Nr. 1/87 Januar
DM 6.50, sfr. 6.50, öS 50, Lit 5900, hfl. 7.50

PEEKER

MAGAZIN FÜR MIKROCOMPUTER

Shape-Entwicklungspaket

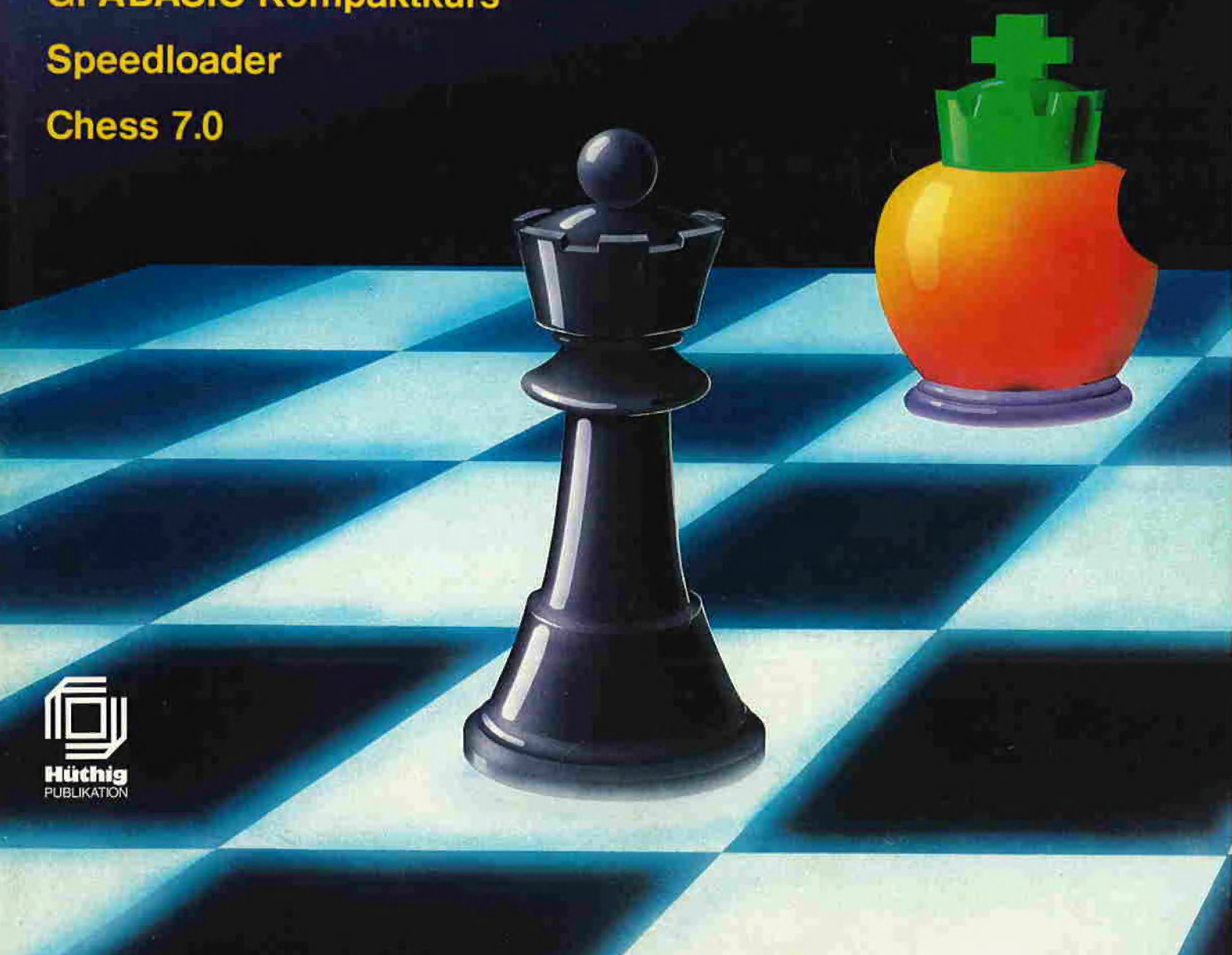
Ampersand-Utilities

Kyan-Luxus-Catalog

GFABASIC-Kompaktkurs

Speedloader

Chess 7.0



Hüthig
PUBLIKATION

Wichtige Information für unsere peeker-Leser

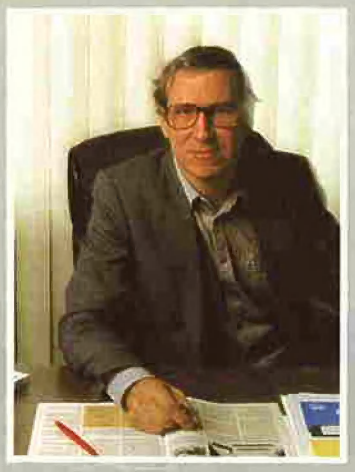
Sie erhalten Ihren »**peeker**«

- bei Ihrem **Zeitschriftenhändler am Bahnhof**
- bei allen **Apple-Händlern**
- und natürlich **beim Verlag.**

Denn der sicherste Weg, keine Ausgabe des »**peeker**« zu versäumen, ist noch immer das Abonnement zum Jahresvorzugspreis. Dabei sparen Sie bares Geld. Beachten Sie in diesem Zusammenhang unsere Anzeige auf Seite 9, oder rufen Sie uns an:
Telefon 0 62 21/489-281 (Peeker-Leserservice).

Hüthig PUBLIKATION

Dr. Alfred Hüthig Verlag, Im Weiher 10, 6900 Heidelberg
BTX ✱51851#



EDITORIAL

Vorschau

Neben den normalen Programmen sind für die nächsten Monate einige spezielle Leckerbissen in Vorbereitung, die wir an dieser Stelle bereits ankündigen möchten. Nachdem wir im letzten Jahr an die Fortsetzungsbezieher unserer Sammeldisketten einen ausgewachsenen Applesoft-Editor kostenlos verteilt haben, steht für eine der nächsten Disketten ein vollständiger ProDOS-Assembler für 6502/65C02 auf dem Programm, der hinsichtlich des Redigierkomforts, der Verarbeitungsgeschwindigkeit und der sonstigen Leistungen den herkömmlichen Assembler (Big Mac usw.) keineswegs nachsteht, sondern sie sogar in vielen Features übertrifft. Außerdem können Sie Big-Mac-Quelltexte von den früheren Peeker-Sammeldisketten normalerweise fast unverändert assemblieren.

Darüber hinaus werden wir unseren treuen Fortsetzungsbeziehern das Dateiverwaltungsprogramm DB-Meister als kostenlose Zugabe zur Verfügung stellen, und zwar teilweise sogar in Quelltextform. Aus Platzgründen müssen diese Programme jedoch leider auf mehrere Disketten (ab Disk #25) verteilt werden.

Beachten Sie bitte, daß der Assembler und der DB-Meister nicht auf den normalen Sammeldisketten für Einzelbezieher enthalten sein werden, denn diese Programme sind nur als Treueprämien für Fortsetzungsbezieher gedacht. Wenn Sie also ohnehin einen Assembler oder ein Dateiverwaltungsprogramm kaufen wollten, können Sie als Fortsetzungsbezieher viel Geld sparen. Beispielsweise würden sich 6 Sammeldisketten ab Disk #25 auf nur DM 120,– belaufen, während der DB-Meister allein DM 290,– kostet.

In unserer Kompaktkurs-Reihe sind zwei Großbeiträge in Arbeit. Der Kurs zu dem beliebten GFABASIC beginnt mit diesem Heft. Übrigens hat der Firma GFA Systemtechnik unsere Aufstellung des Befehlssatzes aus Heft 11/86 so gut gefallen, daß sie dort als Referenzkarte erscheinen wird. GFABASIC ist auch über unseren Hüthig Software Service erhältlich, wobei wir an unsere GFABASIC-Käufer nach Abschluß des Kurses eine Atari-Sonderdiskette mit allen im Peeker veröffentlichen GFABASIC-Programmen sowie weiteren Utilities kostenlos verschicken werden.

Als zweiter Kompaktkurs ist ein umfangreicher Zweiteiler über den 65816/65802-Prozessor unter besonderer Berücksichtigung des Apple IIgs vorgesehen, der auf die Februar- und Märzausgaben des Peekers verteilt werden wird.

Damit Applesoft-Programmierer nicht zu kurz kommen, haben wir einige lupenreine BASIC-Programme eingeplant, beispielsweise in diesem Heft das Shape-Entwicklungspaket. Im übrigen stehen wie bisher nützliche Assembler- und Pascal-Utilities auf dem Programm.

