
8C2C 2803           JR      Z,SKADR         ;
8C2E 2A3C8D         LD      HL,(SVSTA)      ;
8C31 CB4F   SKADR:  BIT     1,A             ;LADEN ODER VERIFY
8C33 3EC5           LD      A,0C5H          ;
8C35 C2608C         JP      NZ,LOVER        ;
                    ;*****************************************************
                    ;LDBLOC: LAEDT BLOCK
                    ;*****************************************************
8C38 167E   LDBLOC  LD      D,PORTIN        ;
8C3A 09             EXX                     ;
8C3B CD8E8C         CALL    LDSYN           ;SYNCHRONISIERT
8C3E 09             EXX                     ;
8C3F 77             LD      (HL),A          ;BYTE NACH (HL)
8C40 ED41           CPI                     ;INC HL UND DEC BC
8C42 E2508C         JP      PO,LTEST        ;TEST BEI ENDE
8C45 09     LDNEXT: EXX                     ;
8C46 CD6F8C         CALL    LDBYT           ;NAECHSTES BYTE
8C49 09             EXX                     ;
8C4A 77             LD      (HL),A          ;NACH (HL)
8C4B ED41           CPI                     ;UND SCHLEIFE BIS
8C4D EA458C         JP      PE,LDNEXT       ;ENDE
8C50 4A     LTEST:  LD      C,D             ;SETZE PORTADRESSE
8C51 CD6F8C         CALL    LDBYT           ;LADE LD-PRUEFBYTE
8C54 00             NOP                     ;WARTE 12 TAKTE
8C55 00             NOP                     ;
8C56 00             NOP                     ;
8C57 47     TEST:   LD      B,A             ;
8C58 E5             DEFB    0E5H            ;WARTE 29 TAKTE
8C59 00             DEFB    00H             ;
8C5A E1             DEFB    0E1H            ;
8C5B CD6F8C         CALL    LDBYT           ;LADE HI-PRUEFBYTE
8C5E 48             LD      C,B             ;GELADENE
```



```
8C5F 47             LD      B,A             ;PRUEFSUMME NACH BC
8C60 D9             EXX                     ;
8C61 E5             PUSH    HL              ;
8C62 D9             EXX                     ;BESTIMMTE
8C63 E1             POP     HL              ;PRUEFSUMME NACH HL
                    ;*****************************************************
                    ;LDTES: PRUEFSUMMENTEST
                    ;*****************************************************
8C64 AF             XOR     A               ;
8C65 ED42           SBC     HL,BC           ;VERGLEICHE HL MIT BC
8C67 3E23           LD      A,023H          ;LADE AKKU MIT '#'
8C69 C8             RET     Z               ;RET FALLS OK
8C6A 3E46           LD      A,046H          ;SONST 'F'
8C6C C9             RET                     ;UND RETURN
                    ;*****************************************************
                    ;LOVER : VERIFY
                    ;*****************************************************
8C6D 167E   LOVER:  LD      D,PORTIN        ;PORTADRESSE
8C6F D9             EXX                     ;
8C70 CDBE8C         CALL    LDSYN           ;SYNCHRONISIERT
8C73 D9             EXX                     ;
8C74 ED41           CPI                     ;VERGLEICH
8C76 2813           JR      NZ,ERRVE        ;ERROR FALLS <>
8C78 E2838C         JP      PO,VTEST        ;SPRUNG FALLS ENDE
8C7B D9     VERNXT: EXX                     ;
8C7C CD6F8C         CALL    LDBYT           ;NAECHST. BYTE
8C7F D9             EXX                     ;
8C80 ED41           CPI                     ;VERGLEICH
8C82 EA788C         JP      PE,VERNXT       ;BEI PE WEITER
8C85 4A     VTEST:  LD      C,D             ;SONST LADE PRUEF-
8C86 CD6F8C         CALL    LDBYT           ;SUMME
8C89 19CC           JR      TEST            ;
8C8B 3E3E   ERRVE:  LD      A,03EH          ;FEHLER BEI VERIFY
8C8D C9             RET                     ;
                    ;*****************************************************
                    ;LDSYN: SYNCHRONISATION
                    ;*****************************************************
8C8E 1E00   LDSYN:  LD      E,000H          ;INIT LADEBYTE
8C90 0E7E           LD      C,PORTIN        ;PORTADRESSE
8C92 F3             DI                      ;INTERRUPTS AUS
8C93 57             LD      D,A             ;C=ABSCHLUSSMARKE
                    ;WARTE AUF EIN SYCHA
8C94 00     SYNEU1: DEFB    00H             ;WARTE 4 TAKTE
8C95 A6     LDNXB:  DEFB    0A6H            ;WARTE  7 TAKTE
8C96 C3978C SYNEU2: JP      SYNEI           ;WARTE  10 TAKTE
8C99 CBC3   SYNEI:  SET     0,E             ;LADE NUR EIN BIT
8C9B CD6B8C         CALL    LDBIT           ;
8C9E FE16           CP      016H            ;SYCHA ERKANNT?
8CA0 5F             LD      E,A             ;
8CA1 20F2           JR      NZ,LDNXB        ;NEIN ? DANN NEXT
                    ;TEST AUF WEITERE SYCHA
8CA3 A7             DEFB    0A7H            ;WARTE 9 TAKTE
8CA4 08             DEFB    08H             ;
8CA5 0604           LD      B,004H          ;ZAEHLER AUF 10
8CA7 A6     SYCON:  DEFB    0A6H            ;WARTE 7 TAKTE
8CA8 CD6F8C         CALL    LDBYT           ;LADE NAECHSTES BYTE
8CAB FE16           CP      016H            ;VERGL. MIT SYCHA
8CAD 20E5           JR      NZ,SYNEU1       ;UNGLEICH ? SYNC NEU
8CAF A6             DEFB    0A6H            ;WARTE 7TAKTE
8CB0 10F5           DJNZ    SYCON           ;WEITER BIS 10=SYCHA
                    ;UEBERLESEN FOLGENDER SYCHA
8CB2 1804           JR      SYEIN           ;
8CB4 A6     SYIGN:  DEFB    0A6H            ;WARTE 22 TAKTE
8CB5 08             DEFB    08H             ;
8CB6 08             DEFB    08H             ;
8CB7 08             DEFB    08H             ;
8CB8 CD6F8C SYEIN:  CALL    LDBYT           ;LADE NAECHSTES BYTE
8CBB FE16           CP      016H            ;VERGL.MIT SYCHA
8CBD 28F5           JR      Z,SYIGN         ;JA, DANN WEITER LADEN
                    ;TEST AUF ABSCHLUSSMARKE
8CBF BA             CP      D               ;ABSCHLUSS RICHTIG?