Apple-II-Besitzer, die sich mit dem Gedanken tragen, den neuen Apple IIgs zu erwerben, müssen wir im Hinblick auf IIgs-Programme einstweilen vertrösten. So war auf der jüngsten COMDEX-Messe – Bericht folgt im Februar-Heft – nicht einmal die Endversion von ProDOS 16 zu sehen. Gerüchten zufolge soll die Platine wegen des 4-MHz-65816 noch einmal geändert werden. Deshalb werden sich die Erscheinungstermine für viele der angekündigten Apple-IIgs-Programme um mehrere Monate verschieben.

Ulrich Stiehl

Verteilung der Treueprämien

Disk #25 (1/87)
Maskengenerator-Module zu DB-Meister

Disk #26 (2/87)
Kompletter 6502-ProDOS-Assembler
Dateipflege-Module zu DB-Meister

Disk #27 (3/87)
Sortier/Druck-Module zu DB-Meister

(Ab Disk #28 Spezial-Module und Auswahl der Quellprogramme)

INHALT

Impressum

Peeker
Magazin für Mikrocomputer
4. Jahrgang 1987
ISSN 0176-9200

Verleger und Herausgeber:
Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften:
Heinz Melcher
Chefredakteur: Ulrich Stiehl (us)
Redaktion: Dagmar Berberich
Anzeigenleitung: Karl M. Dietzow
Anzeigendisposition: Diana Walter

Telefonnummern:
Zentrale: 06221/489-0
Redaktion: 06221/489-352
Anzeigen: 06221/489-206
Abonnement: 06221/489-283
Software: 06221/489-231
Bücher: 06221/489-353
(Bestellungen bitte nur schriftlich)

Abonnement:
Der Abonnent kann seine Bestellung innerhalb von 7 Tagen schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag GmbH, Postfach 10 28 69, 6900 Heidelberg 1, widerrufen. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels). Das Abonnement verlängert sich zu den jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Jahresende schriftlich gekündigt wird. Die Abonnementsgelder werden jährlich im voraus in Rechnung gestellt, wobei bei Teilnahme am Lastschriftabbuchungsverfahren über die Postscheckämter und Bankinstitute eine vierteljährliche Abbuchung möglich ist.
Nichterscheinen infolge höherer Gewalt berechtigt nicht zu Ansprüchen gegen den Verlag.

Hüthig PUBLIKATION

PEEKER

Heft 1/1987

Anschrift:
Dr. Alfred Hüthig Verlag GmbH
Im Weiher 10, Postfach 10 28 69
6900 Heidelberg
Telefon (0 62 21) 4 89-0
Telex 4-6 17 27 hued d
Telefax (0 62 21) 489 279
BTX * 51851 #

Auslieferung für die Schweiz:
Delta-Verlag
Herr R. de Forest
Gugelmattstraße 31
8967 Widen
Telefon 057 / 33 86 86

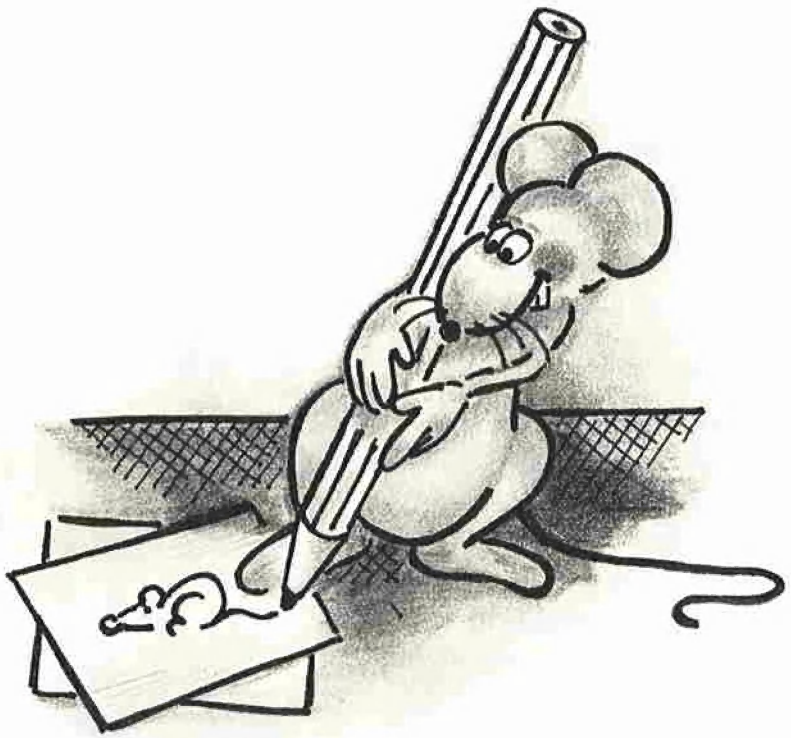
Vertrieb:
Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement Inland DM 75,–, einschl. MwSt und Versandkosten.
Jahresabonnement Ausland DM 75,– plus DM 20,– Versandkosten.
Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089 / 31 90 06 13, Telex 0 522 656
Vertriebsleitung:
Walter Menzel, Tel. (0 62 21) 48 92 80

Bankverbindungen:
Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postgirokonten: Ludwigshafen 4799-673, BLZ 545 100 67; Österreich: Wien 75558 88; Schweiz: Basel 40-24417-4; Niederlande: Den Haag 1 457 28; Italien: Mailand 5 968 92 08; Belgien: Brüssel 07 230 26-85; Dänemark: Kopenhagen 603 4969; Norwegen: Oslo 199 4243; Schweden: Stockholm 5477 76-5
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 02 65 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

Herstellung:
Produktionsleitung: Gunter Sokollek
Gestaltung: Rainer Schmitt
Titelbild: Werner Hable
Satz und Druck:
Heidelberger Verlagsanstalt
Printed in Germany

Ein Shape-Entwicklungspaket

für Grafik-Anfänger



von Wolfgang Landgraf

In Ergänzung zu dem Memory-Spiel mit der Apple-Maus („Mousory") aus Peeker, 7/86, S. 12ff., das insbesondere unter den jüngeren Peeker-Lesern viel Beachtung gefunden hat, wird nachfolgend ein leichtverständliches Shape-Entwicklungspaket vorgestellt, das aus drei lupenreinen Applesoft-Programm-Modulen besteht und beispielsweise für die Entwicklung eigener Mousory-Shapes verwendet werden kann. Besonderer Wert wird dabei auf die Dokumentation der Programme durch kommentierte Variablenlisten gelegt. Die drei Module heißen
TABLE.EDITOR → Table-Editor
SHAPE.EDITOR → Shape-Editor
SHAPE.CREATE → Shape-Creator

I. Der Table-Editor

Programm zur Bearbeitung von Shapetabellen

Dieses Programm hilft einem aus manchen Schwierigkeiten heraus, vor denen man bisher bei der Erstellung von Shapetabellen steht. Benötigt man zum Beispiel nur je ein einziges Shape aus verschiedenen Tabellen, so ist man gezwungen, alle Tabellen an verschiedene Stellen zu laden, um dann umständlich durch Poken der unterschiedlichen Anfangsadressen an Speicherstelle 232 und 233 den Zeiger auf den jeweiligen Tabellenbeginn einzustellen. Oder aber ein bestimmtes Shape hat es einem so angetan, daß man es in eine seiner Tabellen einbauen will. Bisher mußte man sich eine Skizze erstellen und das Shape neu eingeben – alles in allem ein großer Aufwand. Nachdem man dann endlich mit allem fertig war, stellte sich heraus, daß die Tabelle hätte größer sein können. Oder aber das erste Shape hätte besser an die letzte Stelle gepaßt. Man müßte nun die Tabelle komplett neu definieren und mit allem von vorn beginnen, was unnötig zeitraubend ist. Mit dem Table-Editor kennt man diese Probleme nicht mehr. Dieses Programm bietet im einzelnen:

1. Shapetabelle aus Einzelshapes erstellen
2. Shapetabelle bearbeiten
3. Shapes aus ihrer Tabelle herauslösen

Darüber hinaus ruft der Table-Editor das Programm Shape-Editor auf, mit dessen Hilfe Shapes punktweise geändert werden können, oder aber das Programm Shape-Creator – ein Tabelleneditor, mit dem Shapes erst dann abgespeichert werden, wenn man damit zufrieden ist.
Der Table-Editor ist größtenteils menügesteuert. Erklärungen sind auf dem Bildschirm zu sehen. Eventuell auftretende Fehler beim Zugriff auf Disketten werden in einem Teil des Programms abgefangen. Sollte man ein Unterprogramm aus Versehen angewählt haben, so kann man mit dem Ausrufezeichen <!> zum vorhergehenden Programmteil zurückspringen.
In den folgenden Abschnitten werden die einzelnen Programmpunkte ausführlicher besprochen.