8CC0 2804           JR      NZ,SYNEU2       ;NEIN SYNCHR. NEU
8CC2 210000         LD      HL,00000H       ;INIT PRUEFSUME
8CC5 A6             DEFB    0A6H            ;WARTE 23 TAKTE
8CC6 A6             DEFB    0A6H            ;
8CC7 A7             DEFB    0A7H            ;
8CC8 D8             DEFB    0D8H            ;
                    ;*****************************************************
                    ;LDBYT: LAEDT EIN BYTE/BIT
                    ;*****************************************************
8CC9 1E80   LDBYT:  LD      E,080H          ;SETZE FLAG FUER 8 BITS
8CCB DDE9   LDBIT:  JP      (IX)            ;WARTE
8CCD 00     LDW36:  NOP                     ;VARIABLE
8CCE 00     LDW32:  NOP                     ;EINSRUNGS-
8CCF 00     LDW28:  NOP                     ;MOEGLICHKEITEN
8CD0 00     LDW24:  NOP                     ;FUER
8CD1 00     LDW20:  NOP                     ;IX
8CD2 00     LDW16:  NOP                     ;
8CD3 00     LDW12:  NOP                     ;
8CD4 00     LDW08:  NOP                     ;
8CD5 00     LDW04:  NOP                     ;
8CD6 ED78   LOSAM:  IN      A,(C)           ;SAMPLE-WERT
8CD8 FAFE8C         JP      M,HIGH          ;
8CDB ED78   LOW:    IN      A,(C)           ;WARTE AUF STEIGENDE
8CDD F2E08C         JP      P,LOW           ;FLANKE
8CE0 AF             XOR     A               ;LOESCHE CARRY
8CE1 08             EX      AF,AF           ;NACH AF' UND HOLE
8CE2 00             NOP                     ;ALTEN CARRY
8CE3 3801           JR      NC,NULL1        ;C=0 DANN 0
8CE5 23             INC     HL              ;SONST INC PRUEFSUMME
8CE6 78     NULL1:  LD      A,E             ;ROTIERE CARRY IN E
8CE7 1F             RRA                     ;
8CE8 D8             RET     C               ;RET BEI ENDFLAG
8CE9 5F             LD      E,A             ;SPEICHER BITS
8CEA 3A1B3F         LD      A,(BILDP2)      ;
8CED EE01           XOR     01H             ;STERNCHEN BLINKT
8CEF 321B3F         LD      (BILDP2),A      ;
```

```
8CF2 3A4038 BRCHC:  LD      A,(ZTAST)       ;ZEICHEN VON TAST.
8CF5 FE04           CP      04H             ;BREAK?
8CF7 00             NOP                     ;
8CF8 C2C88C         JP      NZ,LDBIT        ;LADE NAECHSTES BIT
8CFB C300B9         JP      START           ;
8CFE ED78   HIGH:   IN      A,(C)           ;WARTE AUF FALLENDE
8D00 FAFE8C         JP      M,HIGH          ;FLANKE
8D03 37             SCF                     ;SETZE CARRY
8D04 08             EX      AF,AF'          ;NACH AF' UND HOLE ALTES
8D05 3F             CCF                     ;CARRY UND INVERTIERE
8D06 3001           JR      NC,NULL2        ;C=0 ERGIBT BIT=0
8D08 23             INC     HL              ;INC PRUEFSUMME
8D09 78     NULL2:  LD      A,E             ;ROTIERE CARRY IN E
8D0A 1F             RRA                     ;
8D0B D8             RET     C               ;RETURN BEI ENDFLAG
8D0C 3F             LD      E,A             ;
8D0D E5             DEFB    0E5H            ;WARTE 21 TAKTE
8D0E E1             DEFB    0E1H            ;
8D0F 18E1           JR      BRCHC           ;
                    ;*****************************************************
                    ;LDWAIT: WARTE FUER 3600 BAUD
                    ;*****************************************************
8D11 3E08   LDWAIT: LD      A,LCON36        ;WARTE 172 TAKTE
8D13 3D     LDWCON: DEC     A               ;
8D14 20FD           JR      NZ,LDWCON       ;
8D16 C3D68C         JP      LOSAM           ;
                    ;*****************************************************
                    ;EXIT NACH DOS, DOS-BASIC ODER ROM-BASIC
                    ;*****************************************************
8D19 3A1740 BWARM:  LD      A,(ECHEC)       ;
8D1C FE83           CP      83              ;
8D1E CACC06         JP      Z,ROMBAS        ;INS ROM-BASIC SPRINGEN
8D21 FE43           CP      43H             ;
8D23 CA2040         JP      Z,DOS           ;INS DOS SPRINGEN
8D26 C3CC06         JP      DOSBAS          ;INS DOS-BASIC SPRINGEN
                    ;*****************************************************
                    ;DER PUFFERBEREICH
                    ;*****************************************************
0801        ERROR:  DEFS    1
0801        ZFLAG:  DEFS    1
080C        SVNAME: DEFS    12
0801        SVPUN:  DEFS    1
0803        SVTYP:  DEFS    3
0801        SVFLAG: DEFS    1
0802        SVSTA:  DEFS    2
0802        SVLEN:  DEFS    2
0804        SVFREI: DEFS    4
080C        LDNAME: DEFS    12
0801        LDPUN:  DEFS    1
0803        LDTYP:  DEFS    3
0801        LDFLAG: DEFS    1
0802        LDSTA:  DEFS    2
0804        LDSTH   EQU     LDSTA+1
0802        LDLEN:  DEFS    2
0804        LDFREI: DEFS    4
085C        LDEND   EQU     LDFREI+3
            ;
085D 0D     TEXT1:  DEFB    00DH
085E 53             DEFM    'SUPERTAPE FUER TRS-80, MODELL 1'
087D 0D             DEFB    00DH
087E 80             DEFB    080H
087F 28             DEFM    '(S)AVE (Q)UICK  (L)OAD  (V)ERIFY (E)XIT'
08A6 0D             DEFB    00DH
08A7 0D             DEFB    00DH
08A8 43             DEFM    'COMMAND:'
08B0 0E             DEFB    00EH
08B1 23             DEFB    '#'
08B2 0D     TEXT2:  DEFB    00DH
08B3 53             DEFM    'SAVE'
08B7 0D             DEFB    00DH
08B8 4E             DEFM    'NAME.......TYP'
08C7 0D             DEFB    00DH
08C8 23             DEFB    '#'
08C9 0D     TEXT3:  DEFB    00DH
08CA 53             DEFM    'START ENDE'
08D4 0D             DEFB    00DH
08D5 23             DEFB    '#'
08D6 0D     TEXT4:  DEFB    00DH
08D7 54             DEFM    'TAPE READY ? #'
08E5 0D             DEFB    00DH
08E6 56     TEXT5:  DEFM    'VERIFY'
08EC 0D             DEFB    00DH
08ED 4E             DEFM    'NAME.......TYP'
08FC 0D             DEFB    00DH
08FD 23             DEFB    '#'
08FE 0D     TEXT6:  DEFB    00DH
08FF 4C             DEFM    'LOAD'
0903 0D             DEFB    00DH
0904 4E             DEFM    'NAME.......TYP'
0913 0D             DEFB    00DH
0914 23             DEFB    '#'
0915 0D     TEXT7:  DEFB    00DH
0916 4E             DEFM    'NEUE ADRESSE ? #'
0926 0D     TEXT8:  DEFB    00DH
0927 41             DEFM    'ADRESSE'
092E 0D             DEFB    00DH
092F 23             DEFB    '#'
0930 0D             DEFB    00DH
0931 0D     TEXT9:  DEFB    00DH
0932 42             DEFM    'BASIC-START'
093D 0D             DEFB    00DH
093E 23             DEFB    '#'
093F 0D     TEXT10: DEFB    00DH
0940 4B             DEFM    'KEIN BASIC'
094A 0D             DEFB    00DH
094B 23             DEFB    '#'
0900                END     START
```



»Sofort aufhören! Der Fehler liegt hier!«

Hardware-Tests

Hinweis für Leute, die an der seriellen Schnittstelle von RB-Elektronik interessiert sind.
Diese Schnittstelle kostet bei RB-Elektronik 249,-DM. Die gleiche Schnittstelle habe ich bei der Firma Dr, Auermann in Strassenhaus für 199,-DM gekauft.

Ulrich Böckling

Hardware-Tests -- Hardware-Tests

Privatanzeige

Verkaufe GENIE 1 64K mit SPEEDUP in speziellem Gehäuse
mit externer Tastatur.