1. Shapetabelle aus Einzelshapes

Zwei Möglichkeiten stehen zur Auswahl: Zum einen kann eine Tabelle komplett neu erstellt werden, zum anderen kann man an eine schon existierende Tabelle weitere Shapes anhängen. Sollte in dieser Tabelle kein Platz mehr sein, muß zuvor der Programmpunkt „Shapetabelle editieren" gewählt werden.

1.1. Alte Tabelle editieren

Zuerst wird nach dem Namen der zu bearbeitenden Tabelle gefragt und diese dann ab $6000 in den Speicher geladen. Sollte die gewählte Tabelle voll sein, kehrt das Programm automatisch zum Hauptmenü zurück. Ansonsten wird angezeigt, wieviele Shapes noch Platz finden, und nun wird die Eingabe des Dateinamens des Shapes, welches an die Tabelle angehängt werden soll, erwartet. Das Shape wird ab $4000 geladen, auf dem Bildschirm gezeigt und dann eine Bestätigung der Wahl verlangt. Hat man die Frage bejaht, wird das Shape an die alte Tabelle angehängt. Danach kann man die erweiterte Tabelle auf Diskette abspeichern oder ein weiteres Shape anhängen. Der vergrößerten Tabelle kann ein neuer Name gegeben werden, durch Betätigung der <Return>-Taste wird die Tabelle unter ihrem alten Namen auf Diskette geschrieben.

1.2. Neue Tabelle erstellen

Dieser Programmteil ist mit dem vorher besprochenen identisch, nur daß nicht nach einer alten Tabelle gefragt wird, sondern nach der Anzahl von Shapes, die in der neuen Tabelle Platz finden sollen. Die Tabelle kann dann unter einem beliebigen Namen gespeichert werden. Dem gewählten Namen wird zur Kenntlichmachung als Shapetabelle das Kürzel „ST." vorangestellt; man kann also auch gefahrlos Zahlen als Namen wählen.

1.3. Aufruf des Shape-Creator

siehe unten

2. Shapetabelle editieren

Hier gibt es wiederum zwei Möglichkeiten: Zum einen kann eine beliebige Tabelle verlängert werden, zum anderen können Shapes in der Tabelle vertauscht werden.

2.1. Tabelle verlängern

Bedingt durch die „Anatomie" der Shapetabellen (**Abb. 1**), kann eine volle Tabelle nicht ohne weiteres verlängert werden. Wie man es trotzdem schafft, wird darunter gezeigt. Der Ablauf ist Punkt 1.1 zu entnehmen. Es wird nur gefragt, wieviele Shapes angehängt werden sollen. Dann wird kontrolliert, ob die neue Anzahl größer als 255 ist. Nach erfolgter Manipulation kann die Tabelle auf Diskette geschrieben werden. Der Trick ist folgender: Für jedes neue Shape wird der Tabellenbeginn um 2 Bytes nach unten verschoben, die Abstände der einzelnen Shapes vom Tabellenanfang werden um die Zahl „Neue Shapes * 2" erhöht und abschließend der Entfernungwert „Ende letztes Shape + 1" als Anfangswert für das folgende Shape an seinen Platz in der Tabelle geschrieben.

2.2. Shape-Folge ändern

Eigentlich wird die Reihenfolge der Shapes innerhalb der Tabelle nicht richtig verändert, denn der Zeitaufwand dafür wäre viel zu groß. Der Trick besteht vielmehr darin, die Shapezeiger am Anfang der Tabelle miteinander zu vertauschen. Man sucht sich lediglich zwei Shapes heraus. Das Programm besorgt dann den Rest. Mit den Pfeiltasten wählt man die zu tauschenden Shapes an, mit <Return> wird angenommen. Die Shapes werden zur Kontrolle auf dem Bildschirm nebeneinander gezeigt. Sollte die gewählte Tabelle nur ein einziges Shape besitzen, so wird das Programm automatisch abgebrochen. Vorsicht ist geboten, wenn dieser Programmteil bei einer noch nicht vollständigen Tabelle angewandt wird. Die meisten Shape-Programme, so auch das hier beschriebene, nutzen die Abstandszeiger des ersten Shapes vom Tabellenanfang zur Feststellung der Tabellenkapazität nach der Formel:
(Low-Byte + High-Byte * 256)/2-2
= Tabellenkapazität.
Ist nun das erste Shape vertauscht worden, bevor die Tabelle fertiggestellt wurde, kommt es zu Schwierigkeiten bei der weiteren Bearbeitung, die zum Verlust der Tabellendaten führen können. Deshalb sollte man sich merken, diesen Programmpunkt nur dann anzuwenden, wenn die Tabelle fertig ist und die Umstellung von Shapes den letzten Arbeitsgang darstellt.

3. Einzelshapes erstellen

Wer sich eine Bibliothek aus Einzelshapes zusammenstellen möchte, aus der er nach Lust und Laune neue Tabellen bauen kann, der wähle diesen Programmpunkt. Es spielt keine Rolle, wie groß das Shape ist und womit es erstellt wurde, denn das Programm tut seine Pflicht und trennt das auserwählte Shape säuberlich aus dem Tabellenverbund.
Wie schon oben beschrieben, wird eine Tabelle in den Speicher geladen. Nun kann man durch Betätigung der Pfeiltasten (← und →) in der Tabelle blättern, bis man das Lieblings-Shape erspäht. Mit der <Return>-Taste wird das Shape separiert. Es erscheint rechts neben dem Original zur Kontrolle und kann, unter Namensgebung versteht sich, auf Diskette gesichert werden. Hier stellt das Programm automatisch „SH." für Shape voran, damit man es von einer Tabelle leicht unterscheiden kann.

Änderungsvorschläge

Das Programm läuft unter DOS 3.3 und ProDOS. Wenn es unter ProDOS betrieben wird, sollte man den ProDOS-spezifischen Befehl „PREFIX/" an den Pro-

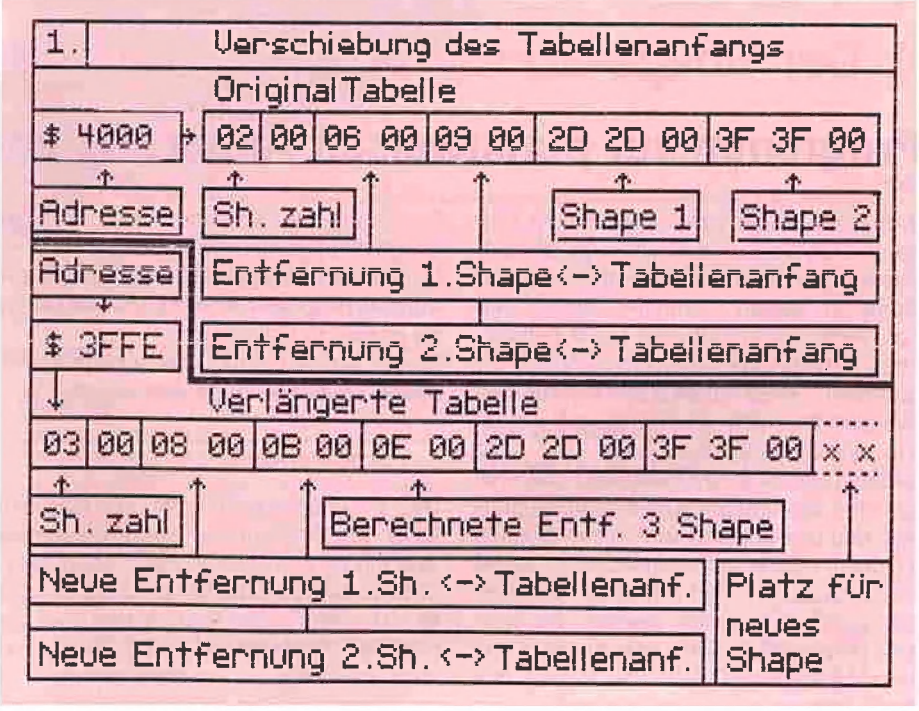


Abb. 1: Aufbau der Shapetabelle

grammbeginn setzen, um gefahrlos Disketten wechseln zu können, ohne die Fehlermeldung „PATH NOT FOUND“ zu riskieren. Der CATALOG-Befehl muß in CAT geändert werden.
Wer möchte, kann das Programm leicht so modifizieren, daß es zwei Laufwerke unterstützt. Um Konflikte mit der ersten Grafikseite zu vermeiden, sollten die REMs weggelassen werden.

Tastenfunktionen

```
<Return> = Annahme der Wahl, Ende eines
           Programmteils
< --> >  = Shapetabelle hinauf
< <-- >  = Shapetabelle hinunter
<!>      = Abbruch eines Programmteils
           bei Irrtum
<?>      = Catalog der Diskette
```

Kommentar zum Programm

```
1000-1150 Zeige "Haupmenü"
1160      Verteilen auf einzelne Pro-
          grammteile
1170-1270 Unterprogramm zur Auswahl der
          Programmteile
1280      Unterprogramm "Shapetabelle
          erstellen"
1290-1330 Zeige Menü "Shapetabelle
          erstellen"
1340-1360 Verteiler für Untermenü
1370-1450 Eingabe des Namens der zu
          erstellenden Tabelle
1460-1500 Eingabe der Shapezahl bei
          neuer Tabelle , Vorbereitung
          des Tabellenanfangs
1510      Bestimmung des Anfangspunktes
          Shape Nummer 1
1520-1560 Eingabe des anzuhängenden
          Einzelshapes
1570-1610 Einlesen der Shapebytes,
          Poken an neue Adresse
1620      Abfrage "Weitermachen ?"
1630-1700 Speichern der Tabelle
          "ST."xxxx auf Diskette
1710      Unterprogramm "Shapetabelle
          editieren"
1720-1750 Zeige Menü "Shapetabelle
          editieren"
1760-1780 Verteiler für Untermenü
1790-1820 Eingabe des Namens der zu
          editierenden Tabelle
1830-1860 Auswertung der Shapezahl in
          der Tabelle
          Bestimmung der Tabellenlänge
1870-1890 Abfrage "Wieviel Shapes
          hinzu?", Kontrolle, ob
          Shapezahl dann größer als 255
1900-1940 Verlagerung des Tabellen-
          anfangs, Aktualisierung der
          Abstandswerte der einzelnen
          Shapes vom Tabellenanfang
1950-1980 Bestimmung Endpunkt des
          letzten Shapes bzw.
          Anfangspunkt für das neue,
          darauffolgende Shape
1990-2000 Unterprogramm "Shapes
          tauschen"
          Kontrolle, ob nur ein Shape
2030-2080 Auswahl der zu tauschenden
          Shapes
2090-2110 Tauschen der Anfangsbytes
2120      Abfrage "Nochmal?"
2130-2200 Speichern der Tabelle
          "ST."xxxx auf Diskette
2210      Unterprogramm "Einzelshape
          erstellen"
2220-2270 Eingabe des Namens der zu
          ladenden Tabelle
2280      Zeiger auf Tabellenanfang,
          Vorbereitung des Einzel-
          shapeanfangs
2290-2390 Auswahl des Einzelshapes im
          Tabellenverbund
2400-2470 Einlesen der Shapebytes,
          Poken an neue Adresse
2480-2500 Abfrage, ob das Shape auf
          Diskette geschrieben
          werden soll
2510-2530 Speichern des Shapes
          "SH."xxxx auf Diskette
2540-2570 Abfrage "Weitermachen?"
2580      Unterprogramm "Einzelshapes
          editieren"
2590-2600 Hier gehört der Aufruf für
          Shape-Editor hin
2610      Unterprogramm "Shape.Creator"
2620-2630 Hier gehört der Aufruf für
          Shape-Creator hin
2640-2680 Catalog Unterprogramm
2690-2710 Programmende
2720-2740 Subroutine "Bildschirm
          löschen"
2750-2810 Subroutine "Fehlerbehandlung"
```

Variablenübersicht

```
D$  = Ctrl-D (DOS-Kommando)
E$  = Eingabevariable
FF  = Errorflag, sorgt für Verteilung
      nach Fehlerroutine
FH  = Flag zur Unterscheidung Haupt-,
      Untermenü
FM  = Flag für Verteilung nach Catalog-
      routine
HH  = Hilfsflag im Unterprogramm Tabelle
      editieren
HT  = Horizontalposition Auswahlcursor
I   = Schleifenvariable
J   = Wert des aktuellen Shapebytes
N   = Shapezahlvariable
N1  = Aktuelle Shapenummer
N(I)= 1. und 2. Shape beim Tausch
N$  = Eingabestring, Name Tabelle
      oder Shape
N1$ = Aktueller Tabellenname
Q$  = "!" Variable für Programmteil-
      abbruch
SB  = Beginn des aktuellen Shapes in
      der Tabelle
SE  = Ende des aktuellen Shapes in der
      Tabelle
SL  = Länge der zu editierenden Tabelle
SZ  = Momentane Zahl der Shapes in
      der Tabelle
S1  = High-Byte Shape-Anfang
S2  = Low- Byte Shape-Anfang
S3  = High-Byte Shape-Ende
S4  = Low- Byte Shape-Ende + 1
TA  = Tabellenanfang
TB  = Beginn des ersten Shapes in
      der Tabelle
W1  = Low- Byte 1.Tauschshape
W2  = High-Byte 1.Tauschshape
W3  = Low- Byte 2.Tauschshape
W4  = High-Byte 2.Tauschshape
```

II. Der Shape-Editor

Programm zur punktweisen Änderung von Shapes

Wer schon einmal ein Shape oder gar eine ganze Tabelle nur mit Hilfe von Bleistift, Papier und dem BASIC-Handbuch von Apple in seinen Computer eingegeben hat, weiß diverse auf dem Markt befindliche Shapetable-Editierprogramme zu schätzen. Leider haben alle mir bekannten Programme einen gravierenden Nachteil: Ein einmal definiertes Shape kann nicht beliebig verändert werden, weil die Programme die eingebenen Cursorbewegungen und das Setzen von Punkten direkt in Zeichenvektoren umwandeln. Maximal der letzte gesetzte Punkt kann eventuell wieder zurückgenommen werden. Es kann also passieren, daß ein oder mehrere aus Versehen doppelt definierte Punkte das Aussehen eines Shapes ruinieren, wenn man den XDRAW-Befehl anwendet.
Der Startpunkt, von dem aus sich das Shape aufbaut, ist unveränderbar.
Soll ein Shape spiegelbildlich dargestellt werden, bleibt einem nichts anderes übrig, als es neu zu definieren.
Diese leidigen Probleme gehören mit Shape-Editor der Vergangenheit an.

1. Vorbemerkung

Das Programm läuft unter ProDOS und DOS 3.3. Änderungen des Programmes sind mit Vorsicht zu genießen. Da es nämlich mit seinen Variablen so groß ist, daß es nicht mehr in den Bereich zwischen der normalen Startadresse $0800 und der ersten Grafikseite als Arbeitsseite paßt, wird es in den Bereich zwischen $4000 und $5FFF geladen. Die Shapetabelle, aus der die zu bearbeitenden Shapes stammen, wird ab $6000 plaziert. Bei Änderungen im Programm, die zu einer Verlängerung führen, kann es zu Konflikten mit dem Anfang der Shapetabelle kommen, d.h. die später eingeladene Tabelle löscht eventuell Teile des BASIC-Programms. Nun könnte man die Anfangsadresse der Shapetabelle weiter nach oben in den Speicher legen, aber dann gibt es bei sehr langen Tabellen Schwierigkeiten mit dem jeweils verwendeten Betriebssystem.

Unter ProDOS kann man Tabellen mit einer Länge von maximal 13824 Bytes einladen. Normales DOS 3.3 bietet denselben Platz. Ideal wäre es, den Shape-Editor unter einem in die Language-Card „gemovten“ DOS zu betreiben (s. Peeker, Heft 2/

»peeker« schenkt Ihnen...

. . . den praktischen Disk-Locher mit dem Sie die Speicherkapazität Ihrer Disketten im Nu verdoppeln können!

Überzeugen Sie jetzt einfach Ihre Freunde und Bekannten von den Vorzügen eines »peeker«-Abonnements«. **Sie bekommen als Dank für einen neuen Abonnenten diesen superpraktischen Disk-Locher!**

Füllen Sie den Bestellcoupon aus und ab zur Post!

Bestellcoupon

Ich habe den neuen Abonnenten geworben und erhalte kostenlos den Disk-Locher.