1 SHUGART DS/DD 40-Track Laufwerk, RB-Hochauflösende Grafik,
RB-V24-Schnittstelle, Akustikkoppler und grüner Monitor.
Preis VHS
Joachim HERL, Petersenstraße 14, 5000 Köln 91, Tel. 0221/843160

BÖRSE

Kurz noch eine Vorbemerkung der Redaktion zu der Spalte Börse (insbesondere möchte ich noch auf den Fragekasten eingehen).

Die Textabschnitte für diese Seiten stelle ich aus Euren Briefen zusammen. Dazu habe ich zur besseren Ordnung und Übersicht zwei Themen erwählt. Und zwar einmal die Spalte *HER HAT WAS? - HER SUCHT WAS?* und als zweites *FRAGEN, FRAGEN, FRAGENKASTEN*. Ich möchte Euch nun bitten, Eure Infobeiträge zu diesen Themen so zu gestalten, wie Ihr im Moment aus den folgenden Seiten ersehen könnt. Zu jedem Thema getrennt und unterschrieben.

Bezüglich des Fragekasten hätte ich noch an alle CLUB 80-er eine Bitte. Bei eventueller Beantwortung einer Frage kann dies über die immer beiliegende aktuelle Adressenliste erfolgen, dabei wäre es nett wenn Ihr die Lösung auch der INFO zukommen lassen würdet. Manch eine Beantwortung wäre sicher einen Artikel in der INFO wert.

Vielleicht gibt es dann auf diesem Wege mehere Lösungsmöglichkeiten für ein Problem und man kann sich die auswählen, die einem am günstigsten erscheint. Gleichzeitig wird durch den Infoartikel das Info zum Nachschlagewerk für diejenigen, die sich erst später mit der Thematik befassen und dann die gleichen oder ähnlichen Fragen haben.

Vielen Dank im voraus für Eure Bemühungen.

Die Redaktion

Wer hat was ? ? ?
Wer sucht was ? ? ?

Wer hat ein günstiges Laufwerk zu verkaufen (nach Möglichkeit 40 Spuren/ ss/ dd) ?

Ferner suche ich das relativ neue Buch von L. Röckrath "Programmieren in Maschinensprache". Wer kann es mir mal für einige Tage leihen ?

Günther WAGNER

Derjenige, der am Clubtreffen die Unterlagen (Schaltpläne) für den GENIE wollte, sollte sich bitte nochmals melden.
Manfred HELD

Ich suche schon seit geraumer Zeit ein leeres TRS80 Modell 3-Gehäuse oder Genie 3-Gehäuse. Wo kann ich dies günstig beziehen? Neue Terminalgehäuse sind nicht unter 500,-DM zu erhalten.
Peter Speiß

Ich kann Ringkerntrafos (Durchmesser: 9 cm, Höhe: 3,5 cm) mit den Anschlußwerten 2 * 15 Volt / 2 * 3,3 A zu einem Preis von 30 DM beschaffen. Wer sich in der Bauteilebeschaffung auskennt weiß, daß das ein sensationeller Preis ist. Die Bestellungen bitte direkt an mich schicken.
Hartmut OBERMANN

Fragen, Fragen, Fragekasten

Ich habe nun endlich ein Superscripsit das unter NEWDOS läuft - allerdings mit einem (für mich erheblichen) Schönheitsfehler. Es ist die Version für die amerikanische Tastatur und ich habe die deutsche. Etliche Zeichen sind vertauscht und die Umlaute sind überhaupt nicht erreichbar. Wer weiß Abhilfe (z.B. wo die Tastaur angesprochen wird etc.) ?

Ich habe intern das Laufwerk 0 und extern 2 Laufwerke (Nr. 2 und 3) angeschlossen. Nun habe ich Laufwerk 3 einmal probeweise intern angeschlossen als Laufwerk 1. Dabei ergab sich folgendes: Das Laufwerk 1 lief einwandfrei, formatiert und kopiert einwandfrei - aber: Die alten Disketten (bzw. die Disketten des baugleichen Laufwerk 2) lassen sich nicht im Laufwerk 1 lesen und umgekehrt die vom Laufwerk 1 nicht mehr im Laufwerk 2. Ein Austausch der beiden Laufwerke brachte das gleiche Ergebnis. Das heißt, das ich zwei 80-Spur-Laufwerke hätte, die untereinander verschieden formatieren (nicht aber, wenn diese als Laufwerk 2 und 3 extern angeschlossen werden). An was liegt das? Wer weis Rat? Wer weis, wie ich ev. im Computer z.B. den Controller so einstellen kann, daß das externe Laufwerk 2 als Laufwerk 1 angesprochen wird?

Günther WAGNER

Wer kennt das Innenleben des EG 64 MBA von TCS? Da sich in dem kleinen Kästchen nur 6 IC's und zwei Kondensatoren befinden, finde ich den Preis von ca. 190.-DM etwas happig. Ein Nachbau würde sich also lohnen. Von den IC's (normale TTL-Bausteine) ist die Beschriftung abgeschliffen!

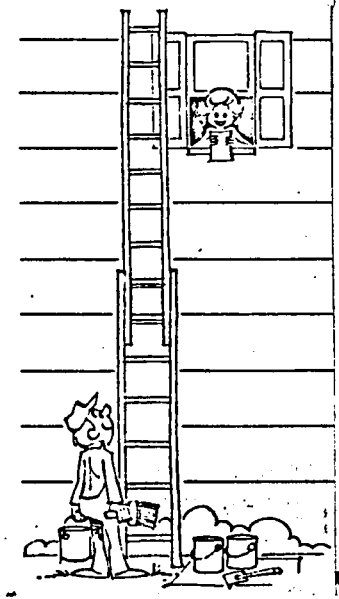
Seit längerer Zeit versuche ich mich mit dem Adventure "Microworld". Es handelt sich dabei um eine Expedition durch das Innenleben eines Computers. Wer kennt die Auflösung oder zumindest einen Teil davon.
Peter Spieß

Aus aktuellem Anlaß sammle ich seit kurzem Artikel über ein sehr heikles Thema. Abmahnung!!!

Seltsamerweise häufen sich die Beiträge in den Computerzeitschriften zu diesem Thema in letzter Zeit. Das läßt darauf schließen, daß sich diese, in Geschäftskreisen durchaus übliche Art eine "Unstimmigkeit" aus der Welt zu schaffen, langsam auch im Hobby-Computerbereich breit macht. Was es damit auf sich hat, wie man sich dagegen schützt bzw. sich dagegen wehrt, könnt ihr in den folgenden Artikeln erfahren.

Ich kann nur jedem wünschen, daß er nie von solchen Praktiken, manchmal windiger und fingier Anwälte verschont bleibt.

Hartmut Ebermann



"Hey Harold! The program says PRINT@, not PAINT@"

Die neue Abmahnmasche: Vorsicht bei Programmangeboten

Die neueste Abmahn-Masche, mit der unterbeschäftigte Rechtsanwälte hart am Rande der Legalität zu Geld zu kommen suchen, trifft die Programmierer

So bekam kürzlich ein Leser, der eine selbstgeschriebene Grafik-Routine für 30 Mark in einer Kleinanzeige angeboten hatte, von einem Rechtsanwalt eine Abmahnung samt Gebührenforderung über 501,60 Mark (willkürlich vom Anwalt festgesetzter »Streitwert«: 20000 Mark). Begründung: In der Anzeige fehle der Hinweis, daß es sich um einen gewerblichen Anbieter handle — das verstoße aber gegen das Gesetz gegen unlauteren Wettbewerb.