Name, Vorname

Straße, Postfach

PLZ, Ort

X

Datum, Unterschrift

Ich bin der neue Abonennt. Bitte liefern Sie mir bis auf Widerruf, zumindest aber für 1 Jahr, »peeker« zum Jahresbezugspreis von DM 75,– (Ausland plus DM 20,– Porto) an folgende Anschrift:

Name, Vorname

Straße, Postfach

PLZ, Ort

X

Datum, Unterschrift

Gewünschte Zahlungsweise
☐ gegen Rechnung
☐ bargeldlos durch Bankeinzug

Konto-Nr. Bankleitzahl

Geldinstitut

Vertrauensgarantie:
Diese Bestellung kann ich innerhalb einer Woche bei Dr. Alfred Hüthig Verlag GmbH, Im Weiher 10, 6900 Heidelberg 1 widerrufen. Zur Wahrung der Frist genügt die rechtzeitige Absendung. Ich bestätige die Kenntnisnahme mit meiner Unterschrift:

X

2. Unterschrift

Peeker 1/87

Coupon ausschneiden und einsenden an:

»peeker«
Abonnementservice
Im Weiher 10
6900 Heidelberg 1

Hüthig
PUBLIKATION

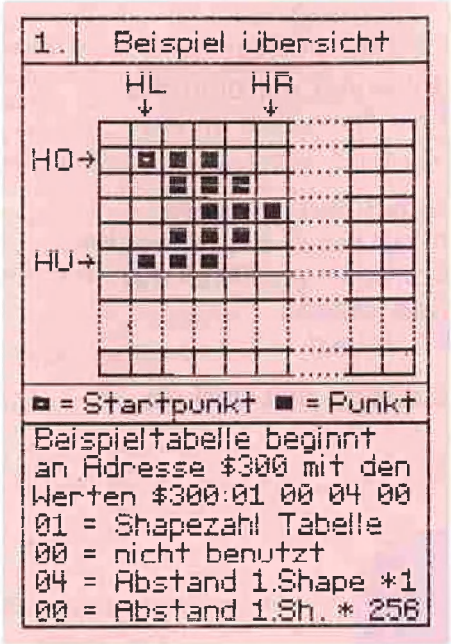


Abb. 2: Beispiel-Shape

86), da nun die Tabellen 25545 Bytes lang sein dürfen.
Wer sich für ProDOS entscheidet, sollte den ProDOS-spezifischen Befehl „PREFIX/" zur Abschaltung des Präfixes an den Anfang des Programmes setzen, damit die Disketten gewechselt werden können. Ebenso muß der „CATALOG"-Befehl in „CAT" geändert werden.
Der Shape-Editor läßt sich sehr gut für Animationen mit Shapes einsetzen, etwa um Bewegungen zu simulieren, wenn man das „Vorgänger-Shape" einlädt, es verändert, abspeichert und dann die Einzelshapes in einer Tabelle zusammenfügt. Dies kann mit dem Table-Editor realisiert werden.
Der Shape-Editor ist für Shapes in der Größe von 21 * 21 Bildpunkten konzipiert. Der Startpunkt kann innerhalb dieser Matrix frei gewählt werden.
Normalerweise wird das Programm vom Table-Editor aufgerufen; es ist aber auch allein lauffähig. Dann muß Zeile 2360 folgendermaßen geändert werden:
2360 POKE 216,0 : END

2. Programmablauf

Zuerst wird nach dem Namen einer Shapetabelle gefragt, aus der das zu editierende Shape stammt. Man kann aber ebensogut ein Einzelshape aufrufen. Mit den Pfeiltasten wählt man das Shape aus, mit <Return> wird das Shape übernommen. Nach dem Aufbau des Bildschirms (kleiner Rahmen und großer Rahmen mit Skala), kann das ausgewählte Shape mit den <I-J-K-M>-Tasten im kleinen Rahmen hin und her bewegt werden. Die Betätigung der <S>-Taste spiegelt das Shape, nochmaliger Druck auf <S> stellt den normalen Zustand wieder her. Das Shape im kleinen Rahmen ändert sich dadurch aber nicht, die Spiegelung kommt nur bei der Übernahme in den großen Rahmen zum Tragen. Zur Kontrolle wird die Bedeutung der jeweiligen Tasten im Textfenster wiedergeben. Die <P>- und <L>-Tasten haben noch keine Funktion. Die <Return>-Taste beendet den Programmteil, d.h. alle Punkte, die sich innerhalb des kleinen Rahmens befinden, werden in den großen Rahmen übertragen. Nun baut sich vor den staunenden Augen die vergrößerte Version des Urshapes im Editierrahmen auf, je nach ausgewählter <S>-Funktion gespiegelt oder seitengleich. Ebenso wird ein Shape im Maßstab 1:1 im kleinen Rahmen unten rechts gezeigt. Die Bewegungen des Editiercursors werden, wie oben schon erwähnt, mit <I-J-K-M> gesteuert. Ein Punkt wird mit <L> gelöscht und mit <P> gesetzt. Sollte man einen nicht vorhandenen Punkt erneut löschen wollen, ignoriert das Programm den Tastendruck. Das gleiche gilt für das Setzen eines schon gezeichneten Punktes. Im kleinen Rahmen rechts kann man die Auswirkung auf das Shape direkt mitverfolgen. Mit <S> läßt sich der Startpunkt bestimmen.* Das kann man so oft machen, wie man will, denn nur der zuletzt bestimmte Startpunkt ist gültig. Wird kein Startpunkt definiert, nimmt das Programm automatisch die Mitte des Editierrahmens (Koordinate 11,11) als Anfangspunkt für das neue Shape an. Nach <Return> wird das Shape berechnet und zur Kontrolle unterhalb des alten Shapes gezeigt. Ein blinkendes Kreuz bezeichnet die Position des Startpunktes. Das Shape kann nun unter irgendeinem Namen auf Diskette abgespeichert werden. Das Programm stellt dem Namen automatisch „SH." für Shape voran. Es können also auch Zahlen als Namen eingegeben werden. Danach wird gefragt, ob man mit einer neuen Tabelle oder einem neuen Shape weitermachen oder das Programm beenden will.
Jeder Programmabschnitt kann durch Betätigung von <Esc> abgebrochen werden, um zum vorhergehenden Teil des Programms zu kommen.

* Bei dem Programm auf der Sammeldiskette wurde zusätzlich der Befehl <W> zum Invertieren des Shapes eingefügt.

3. Technisches

Der kleine Rahmen umschließt einen Bereich, der in den Zeilen 1270-1410 byteweise ausgelesen wird. Diese Daten werden in der Subroutine von Zeile 2690-2950 in das 21 * 21 Punkt großes Feld B%(x,y) übertragen. Nun kann im nächsten Programmschritt leicht jede Stelle des Feldes mit dem Cursor angefahren werden, um dort Punkte zu löschen oder zu setzen. Auch der neue Startpunkt wird im Feld markiert. Nun berechnet das Programm die Grenzen des Shapes, damit nicht zuviel Speicherplatz verbraucht wird. Dann werden zuerst Vektoren vom Startpunkt bis in die obere linke Ecke des neuen Shapes in den Speicher geschrieben.

Das Shape wird nun Zeile für Zeile innerhalb der Shapegrenzen „abgetastet", und die entsprechenden Vektoren werden an ihre Adresse gepokt. Eine Tabelle der entsprechenden Werte stellt **Abb. 3** dar. Ein so erzeugtes Shape benötigt natürlich mehr

2. Bedeutung der Vektoren, C = 1 C = 2 C = 3

Vektor	Punkt	Bewegung	Wert	Wert	Wert
↑	nein	hoch	0	0	X
→	nein	rechts	1	8	64
↓	nein	runter	2	16	128
←	nein	links	3	24	192
↕	ja	hoch	4	20	X
●→	ja	rechts	5	40	X
↕●	ja	runter	6	48	X
←●	ja	links	7	56	X

An Position C = 3, keine Bewegung nach oben, es darf auch kein Punkt gesetzt werden (X).

Abb. 3: Bedeutung der Vektoren

Platz als sein Vorbild. Deshalb kann es nicht an seinen alten Platz in der Tabelle zurückgeschrieben werden, denn die erforderlichen Berechnungen der Abstände der nachfolgenden Shapes sowie deren Verschiebung würde zuviel Zeit in Anspruch nehmen und wäre darüber hinaus recht kompliziert.
Jedes neue Shape wird im Bereich ab $0800 erzeugt und auch mit den entsprechenden Werten auf Diskette geschrieben. Vorsicht also, ein solches Shape ohne A-Parameter einfach mit dem Befehl „BLOAD SH.xxxx“ von Diskette zu laden. Ein schon im Speicher befindliches BASIC-Programm würde sonst zerstört. Bitte also z.B. „BLOAD SH.xxxx, A$6000“ eingeben und die Speicherstellen 232 und 233 dezimal entsprechend auf den Anfang der Tabelle einstellen. Wer es nicht weiß: Diese beiden Speicherstellen auf der Zero-Page (auch $00E8 und $00E9 hexadezimal) sind die Zeiger für den Beginn einer Shapetabelle, wobei 232 das Low-Byte beherbergt und 233 für das High-Byte zuständig ist.