Das wäre in Ordnung, wenn es sich bei dem Anbieter um eine Firma handeln würde — oder wenn der Softwareverkauf gewerblich betrieben würde. Nun gibt es aber viele Computerbenutzer, die zwar bereit (und vielleicht sogar interessiert sind), das eine oder andere selbstgeschriebene Programm an Interessenten abzugeben — die aber daraus keineswegs ein Geschäft oder gar Gewerbe machen wollen. Um Ärger mit gewerblich tätigen Firmen, Rechtsanwälten und vor allem dem Finanzamt zu vermeiden, sollten Sie entweder nur tauschen (Tausch zwischen Privatleuten im Rahmen ihres Hobbys ist keine gewerbliche Tätigkeit) oder darauf achten, daß Sie lediglich einen Kostenersatz berechnen. Es ist zweckmässig, den Betrag zu spezifizieren — zum Beispiel 1,30 Mark Porto, 10 Fotokopien á 0,50 Mark, eine Diskette á 4,85 Mark und so weiter.

Wenn Sie einen — und sei er auch nur bescheiden — Gewinn erzielen wollen, müssen Sie auf schriftlichen Unterlagen in Inseraten und so weiter durch eine geeignete Angabe wie »Firma«, »Programmierbüro«, »Softwarevertrieb« oder ähnliches erkennen lassen, daß Sie sich gewerblich betätigen. Sie müssen außerdem das Gewerbe bei der Gemeinde beziehungsweise Stadt anmelden und ein Minimum an Buchführung machen, damit Sie dem Finanzamt jederzeit Umsätze, Kosten und Gewinn nachweisen können. In den meisten Fällen werden Umsatz und Ertrag so gering sein, daß ohnehin keine ernstzunehmende Menge Steuern zu bezahlen ist.

Sollten Sie als Privatmann eine Abmahnung der oben erwähnten Art bekommen, dann schreiben Sie umgehend zurück, daß Sie ihren Computer nur privat benutzen, die Programme für private Zwecke geschrieben haben und durch das Anbieten ihrer selbstgeschriebenen Progamme Kontakt zu anderen Computerbenutzern zum Zweck des Erfahrungs- und Programmaustausches suchen. Ihre selbstgeschriebenen Programme gäben Sie entweder im Tausch oder gegen Ersatz der durch Erstellen und Versenden der Kopie entstehenden Kosten ab. Falls das zutrifft, brauchen Sie auch keine Unterwerfungserklärung abzugeben und keine Gebühren zu zahlen.

(py)

Abmahnschwindler nie gefaßt

Mit einem Abmahnschwindel besonderer Art tat sich im vergangenen Jahr eine »R + S Computerorganisation« in Berlin hervor: Sie trat als angeblicher Wettbewerber auf, verlangte von zahlreichen Anbietern von Raubkopien die Abgabe einer Unterlassungserklärung sowie die Bezahlung einer Gebührenrechnung in Höhe von mehreren hundert Mark. Der Rechnungsbetrag sollte in bar zusammen mit der Unterlassungserklärung an eine Postfachadresse in Berlin geschickt werden. Die Staatsanwaltschaft schaltete sich sehr schnell ein und stellte fest, daß das angegebene Postfach in der vorgegaukelten Form nicht existierte. Es handelte sich dabei, wie die Justizpressestelle jetzt auf Anfrage mitteilte, um die Nummer einer Postlagerkarte bei einem Berliner Postamt, die tatsächlich ausgegeben worden war »ohne daß die Personalien des Empfängers notiert worden wären oder hätten notiert werden müssen«. Bei Beobachtungen in dem Postamt stellte die Kriminalpolizei im vergangenen Jahr zwar einen 15jährigen Jungen, der mit der Postlagerkarte und einer Vollmacht der Schwindelfirma die Post abholen wollte. Als die Polizisten ihn nach dem Auftraggeber fragten, deutete er auf einen etwa hundert Meter vom Postamt entfernt stehenden Mann, der daraufhin zusammen mit einem anderen, mit einem Auto die Flucht ergriff. Da der Junge nach Feststellungen der Polizei als Mittäter ausscheidet und die beiden Flüchtigen nicht identifiziert werden konnten, wurde das Verfahren gegen R + S wohl oder übel eingestellt.

(py)

Abmahnungen

Die Kleinanzeigen in Zeitschriften sind eine beliebte Lektüre nicht nur für Computer-Interessierte, sondern auch für Abmahnvereine.

Kleinanzeigen versprechen ein gutes Geschäft. Doch nicht nur dem Anbieter. Immer häufiger treten jetzt Leute auf, die sich weniger für die angebotenen Waren interessieren, als vielmehr dafür, wie die Ware angeboten wird. Sehr häufig lassen sich da Verstöße gegen das UWG, das „Gesetz gegen den unlauteren Wettbewerb" nachweisen.

Jede Anzeige muß bestimmten Maßstäben gerecht werden. Die Generalklausel des UWG (§ 1) lautet: „Wer im geschäftlichen Verkehr zu Zwecken des Wettbewerbs Handlungen vornimmt, die gegen die guten Sitten verstoßen, kann auf Unterlassung und Schadensersatz in Anspruch genommen werden." Gegen die guten Sitten wird dann verstoßen, wenn angenommen wird, daß der Tatbestand des Kundenfangs, der Behinderung, der Ausbeutung und/oder des Rechtsbruchs vorliegt. Konkrete Fälle sind: Ein Akustikkoppler ohne Postzulassung wird ohne Hinweis auf die fehlende Postzulassung angeboten, oder es wird kein Firmenstatus angegeben, sondern nur eine Telefonnummer des Anbieters.

Abmahnvereine

Im strengen Sinne können in diesen Fällen Unterlassungsansprüche erhoben werden. Dazu sind in erster Linie die Mitbewerber und auch die „Verbände zur Förderung gewerblicher Interessen" nach UWG § 13 berechtigt. Bei letzteren, gemeinhin als Abmahnvereine bekannt, besteht jedoch teilweise Anlaß, ihr Vorgehen kritisch unter die Lupe zu nehmen. Mit Hilfe des UWG läßt sich nämlich nicht nur gut abmahnen, sondern auch gut absahnen.

Auf eine Kleinanzeige in einer Zeitschrift bekam ein Inserent von fünf verschiedenen Rechtsanwälten beziehungsweise Organisationen Abmahnschreiben mit Geldforderungen zwischen 500 und 1000 Mark und Klagedrohungen, weil in diesem Fall die Mehrwertsteuer getrennt ausgewiesen worden war.

Eine Abmahnung enthält in der Regel folgendes: Der Abgemahnte wird zuerst auf seinen Verstoß hingewiesen; unter Hinweis auf die Einleitung gerichtlicher Schritte (einstweilige Verfügung oder Klage) wird er aufgefordert, den Verstoß zu unterlassen; unter Fristsetzung wird er dazu aufgefordert, eine „Unterwerfungserklärung" zu unterzeichnen, in der er den Verstoß gegen das UWG zugibt.

Wird dafür ein Rechtsanwalt eingesetzt, so kostet dies alles Gebühren, und damit machen manche dubiose Abmahnvereine das schnelle Geld. Sie durchforsten die Kleinanzeigen der Zeitschriften nach Verstößen gegen das UWG und schicken Abmahnschreiben los. Für den Inserenten, der sich oft keines Vergehens bewußt ist, bringt das vor allem viele Kosten mit sich.

Die Gebühr für eine Abmahnung durch einen Rechtsanwalt berechnet sich nach dem Streitwert. Bei mittleren Wettbewerbsverstößen wird in der Regel vom Abmahner ein Streitwert von 10 000 Mark (bei Geschäftsleuten oft auch 30 000 Mark) angenommen. Der Rechtsanwalt macht die halbe BRAGO-Gebühr, Portopauschale von 40 Mark und Umsatzsteuer geltend und kommt so auf rund 300 Mark. Nimmt sich auch der Abgemahnte einen Anwalt, so kann dessen Tätigkeit nach denselben Grundsätzen bewertet werden und weitere 300 Mark kommen dazu.

Reagiert der Betroffene nicht auf die Abmahnung, so kann es noch teurer werden. Macht der Abmahner die angedrohten Schritte wahr, dann beantragt er im Regelfall eine einstweilige Verfügung des zuständigen Landgerichts. Diese

erhält der Abgemahnte als Zustellungsurkunde vom Gerichtsvollzieher. Mit der einstweiligen Verfügung wird der „Wettbewerbsstörer" aufgefordert den Verstoß gegen das UWG zu unterlassen. Bei Zuwiderhandlung wird eine Geldstrafe bis zu 500 000 Mark oder eine Haft bis zu zwei Jahren angedroht.

Der Betroffene muß auch diesmal zahlen: Die vom Anwalt geltend gemachten Gebühren haben sich verdoppelt. Dazu kommen die Kosten des Landgerichts für die einstweilige Verfügung von rund 100 Mark.

Teure Gebühren

Doch damit nicht genug: Fordert der Anwalt den Abgemahnten nach Erlaß der einstweiligen Verfügung erneut auf, eine Unterwerfungserklärung abzugeben, werden für dieses „Abschlußschreiben" wiederum Gebühren fällig. Diesmal berechnet nach dem Wert, der der Klage zugrunde gelegt wurde. Im Regelfall wird dann vom dreifachen des ursprünglichen Streitwertes ausgegangen, also bei zuerst 10 000 Mark von nunmehr 30 000 Mark.

Zahlt der Betroffene die Gebühren des Abmahners nicht, so erhält er einen Mahnbescheid, der ihn wieder etwa 100 Mark kosten kann.

Ein Prozeß gegen die einstweilige Verfügung muß vom Landgericht des Abmahners geführt werden. Wegen des Anwaltszwanges kann es sein, daß der bisherige „Vertrauensanwalt" nicht zugelassen wird. Die Kosten des neuen Anwalts und die Gerichtskosten summieren sich.

Gegenmaßnahme

Aber es gibt einen Lichtblick. In seinem Urteil vom 12. April 1984 hat sich der Bundesgerichtshof zu den Abmahnpraktiken geäußert: Das Abmahn-Opfer muß nun nicht mehr die für die Abmahnung erhobenen Gebühren zahlen, wenn die Einschaltung eines Rechtsanwaltes überflüssig war. Dies ist immer dann der Fall, wenn der Abmahnende die für die Abmahnung „maßgeblichen Kriterien, insbesondere Branchenübung und Verkehrsauffassung aus eigener Sachkunde beurteilen kann und der Erwerb der übrigen erforderlichen Sachkenntnis ihm angesichts des Umfanges seiner Abmahntätigkeit zuzumuten ist. Bei einer solchen Ausstattung des Klägers würde sich bei den typischen und durchschnittlich schwierigen Abmahnungen die Einschaltung eines Rechtsanwalts erübrigen und er wäre daher nicht ..." erforderlich.

Bei Eingang der Abmahnung sollte der Abgemahnte also im Zweifel nach Einholung rechtlicher Beratung unverzüglich eine Unterwerfungserklärung abgeben und unter Hinweis auf das Urteil des BGH die Zahlung der geforderten Gebühren verweigern. Damit wäre dem UWG genüge getan und der Gebührenschinderei könnte damit ein Riegel vorgeschoben werden.

Hans Preisker

Abmahnung und Durchsuchung – was tun!

Wer aktiv ist, berührt automatisch Interessen anderer, nur wer nichts tut, kann niemand stören. Wer mit seinem Computer arbeitet, ist aktiv, und je mehr er aktiv ist, desto schneller kommt er mit anderen in Konflikt. Mit anderen in Konflikt zu kommen, heißt aber noch nicht, mit dem Gesetz in Konflikt zu kommen, denn jeder meint, daß das Recht auf seiner Seite sei. Es gibt auch Rechte für den einen und den anderen. Kommt das Recht des einen mit dem des anderen in Konflikt, entscheiden die Gerichte, wessen Recht vorrangig ist.

Viele junge Computerfreunde haben mit Recht und Gesetz nichts zu tun gehabt. Sie sind unerfahren, die Eltern haben sich oft so bewegt, daß auch sie nie mit anderen Streit hatten. So können viele auch nicht aus Erfahrungen der Eltern ein Wissen ableiten. Umso wichtiger ist es, daß hier einmal klar gesagt wird, was eine Abmahnung ist und wie man sich dabei verhält. Und erst recht muß man wissen, was man bei Durchsuchungen tut oder besser nicht tut. Zwischen Abmahnung und Durchsuchung sind wesentliche Unterschiede. Wer abgemahnt wird, wird von einem Privatmann privat verfolgt. Kommt die Polizei zur Durchsuchung, wird man vom Staat verfolgt. Wer abgemahnt oder durchsucht wird, braucht erst einmal selbst nichts zu tun. Nichts ist falscher, als sofort zu reagieren. Liegt die Abmahnung auf dem Tisch, notiert man die Frist und denkt einmal nach.

Abmahnungen

Während dieser Frist denkt man nach, ob die Abmahnung begründet ist oder begründet wäre, beides kann der Fall sein. Wer wirklich Urheberrechte verletzt und es nicht mehr tun will, gibt eine Unterlassungserklärung ab. Aber immer schön in die Unterlassungserklärung hineinschreiben: "Ohne Anerkenntnis, Veranlassung gegeben zu haben", und immer schön den Satz rausstreichen, daß man die Kosten tragen will. Wenn nichts bewiesen werden kann, kann einem insbesondere nicht bewiesen werden, daß man die Abmahnung veranlaßt hat.

Wer meint, daß er tun darf, was abgemahnt wird, braucht **nichts zu tun.** Er wartet die einstweilige Verfügung ab. Nichts tun ist dann am besten, um den Gegner nicht erst auf intelligente Gedanken zu bringen. Wichtig ist (was die meisten Anwälte falsch machen): **Keinen Widerspruch** einlegen! Der Könner beantragt **Fristsetzung zur Erhebung der Hauptsacheklage beim Gericht. Einen Anwalt braucht man solange noch nicht.** Dann muß der Abmahner Hauptsacheklage erheben und beweisen. Beim Widerspruch verbleibt der Rechtsstreit im Bereich des juristischen Wischi-Waschi der Wahrscheinlichkeitstheorien. Das ist schlecht, besonders wenn man eine seriöse Firma als Gegner hat. Gewinnt man den Hauptsacheprozeß, wird die einstweilige Verfügung auch aufgehoben und man hat einen Schadensersatzanspruch für die Zeit ihrer Geltung.

Durchsuchungen

Gegen Durchsuchungen kann man nichts machen, sondern muß diese über sich ergehen lassen. Grundsätzlich muß man wissen: die Polizei ist nur ausführendes Organ. Jeder Polizist ist, leger gesagt, nur Hampelmann eines Staatsanwaltes. Die Polizei kann nichts entscheiden, sie ist für die Durchsuchung nicht verantwortlich. Sie sammelt nur Material. Deswegen gilt: nichts sagen, denn unüberlegte Worte und mißverständliche Sätze merken sich die Polizisten, schreiben sie nieder und verwenden sie gegen den Durchsuchten. Daß die Polizei verpflichtet wäre, sich auch zugunsten des Durchsuchten etwas zu notieren, ist bloße Rechtstheorie (§ 161 StPO).