3. Umsetzung in V.

Vektoren	Reihe
→ → → → ↓	1
↓ ← ← ← ←	2
→ → → → ↓	3
↓ ← ← ← ←	4
→ → → Ende	5

Abb. 4: Beispiel-Shape in Vektoren

Tastenfunktionen

```
( <- )    = Shapeauswahl Tabelle hinunter
( -> )    = Shapeauswahl Tabelle hinauf
(Esc)     = Abbruch eines Programmteils
(Return)  = Annahme
( L )     = Punkt löschen
( P )     = Punkt setzen
( S )     = Spiegelung, Startpunkt setzen
( ? )     = Catalog der Diskette
( ! )     = Programmabbruch
```

Kommentar zum Programm

```
1000-1040 Programm über 1. Grafikseite
          laden, Editier-Shapes laden,
          evtl. Präfix abstellen
1050-1230 Auswahl der Tabelle und
          des Shapes
1240      Aufruf Subroutine
          "Bildschirmaufbau"
1250-1260 Aufruf Subroutine
          "Shape- und Cursor-Steuerung"
1270-1410 Einlesen der Daten aus
          kleinem Rahmen und
          Aufruf Subroutine
          "Feld erstellen"
1420      Aufruf Subroutine "Shape- und
          Cursor-Steuerung"
1430-1590 Berechnung der Shapegrenzen
1600-1850 Vektoren bis zur oberen
          linken Ecke poken
1860-1900 Shapeanfangswerte eingeben,
          Bildanfang berechnen
1910-2240 Zeichenvektoren des Shapes
          berechnen und poken
2250-2360 Abfrage "Shape auf Diskette,
          Programmende" und
          Speicherung "SH.xxxx"
          auf Diskette
2370-2680 Subroutine "Shape- und Cursor-
          Steuerung"
2690-2950 Subroutine "Feld erstellen".
          Gelesene Shapebytes in
          Feld B%(X1,X1) eintragen
2960-3120 Subroutine "Bildschirmaufbau"
3130-3200 Fehlerbehandlung
```

Variablenübersicht

Um Speicherplatz zu sparen, da das Programm und seine Variablen in den Bereich zwischen $4000 und $5FFF passen müssen, haben einige Variablen in unterschiedlichen Programmteilen verschiedene Aufgaben. Der gültige Bereich wird in Klammern hinter der jeweiligen Funktion angegeben. Sollten keine Klammern vorhanden sein, gilt die Funktion für das ganze Programm.

```
A       = Gelesener Shapebyte-Wert
          Shapenummer aus Daten
          (2960-3120)
A(x)    = Hilfsfeld zur Umrechnung der
          gelesenen Werte "A"
          in das Feld B%(x,y)
B       = Indikator für Shapegrenzen
          x-Koordinate aus Daten
          (2960-3120)
B%(x,y) = Feld der einzelnen Shapepunkte
          (21 * 21)
C       = Schleifenzähler bei der
          Berechnung der
          Einzelshapebytes
          y-Koordinate aus Daten
          (2960-3120)
CX      = x-Koordinate des Startpunktes
          auf dem Bildschirm
CY      = y-Koordinate des Startpunktes
          auf dem Bildschirm
D$      = Ctrl-D (DOS-Kommando)
E$      = Eingabevariable
FF      = Errorflag zur Verteilung
          nach aufgetretenem Fehler
FH      = Hilfsflag (1 = Shapezeichnung,
          0 = Cursor)
H       = Hilfsvariable Spiegelung,
          rettet "I" bei Spiegelung
HL      = Linke Grenze des Shapes in
          Feld B%(x,y)
HR      = Rechte Grenze des Shapes in
          Feld B%(x,y)
HO      = Obere Grenze des Shapes in
          Feld B%(x,y)
HU      = Untere Grenze des Shapes in
          Feld B%(x,y)
HS      = Flag Spiegelung (1 = spiegeln,
          0 = nicht spiegeln)
I       = Schleifenzähler
          Indikator für Shapeanfang
          (1860-1900)
J       = Schleifenzähler
K       = Schleifenzähler
M       = Berechnungswert für die
          x-Koordinate "X"
N       = Shapenummer in der Tabelle
N1      = Aktuelle Shapenummer
N$      = Name der Shapetabelle
          (1050-1110)
          Name des neuen Shapes
Q$      = Abbruch-Variable "!"
SX      = x-Koordinate des Startpunktes
          im Feld B%(x,y)
SY      = y-Koordinate des Startpunktes
          im Feld B%(x,y)
TA      = Shapetabellenanfang
U       = Aktuelle Adresse im neuen Shape
          während der Berechnung
X       = x-Koordinate für Editier-Cursor
XA      = Alte x-Koordinate zur Zeichnung
          von Shape oder Cursor
XN      = Neue x-Koordinate zur Zeichnung
          von Shape oder Cursor
X1      = x-Koordinate im Feld B%(x,y)
X2      = x-Koordinate im Feld B%(x,y),
          rechte Grenze
YA      = Alte y-Koordinate zur Zeichnung
          von Shape oder Cursor
YN      = Neue y-Koordinate zur Zeichnung
          von Shape oder Cursor
Y1      = y-Koordinate im Feld B%(x,y)
Y2      = y-Koordinate im Feld B%(x,y),
          untere Grenze
```

4. Beispiel zur Berechnung der Shape-Bytes

Ve.	C	Wert	Adr.	Betrag	Ve.	C	Wert	Adr.	Betrag
→	1	5	$304	5	→	1	5	$309	5
→	2	40	$304	45	→	2	40	$309	45
→	1	5	$305	5	↓	1	6	$30A	6
→	2	8	$305	13	←	2	24	$30A	30
↓	3	128	$305	141	←	1	7	$30B	7
←	1	3	$306	3	←	2	56	$30B	63
←	2	56	$306	59	←	1	7	$30C	7
←	1	7	$307	7	↓	2	16	$30C	23
←	2	56	$307	63	→	1	5	$30D	5
↓	1	2	$308	2	→	2	40	$30D	45
→	2	8	$308	10	→	1	5	$30E	5
→	3	64	$308	74	E	0	0	$30F	0

Ergebnis dez.: 45,141,59,63,74,45,30,63,23,45,05,00

Ergebnis hex.: 2D 8D 3B 3F 4A 2D 1E 3F 17 2D 05 00

Abb. 5: Endgültig berechnetes Beispiel-Shape

III. Der Shape-Creator

Shapetable-Editor für Shapes mit 21 * 21 Bildpunkten

Viele werden jetzt sagen, „Schon wieder ein Shapetable-Programm!". Doch halt, erst lesen, dann kritisieren. Der Shape-Creator ist anders konzipiert als seine Vorgänger. Shapes auf dem Bildschirm erstellen, das können alle Programme, doch wie gehen sie dabei vor?
Der Benutzer gibt über Tastatur die Bewegungen und Punkte ein, das Programm rechnet diese sofort in Shapedaten um, und am Ende ist die Tabelle fertig. Was aber, wenn man sich bei der Eingabe vertan hat, wenn das Shape nun doch nicht so aussieht, wie man es sich vorgestellt hat? Änderungen sind nicht mehr möglich, da jedes Shape während seines Entstehens im Speicher abgelegt wurde.
Hier setzt der Shape-Creator ein. Man kann jedes Shape so lange bearbeiten, bis es die Form hat, die der Programmierer sich wünscht, erst dann werden die Daten im Speicher abgelegt.
Der zweite Vorteil des Shape-Creators liegt darin, daß man sich eines zuvor erstellten Shapes bedienen kann, um ein neues Shape zusammenzubasteln, indem man das Editierfeld nicht löscht. So bleiben alle gesetzten Punkte erhalten, wenn ein neues Shape gezeichnet wird. Animation mit Shapes wird dadurch wesentlich erleichtert.

1. Programmablauf

Zuerst wird gefragt, ob man eine neue Tabelle erstellen oder eine vorhandene Tabelle bearbeiten will. Sollte man sich dafür entscheiden, eine Tabelle zu bearbeiten, muß man den Namen eingeben. Nun wird überprüft, ob die Tabelle noch Platz für weitere Shapes bietet. Ist dies nicht der Fall, fragt das Programm automatisch nach einer neuen Tabelle.
Hat man sich für eine neue Tabelle entschieden oder die zu bearbeitende Tabelle ist noch nicht komplett gefüllt, wird der Bildschirm aufgebaut.
Zu sehen ist der große Editierrahmen sowie rechts davon zwei kleinere Rahmen, einer in Weiß, der andere in Schwarz. In ihnen kann man sehen, wie das Shape in der normalen Größe aussehen wird, das heißt SCALE = 1. Unterhalb des Grafikfensters ist die Bedeutung der einzelnen Tasten zu sehen. Man kann nun damit beginnen, sein Shape zu konstruieren. Der Startpunkt, von dem aus sich das Shape später aufbaut, ist in der Mitte des 21 * 21 Punkte großen Feldes. Er kann aber willkürlich verändert werden. Hieran zeigt sich, daß der Shape-Creator ursprünglich zur Zeichnung der MOUSORY. SETs (aus Peeker, Heft 7/86) geplant war.
Durch Betätigung der <Return>-Taste wird das Shape berechnet, und zwar in folgender Reihenfolge:

1. Berechnung der Shapegrenzen
2. Vom Startpunkt bis zur berechneten linken oberen Ecke des Shapes
3. Ab dort zeilenweise im Wechsel (von links nach rechts, dann von rechts nach links) bis zur untersten Shapezeile

Zur Kontrolle erfolgt die erneute Abfrage, ob das Shape an die aktuelle Tabelle angehängt werden soll. Wer mit seinem Shape nicht ganz zufrieden ist, kann hier seine letzte Chance nutzen. Selbst wenn er die Frage verneint, ist seine vorherige Arbeit nicht umsonst gewesen, da er den Editierrahmen unverändert für den nächsten Versuch übernehmen kann. Die von mir bisher benutzten Shape-Editoren kannten diese Möglichkeit nicht.
Wer sein Shape an die Tabelle anhängt, kann dies selbstverständlich auch nutzen, ansonsten ist der Rahmen natürlich zu löschen.
Jedes schon definierte Shape kann durch Betätigung der <Z>-Taste auf dem Bildschirm gezeigt werden. Mit den Pfeiltasten (→, ←) wird in der Tabelle „geblättert", mit <Return> geht es zurück zum Hauptprogramm.
Mit <T> kann die aktuelle Tabelle nun auch auf Diskette geschrieben werden. Wer eine Tabelle neu eingegeben hat, muß ihr einen Namen geben. Das Programm stellt automatisch „ST." für Shapetabelle voran. Sollte es sich um eine bearbeitete alte Tabelle handeln, muß nur die <Return>-Taste betätigt werden, um die Tabelle unter ihrem alten Namen abzuspeichern. Der Shape-Creator kann nur über diesen Programmteil verlassen werden, um Datenverluste durch versehentliches Verlassen ohne Abspeicherung der aktuellen Tabelle zu vermeiden.

Änderungsvorschläge

Das Programm ist in der vorliegenden Version sowohl unter DOS 3.3 als auch unter ProDOS lauffähig. Wer mit ProDOS arbeitet, sollte für das gefahrlose Wechseln von Disketten an den Anfang des Programms den Befehl „PREFIX/" zur Abschaltung der aktuellen Präfixes stellen. Auch der „CATALOG"-Befehl in Zeile 3000 sollte in „CAT" umgeändert werden.

Shape-Creator füllt die Lücke, die die Programme Shape-Editor und Table-Editor hinterlassen haben. Versierte Programmierer können diese drei Teile eventuell zu einem einzigen Programm kombinieren. Dann hätte man einen Super-Shape-Editor zur Verfügung, der keine Wünsche mehr offen ließe.
Das Programm ist eigenständig, das heißt, es muß nicht vom Table-Editor aus aufgerufen werden.
Zum Abschluß muß ich leider einen kleinen Wermutstropfen auf die aufkeimenden Hoffnungen vom komfortablen Shape-Editieren gießen. Der Shape-Creator eignet sich nur zur Erstellung von Shapes, die mit dem SCALE-Faktor von 1 wiedergegeben werden sollen. Da die Berechnung der Shapewerte zeilenweise erfolgt, kommt es zu Verzerrungen bei der Vergrößerung.

Kommentar zum Programm

```
1000-1100 Programmbeginn an $4000
          verschieben
1110-1310 Auswahl der Shapetabelle,
          Bestimmung der Shapezahl
1320-1390 Startbedingungen, Ausruf
          Subroutine "Bildschirmaufbau"
1400-1580 Tastaturabfrage, Auswertung
1590-1760 Shapegrenzen festlegen
1770-2050 Werte bis zur linken oberen
          Ecke des Shapes poken
2060-2380 Werte Zeile für Zeile bestimmen
2390-2460 Zeigen des Shapes mit
          Startpunkt, Anhängen Ja/Nein
2470-2550 Hauptmenü
2560-2660 Zeigen der bisher definierten
          Shapes
2670-2760 Tabelle auf Diskette speichern
2770-2910 Subroutine Bildschirmaufbau
2920-2980 Fehlerbehandlung
2990-3000 Catalog der Diskette
```

Variablenübersicht

```
A        = Zeilenflag Rechts-Links-Wechsel
           bei Shapeberechnung
B        = Plot-Noplot Flag bei Shape-
           berechnung
B%(x,y)  = Feld von 21 * 21 Punkten
C        = Schleifenwert bei Shape-
           berechnung
CX       = x-Wert Startpunkt-Cursor
CY       = y-Wert Startpunkt-Cursor
D$       = Ctrl-D (Dos-Kommando)
E$       = Eingabevariable
FF       = Fehlerflag, übernimmt Verteilung
           nach aufgetretenem Fehler
HL       = Linke Shapegrenze
HO       = Obere Shapegrenze
HR       = Rechte Shapegrenze
HU       = Untere Shapegrenze
I        = Schleifenvariable
```

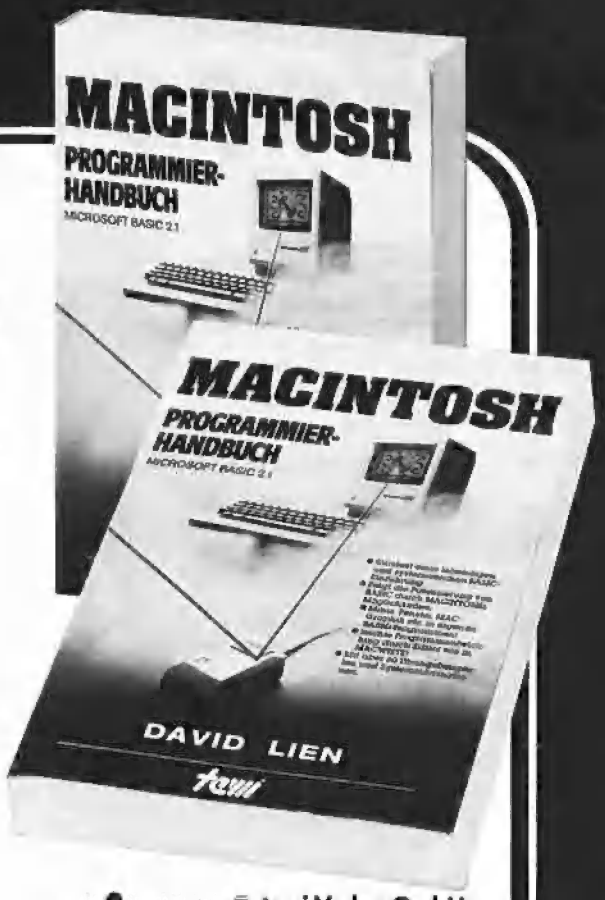
MS BASIC ist eine höchst moderne und leicht erlernbare Programmiersprache!

210 reservierte Begriffe...Programmsynthese aus Modulen (lokale Variablen, Wertübergaben mit COMMON)...Nachladen von Segmenten (CHAIN, mit Parameterübergabe)...Einbinden von MAC-Funktionen (Graphik, Maus, Fenster, Knöpfe usw.)...Fremddateizugriff (von MS-BASIC auf Dateien von z. B. MULTIPLAN, MACPAINT, WORD)...Programmablaufsteuerung (Ereigniserfassung wie ON TIMER, ON MOUSE)...Peripheriebefehle (wie COMI für DFÜ)...strukturierte BASIC-Programmierung ohne Zeilennummern...Gleitkommaarithmetik.....

MACINTOSH und MS BASIC bilden eine komfortable Programmierumgebung!

Mehrfachfenster für Simultanbeobachtung (z. B. Fenster 1: Hauptprogramm, Fenster 2: Unterprogramm; oder Fenster 1: Listing, Fenster 2: Ergebnisse)...Mausedition wie in Textprogrammen...Collagetechnik (Programmabschnitte ausschneiden und versetzen)...