Beschwerden gegen die Durchsuchungen sind unzulässig, weil die Durchsuchung immer abgeschlossen ist, wenn die Beschwerde beim Gericht eingegangen ist. Also laßt die Polizei suchen. Auf jeden Fall sollte man sich aber eine Abschrift des gerichtlichen Durchsuchungsbeschlusses aushändigen lassen. Wenn die Polizei was gefunden hat, erklärt sie, ob sie es beschlagnahmen will. Was sie beschlagnahmen will, und hierauf müßt Ihr bestehen, muß einzeln im Protokoll aufgeschrieben werden. Also nicht einfach "10 Disketten", sondern: 1 Diskette, 3,5", BASF, beschriftet mit...usw. Haben die Beamten keine Zeit, dann besteht darauf, daß alles in einen Sack, notfalls eine Plastiktüte verpackt wird, die zu versiegeln ist. Das Verzeichnis kann dann an einem der nächsten Tage, wenn die Leute mehr Zeit haben, in Eurer Anwesenheit erstellt werden. Immer verbal sehr zurückhaltend sein! Man darf keinem Polizisten Gelegenheit geben, Ordnungsmacht zu spielen. Der Polizist muß ganz in der Rolle des Hilfsbeamten der Staatsanwaltschaft bleiben.

Gegen die Beschlagnahme kann man dann eine Beschwerde schreiben. Meist sind die Durchsuchungsbeschlüsse der Gerichte aus der hohlen Hand geschrieben, ein Verdacht aus den Fingern gesaugt. Aber selbst wenn was dran ist, reicht es meist später im Hauptverfahren nicht aus. Dann gilt der Beweis, vorerst reicht aber der Verdacht. Deswegen darf man keinen Fehler machen und selbst etwas zur Sache sagen. Denn wer jede Woche Krimis im Fernsehen sieht, merkt, daß die Mörder sich durch eigenes Gerede überführen. Deswegen gilt der Grundsatz: Ruhe ist die erste Bürgerpflicht; in der Aufregung sagt man mit Sicherheit das Falsche. Die meisten Leute werden verurteilt, weil sie etwas sagen (sich zur Sache einlassen), was ihnen widerlegt werden kann. Dann braucht ihnen nichts mehr bewiesen zu werden, sondern das Gericht beweist nur, daß die eigene Aussage widerlegt ist. Und dann hat man sich selbst seine Chancen kaputt gemacht.

Was ist das Ergebnis einer Durchsuchung? Die Polizei beschlagnahmt einen Computer. So einen darf man haben. Sagt man also nicht, daß man damit auch kopiert hat (auch mal ganz für sich privat kopiert hat), ist nichts bewiesen. Auch wenn Kopiermaterial gefunden wird, es ist nichts bewiesen, weil man seine eigenen Dateien so oft kopieren kann wie man will. Man darf auch eigene Programme schreiben und diese kopieren. Man kann sich also nur selbst um Kopf und Kragen reden.

Kleinanzeigen zum Superbilligpreis

Und was ist, wenn die Polizei fremde Kopien findet? Wer sich nicht zur Sache einläßt, braucht dann überhaupt keine Antwort zu geben. Wer sich aber anfangs rauszureden versuchte, der muß jetzt solche Antworten entgegnen, die im jetzt peinlich werden. Wer weise von Anfang an geschwiegen hat, erinnert an seine Aussageverweigerung und die gefundenen Raubkopien sind als Beweismittel wertlos. Man kann diese von einem Raubkopierer erhalten haben und wollte diese gerade prüfen. Diese denkbaren Möglichkeiten trägt man aber besser erst im späteren Verfahren vor.

Wie geht das spätere Verfahren weiter? Drei von vier Verfahren werden eingestellt. Also werden, wenn man nichts gesagt hat, nach einiger Zeit die Gegenstände zurückgegeben.

Dies ist das wahrscheinlichste. Wer angeklagt wird, der kann nun, aber erst jetzt, in einem guten Schriftsatz den Verdacht entkräften. Denn erst jetzt legt man die Beweise der Unschuld hin. Wer zu früh seine Karten ausspielt, seine Unschuld zu früh beweisen will, gibt der Staatsanwalt nur Gelegenheit, die Vorwürfe zu untermauern und neue Argumente zu bringen. Anklagen müssen dünn bleiben. Das Wichtigste ist, Nerven zu behalten, weil gerade in der ersten Aufregung die Fehler passieren. Die Zeit für die Verteidigung ist dann, wenn die Anklage geschrieben ist. Vorher füttert man bloß die Polizei mit Wissen.

Weil viele junge Leute auf rücksichtslose Weise und auch durch die heuchlerische Freundlichkeit der Polizei hereingelegt werden, etwas zu sagen, habe ich mit anderen Profis einen COMPUTER PIONEER CLUB gegründet. Wir bilden unsere Mitglieder aus, solchen Angriffen standzuhalten. Wir helfen uns gegenseitig mit Erfahrungen und können gemeinsam manche Kopie machen, ohne uns zu gefährden. Warum wollt Ihr Euch aus Rache dafür hängen lassen, daß die echten Profis nicht schlagbar sind. Also verhaltet Euch richtig und schreit und schreibt, wenn Ihr Euch verunsichert fühlt. Die Polizei ist ganz gewiß nicht der Freund und Helfer. Sie ist es von Gesetzeswegen nicht und könnte es auch gar nicht sein. Wer will schon solche Freunde, die Demonstranten verprügeln und Parksünder anzeigen, hinter Büschen Schnellfahrer stoppen und wenn man sie braucht, nicht zuständig sind. In der Demokratie schließt man sich mit Leuten zusammen, die die gleichen Interessen haben.

Graf Adelmann, 7794 Wald 3

Computerkäufer sind anders!

Daß Computerfans eine Zielgruppe sind, die man auf dem Markt nicht mehr übersehen darf, ist bekannt. Konsequenterweise werden die ersten Marktforschungsunternehmen tätig. Jüngst ist eins der ersten Untersuchungsergebnisse bekannt geworden: Es ist äußerst schmeichelhaft.

Die Computerfans sind mit 71 % überwiegend männlichen Geschlechts. Mit ihrer hervorragenden Ausbildung – 22 % besitzen Abitur und haben ein Studium absolviert – heben sie sich von dem Gesamtdurchschnitt (13 %) deutlich ab.

55 % aller Befragten bezeichneten sich gegenüber allem Neuen aufgeschlossen. Zum Vergleich: Der Durchschnitt der Gesamtbevölkerung liegt bei lediglich 36 %. Computerfans finden Technik zu 52 % faszinierend (Durchschnitt: 27 %). 50 % aller Befragten arbeiten beruflich mit elektronischen Datenverarbeitungs-

Siegmar Wirtz --- Heinz Heise GmbH

Ein BASIC-Kurs fuer Nicht-Mathematiker und echte Amateure. Neben Grund- und Aufbaukurs gibt es zahlreiche Beispiele (eine gute Sammlung!).

Nr. 0002: Computerwissen

Michael Scharfenberger --- Markt & Technik

Tips fuer die Auswahl und Beschreibung von Anwendungsmoeglichkeiten von Hard- und Software sowie Erklaerung von mehr als 500 Begriffen.

Nr. 0003: Computerspiele und Knobeleien programm. in BASIC

Ruedeger Baumann --- Vogel-Buchverlag (CHIP-Wissen)

Von der Spielidee ueber die Spielstrategie kommt es zum Programm selbst. Keine Sammlung von Spielkonserven - keine Programmierkenntn. erford.

Nr. 0004: Mein Home-Computer - Eine Verbraucherfibel

- --- Vogel-Verlag (HC-Leserservice)

Die besten Tips fuer Kauf und Anwendung von Home-Computern.

Nr. 0005: Programmieren mit dem ZX81 in Basic u. Masch.-code

E. Floegel --- Hofacker, Holzkirchen

Sammlung von Spiel-, Schul- und anderen Programmen sowie einem Kapitel ueber die Programmierung des Prozessors Z80 (gute Programme dabei)

Nr. 0006: Games For Your TRS-80

Chris Palmer --- Virgin Books (Great Britain)

Sammlung von 20 Basic-Spiel-Programmen und einer Anleitung, wie man bessere Programme schreibt.

Nr. 0007: Introduction to TRS-80 Graphics

Don Inman --- dilithium Press (Portland-USA)

In diesem Buch wird gezeigt, was man mit der TRS-80 Graphik machen kann und vor allem wie. Beispiele und Aufgaben veranlassen zum experiment.

Nr. 0008: More Basic Computer Games

David H. Ahl --- Creative Computing Press, USA

84 Spiele fuer den TRS-80, wobei einige sehr interessante dabei sind. Das Buch ist fuer Freunde von Basic-Computer-Spielen nur zu empfehlen.

Nr. 0009: BASIC: Dateien, Listen und Verzeichnisse

Busch Rudolf --- Franzis-Verlag GmbH, M nchen

Eine Software-Sammlung mit vielen nuetzlichen Programmen in Kursform (also mit Lern-Effekt).

Busch Rudolf --- Franzis-Verlag

Eine Software-Sammlung mit vielen nuetzlichen Programmen in Kursform (also mit Lern-Effekt).

Nr. 0011: TRS-80 PROGRAMS

Tom Rugg und Phil Feldman --- Dilithium Press, Beaverton,USA

32 BASIC-Programme (Erziehung, Anwendung, Spiele, Graphic, Mathematik und Verschiedenes) fuer Level II.

Nr. 0012: Programme und Tricks fuer Genie I und Genie II

Clemens Becher, Franz Seger --- ?

Viele Programme, Tips und Tricks fuer den Genie.

Nr. 0013: BASIC: Alles ueber PEEK und POKE

Heiko Requardt --- Franzis-Verlag

Eine Software-Sammlung in BASIC (mit vielen guten Tips und Tricks fuer den 'Amateur').

Nr. 0014: TRS-80 und Video Genie ROM-Listing fuer Level II

Luidger Roeckrath --- ?

ROM-Listing, RAM-Adressen, I/O-Adressen, Unterprogramme, Basic-Anweisungen und Funktionen, Aufzeichnungsformate auf Cassette, ...

F U N D G R U B E U N D B Ü C H E R

****

Nach wie vor bauen wir unsere Bücherbibliothek auf. Diese Bücher können übrigens von Jedem Mitglied ausgeliehen werden. Das gleiche gilt für die FUNDGRUBE. Das sind 6 Ordner deren Inhalt aus dem 4. Info ersichtlich ist.
Die Bücher und die Fundgrube-Ordner können bei Günther Wagner entliehen werden.

> ***Wenn man Intelligenz als die Fähigkeit definiert, neue Dinge zu lernen und Lösungen für Probleme zu finden, die das erste Mal auftauchen – wer ist dann intelligenter als das Kind?***

Lebensdauer von Disketten

Disketten, auch Floppy Disks genannt, sind gerade für Mikrocomputer das Massenspeicher-Medium der Wahl, sowohl was die Speicherkapazität betrifft als auch in bezug auf Datensicherheit. Doch wenn man die Arbeit vieler Stunden oder gar Wochen einer solchen Scheibe zur Aufbewahrung anvertraut, dann möchte man es vielleicht einmal ganz genau wissen, wie sicher und für welchen Zeitraum die Daten wieder abrufbar bleiben.

Die eigentliche Information auf einer Magnetplatte wird durch die Ausrichtung winziger Magnete in der Beschichtung gebildet, wobei ein Datenbit etwa vier tausendstel Millimeter lang ist. Bleibt nun die Umgebungstemperatur im Bereich von 0...50 °C und setzt man die Diskette keinen starken Magnetfeldern oder mechanischen Beanspruchungen aus, dann ist diese Information praktisch unbegrenzt haltbar. So liegt zum Beispiel bei der BASF in Ludwigshafen ein Magnetband, dessen Konzertaufnahme aus dem Jahre 1936 bis heute nichts an Qualität verloren hat. Klar, daß man durch Erfahrungen in der Herstellung von Trägern die magnetischen Aufzeichnungen seit damals verbessert hat, und somit von einer unbegrenzten Haltbarkeit der Information auf einer Diskette ausgehen kann.

Auch der Abrieb am Kopf bei häufig gelesenen Floppy-Disks ist wesentlich unkritischer, als man vielleicht annehmen könnte: Einen sauberen Schreib-/Lesekopf vorausgesetzt verträgt jede Spur mehrere Millionen Kopfdurchläufe; also bei 300 Umdrehungen pro Minute eine Zeit von einigen Monaten ununterbrochenen Zugriffs – und das auf jede einzelne Spur!

Daß dann trotzdem manchmal Disketten nicht mehr zu lesen sind und damit das Ergebnis vieler arbeitsreicher Stunden dahin ist, hat meist ganz andere Ursachen. Einer der größten Datenvernichter in diesem Zusammenhang ist wohl der Kaffee. Aber auch andere Getränke oder auch Fingerabdrücke können, auf die magnetisierbare Schicht der Diskette aufgebracht, die Informationen wirksam abdecken und somit jedem Lesezugriff entziehen. Im Fall der Fingerabdrücke ist es meist möglich, durch häufige Leseversuche die Verunreinigung zu beseitigen und so die Datei zu retten. Bei anderen Verschmutzungen hilft oft gar nichts mehr: Die Lösungsmittel, die wirklich helfen können, lösen auch gleich die Trägerschicht. Sollte es trotzdem gelingen, eine so gereinigte Floppy noch einmal zu lesen, muß diese anschließend gleich aus dem Verkehr gezogen werden: Durch die Reinigung verschwindet nämlich auch das Gleitmittel, das den Kopf gegen großen Abrieb schützt. Und ein neuer Schreib-/Lesekopf ist bestimmt teurer als einige neue Disketten. — Kr.

CLUB 80 Mitgliederadressenliste

Name	Vorname	Straße	PLZ	Stadt	Telefon
Alber	Herbert	Alemannenstr. 20	7732	Niedereschach	07721 /7102
Baldes	Hans	Johann-Strauss-Str. 6	8025	Unterhaching	089 /6115179
Beckhausen	Wolfgang	Vuerfelser-Kaule 38	5060	Bergisch-Gladbach 1	02204 /62781
Boecker	Dieter	Lehmweg 4	2930	Varrel 1	04451 /7640
Boeckling	Ulrich	Am Sonnenhang 11	5414	Vallendar	0261 /69522
Bozek	Hans Juergen	Gut Fachenfelde	2093	Stelle	
Buskowiak	Thomas	Eschersheimer Landstr. 257	6000	Frankfurt 1	069 /5601621
Dreyer	Gerald	Am Speiergarten 8	6200	Wiesbaden-Bierstadt	06121 /508218
Grajewski	Werner	Zedernweg 29	4220	Dinslaken	02134 /54573
Hallup	Matthias	Junggesellenstr. 15	4600	Dortmund	
Held	Manfred	Stirnerstr. 22	8035	Pleinfeld	09144 /6563
Hermann	Klaus	Gartenstr. 22	7401	Pliezhausen	07127 /70024
Hummel	Anton	Schubertstr. 2	7612	Haslach	07832 /8289
Jablotschkin	Rainer	Thiekamp 29	4780	Lippstadt 8	
Kasper	Dieter	Zeppelinstr. 9	8952	Marktoberdorf	08342 /1630
Koenig	Hans J.	Hebbelstr. 25	2080	Pinneberg	04101 /209444
Konrad	Josef	Anzengruber 35	8038	Groebenzell	08142 /8494
Kuhn	Eckehard	Im Dorf 14	7443	Frickenhausen 1	07022 /45417
Marx	Andreas	Mecklenburgring 48	6600	Saarbruecken	0681 /812983
May	Holger	Marienstr. 9	5768	Sundern 2	02935 /1668
Neueder	Jens	Panoramastr. 21	7178	Michelbach/Bilz	0791 /42877 (dienstl.44-458)
Obermann	Hartmut	Schwalbacher Str. 6	6209	Heidenrod/Kemel	06124 /3913
Perschbach	Patrick	Waldstr. 52	5000	Koeln 91	0221 /872118
Piller	Walter	Rohnenstr. 8	CH-8835	Feusisberg	01 /7847418
Preuss	Lothar	Lautshof 13	2940	Wilhelmshaven	04421 /84247 (dienstl. 884-1)
Rank	Heinrich	Fruehlingstr. 2	8000	Fuerstenfeldbruck	08141 /3791
Schaefer	Walter	Rathausstr. 4	8168	Miesbach	08025 /1631
Schneider	Manfred	Rheinkasseler Weg 11	5000	Koeln 71	0221 /707044
Schrewe	Christian	Fliederweg 32	4000	Duesseldorf 31	0203 /740897
Schroeder	Gerald	Am Schuetzenplatz 14	2105	Seevetal 1	04105 /2682
Sickmann	Bernhard	Fleigenweg 2	4430	Steinfurt 2	02552 /60344
Smerling	Frank	Tangstedter Str. 5	2080	Pinneberg	04101 /207284
Sopp	Arnulf	Wakenitzstr. 8	2400	Luebeck 1	0451 /791926
Spiess	Peter	Trugenhofenerstr. 27	8859	Rennertshofen	08434 /454
Stephan	Hans-Martin	Am Glasesch 9a (Postf. 1207)	4506	Hagen a.TW.	05401 /99585
Stevens	Peter	Postfach 6327	7800	Freiburg	0761 /35384
Trapp	Harald	Kranichstr. 46	4270	Dorsten 1	02362 /42497
Troesch	Eberhard	Altenessener Str. 414	4300	Essen 12	0201 /342324
Voigtlaender	Holm	Haselnussweg 38	6940	Weinheim	06201 /65241
Wagner	Alexander	Theresienstr. 21c	8224	Chieming	08664 /1500
Wagner	Guenther	Gartenstr. 4	8201	Neubeuern	08035 /3361
Wies	Jean-Claude	Harthweg 9	6600	Saarbruecken	0681 /582513
Wucherer	Juergen	Brauneggerstr. 14	7750	Konstanz	07531 /29145
Zwickel	Walter	Lengfelden 123	A-5101	Bergheim	0043662/51130

Wegen Nichtbezahlung von Beiträgen
- trotz mehrmaliger Anmahnung -
wurden vom CLUB ausgeschlossen :

Peter Schmidt
Robert Trost
Dieter Neukam
Donald Wright

Bitte überprüft Eure Adresse
und meldet mir Fehler oder Veränderungen.
Die Redaktion

Vorstand

Kontaktadresse für Clubangelegenheiten, Clubbücherei /Fundgrube, Clubkasse

Günther WAGNER
Gartenstraße 4
8201 Neubeuern
Tel.: 08035 /3361
< 18 - 20 Uhr >

Programmbibliotek

Kontaktadresse

Hartmut OBERMANN
Schwalbacher Straße 6
6209 Heidenrod /Kemel
Tel.: 06124 /3913

Redaktion

Kontaktadresse

Jens NEUEDER
Panoramastraße 21
7178 Michelbach /Bilz
Tel.: 0791 /42877
tagsüber 0791 /44-450

Adventure-Ecke

Kontaktadresse

Alexander WAGNER
Theresienstr. 21c
8224 Chieming
Tel.: 08664 /1500

Hardware

Kontaktadresse

Walter ZWICKEL
Lengfelden 123
5101 Bergheim (Austria)
Tel.: 0043662 /51130

Redakteure

dieser Ausgabe

Herbert Alber * Ulrich Böckling
Eckehard Kuhn * Jens Neueder
Hartmut Obermann * Gerald Schröder
Arnulf Sopp * Günther Wagner
Walter Zwickel
sowie Artikel aus: MC, CHIP, c't und Computer Persönlich

Bankverbindung des CLUB 80

Sparkasse Rosenheim, BLZ 711 500 00
auf Konto-Nr. 194 712
Postscheckkonto der Sparkasse
Nr. 8077-801

Das INFO erscheint zweimonatlich.

Es erfolgt keine Zensur oder Kontrolle der jeweiligen eingeschickten Infobeiträge durch die Redaktion.

Schluß

Hallo Club-80er,

Ihr wisst ja, daß sich in unserem Club nun einiges verändert hat und so möchte ich am Schluß als "Redaktuer" noch ein paar Sätze loswerden.

Zunächst einmal zu den Terminen. Unsere Clubzeitung kam ja bisher so ungefähr im zweimonatlichen Turnus. Da ich in den nächsten Jahren auch privat ziemlich eingespannt bin, möchte ich das in dieser Weise beibehalten.

Damit ein reibungsloser Ablauf gewärleistet ist habe ich mich zur Einführung eines Redaktionsschlusses entschlossen. Der Redaktionsschluß für das jeweilige INFO ist der letzte Tag der geraden Monate im Jahr. Für die Endfertigstellung, das Vervielfältigen und den Versandt an Euch rechne ich dann so ca. 14 Tage. Somit bekommt Ihr das INFO Anfang bis Mitte Jan., März, Mai, Juli, Sept. und Nov. ins Haus. Um eine termingerechte Ausgabe unseres INFO zu gewährleisten, bitte ich daher um Einhaltung unseres Redaktionsschlusses.

Desweiteren möchte ich an Alle appellieren für die INFO zu schreiben, wobei die Größe des Artikels nicht entscheidend ist. -Ob groß oder klein jeder hat doch im Umgang mit seinem Computer einen Trick oder Kniff über den sich etwas schreiben ließe. Also spitzet Eure Federn und versucht es einmal mit einem kleinen Infobeitrag. Vielleicht stellt sich dann sogar ein neues Talent heraus. Anhand des Inhaltsverzeichnisses könnt Ihr ersehen, welche Themengruppen wir ansprechen. Einer regen Beteiligung sehe ich mit Begeisterung entgegen.

Bei Gelegenheit schreibt doch bitte ob, bzw. wie Ihr unter Tage telefonisch zu erreichen seit. Es wäre für mich einfacher, wenn ich bei Euch wegen eventuellen Rücksprachen -welche das INFO betreffen- tagsüber anrufen könnte.

Ansonsten hoffe ich, daß das INFO in der Form, in der es gerade vor Euch liegt, gefällt. Für Verbesserungsvorschläge oder neue Anregungen wäre ich Euch dankbar.

Bis zum nächsten INFO

Jens Neueder