HIER DER KURSTEXT einer lebendigen und systematischen BASIC-Einführung von dem US-Professor David Lien. Ein Text für das Selbststudium, das in anspruchsvolle BASIC-Programmierungen mit den Funktionen des MACINTOSH mündet. Ideal als Schultext.
450 Seiten, Softcover, DM 59,–

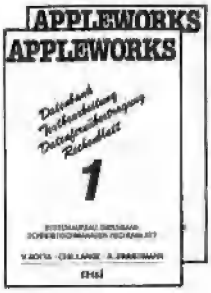


tewi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weitere te-wi-Bücher

M68000 FAMILIE, 2 Bd.
Hilf/Nausch, ges. 968 Seiten
Einzige Motorola-authentische Darstellung von CPU-68000-Architektur, Programmierung, Systemaufbauten. Behandelt alle 68000-Bausteine sowie 68020, 68881.
Bd 1, Grundlagen + Architektur, 568 Seiten, DM 79,–
Bd 2, Anwendung und Bausteine, 400 Seiten, DM 69,–

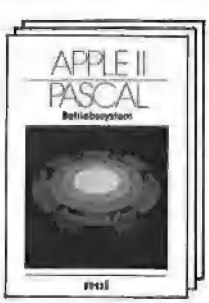
Das APPLE II/II+/IIe/IIc-Handbuch
L. Poole
Erst mit Hilfe dieses Leitfadens werden Sie Ihren Apple II erfolgreich einsetzen, denn Text und Bildmaterial gehen weit über das hinaus, was herstellerseitig an Literatur angeboten wird.
Neu überarbeitet und jetzt um die spezifischen Eigenheiten der Modelle **IIe** und **IIc** erweitert. 472 Seiten, Softcover, DM 66,–



APPLEWORKS integriert in APPLE II, IIe, IIc die Funktionen eines modernen Schreibtisches: Textverarbeitung, Datenbank, Rechenblatt, Datenfernübertragung. Sämtliche System/Anwendungsfragen in 2 Bänden.
Von Botta/Lange/Zimmermann, je 264 Seiten und je DM 49,–

DAS C-BUCH. **NEU**
Textbuch für C-Kurse und C-Anwendungen auf PCs. Beschreibt sämtliche Konstrukte der C-Sprache unter den Betriebssystemen MS DOS, CP/M, ISIS, UNIX und für die C-Compiler von MS, DR, LATTICE, INTEL. Didaktisch und typographisch außergewöhnlich. Mit über 100 lauffähigen Beispielprogrammen für PCs. Zeigt Realisierungen neuester Softwarestrategien in „C".
Von Herold/Unger. Herbst 86.
Etwa 500 Seiten. Softcover. DM 79,–

UMWELTDYNAMIK
30 Programme für kybernetische Umwelterfahrungen auf allen BASIC-Rechnern. Das Buch enthält beides: Ein Programmsystem zur Simulation eigener Problemformulierungen und 29 kommentierte Modellbeispiele wie Baumsterben, Heizungsbedarf, Nahrungsketten usw. Prospekt anfordern.
Von Hartmut Bossel, 480 Seiten, Softcover, DM 59,–



Erstes deutsches Referenzwerk sämtlicher Befehle und Systemroutinen von Apple II, IIplus, IIe
APPLE II PASCAL
Betriebssystem, 272 S., DM 49,–
Sprache, 216 S., DM 39,–
Pascal 1.2 Addendum, 112 S., DM 36,–

Grundlagenbuch, Bestseller APPLE II PASCAL.
Eine praktische Anleitung, 544 S., DM 59,–

Im 46v1

```
J       = Schleifenvariable
M       = Variable beim Anzeigen von
          Shapes
N       = Tabellenkapazität
N1      = Aktuelles Shape beim Anzeigen
N$      = Name Shapetabelle
N1$     = Aktueller Name Shapetabelle
SN      = Aktuelle Zahl der Shapes
          in der Tabelle
SX      = x-Wert Startpunkt im Feld
          B%(x,y)
SY      = y-Wert Startpunkt im Feld
          B%(x,y)
TA      = Tabellenanfang ($6000 oder
          24576 dezimal)
TN      = Startadresse des aktuellen
          Shapes
U       = Aktuelle Adresse innerhalb
          der Shapetabelle
X1      = x-Wert innerhalb B%(x,y)
X2      = x-Wert bei Berechnung der
          Shapegrenzen
XA      = Alter x-Wert des Editiercursors
XN      = Neuer x-Wert des Editiercursors
XM      = Maximale x-Position bei
          Bildschirmaufbau
Y1      = y-Wert innerhalb B%(x,y)
Y2      = y-Wert bei Berechnung der
          Shapegrenzen
YA      = Alter y-Wert des Editiercursors
YN      = Neuer y-Wert des Editiercursors
YM      = Maximale y-Position bei
          Bildschirmaufbau
```

Kurzhinweise

1. Zweck:
Eingeben, Ändern und Zusammenfügen von Shapetabellen
2. Konfiguration:
Apple II+/e/c; DOS 3.3 oder ProDOS
3. Test:
RUN SHAPE.CREATE usw.
4. Sammeldisk:
TABLE.EDITOR
SHAPE.EDITOR
SHAPE.CREATE
ST.EDIT
5. Sonstiges:

Zum Editieren können sie u.a. die Mousory-Shapes von Sammeldisk #19 verwenden.

ST.EDIT

(Tabelle mit 10 Shapes)

```
$0300: 0A 00 16 00 1D 00 24 00
$0308: 2C 00 2F 00 36 00 3F 00
$0310: 45 00 4C 00 62 00 24 17
$0318: 56 3E 0D 04 00 1C 28 15
$0320: 96 3F 24 00 23 2C 35 36
$0328: 3E 27 04 00 27 25 00 23
$0330: 2C B5 AB 1E 27 00 35 3F
$0338: 27 25 27 2D 2E 24 00 C0
$0340: 6B B1 FA 23 00 03 68 1E
$0348: 1E 0D 04 00 24 24 2D 2D
$0350: 36 3F 6F 11 6E 24 24 96
$0358: 0A 2D 6D 21 24 3C 6F 29
$0360: 04 00 24 24 15 15 15 15
$0368: 24 24 0D 2D 2D DF 13 2E
$0370: FD 32 2D 2D 0D 24 24 4D
$0378: 31 36 3E 3F 04 00 00 00
```

TABLE.EDITOR

```
100 REM ** Table.Editor **
101 REM ** Wolfgang Landgraf 10.04.86 **
102 D$ = CHR$ (4):Q$ = "!"
103 IF PEEK (104) < > 8 THEN POKE 103,1: POKE 104,8: POKE
    2048,0: PRINT : PRINT D$"RUN TABLE.EDITOR"
104 TEXT : HOME :VT = 6:HT = 7:V = 0:W = 1
105 FOR I = 0 TO 39: PRINT "-";: NEXT
106 PRINT TAB( 13)"Shape-Table-Plus"
107 FOR I = 0 TO 39: PRINT "-";: NEXT
108 POKE 34,4
109 VTAB 6: HTAB 10: PRINT "Shapetabelle erstellen"
110 VTAB 8: HTAB 10: PRINT "Shapetabelle editieren"
111 VTAB 10: HTAB 10: PRINT "Einzelshapes erstellen"
112 VTAB 12: HTAB 10: PRINT "Einzelshapes editieren"
113 VTAB 14: HTAB 10: PRINT "Catalog Disklaufwerk 1"
114 VTAB 16: HTAB 10: PRINT "Shape-Table-Plus->Ende"
115 GOSUB 1170:FM = 1
116 ON (VT - 4) / 2 GOTO 1290,1720,2220,2590,2650, 2700
117 VTAB 19: FOR I = 0 TO 39: PRINT "-";: NEXT
118 VTAB 21: FOR I = 0 TO 39: PRINT "-";: NEXT
119 VTAB 6: HTAB 7: PRINT "->"
120 VTAB 20: HTAB 10: PRINT "<- Wahl ->,dann <RET>"
121 VTAB 20: HTAB 31: GET E$
122 IF E$ < > CHR$ (8) AND E$ < > CHR$ (21) AND E$ < >
    CHR$ (13) THEN 1210
123 IF E$ = CHR$ (13) THEN RETURN
124 VN = VT + (2 * (E$ = CHR$ (21))) - (2 * (E$ = CHR$
    (8)))
125 IF VN < 6 THEN VN = 10 + 2 * V + 6 * W
126 IF VN > 10 + 2 * V + 6 * W THEN VN = 6
127 VTAB VT: HTAB HT: PRINT "  ": VTAB VN: HTAB HT: PRINT
    "->":VT = VN: GOTO 1210
128 REM ** Shapetabelle erstellen **
129 TA = 24576:VT = 6:HT = 7:N = 0:SZ = 0:SB = 0:V = 1:W
    = 0: POKE - 16303,0
130 HOME : VTAB 6: HTAB 10: PRINT "Alte Tabelle
    editieren"
131 VTAB 8: HTAB 10: PRINT "Neue Tabelle erstellen"
132 VTAB 10: HTAB 10: PRINT "Shape.Creator aufrufen"
133 VTAB 12: HTAB 10: PRINT "Hauptprogramm und Ende"
134 GOSUB 1170:FM = 2
135 ON (VT - 4) / 2 GOTO 1360,1450,2620,1040
136 GOSUB 2730
137 FF = 1: ONERR GOTO 2760
138 HOME : VTAB 22: INPUT "Tabelle ('?'=CATALOG):
    ";N$:N1$ = N$
139 ON ( LEFT$ (N$,1) = "?") GOTO 2650: ON ( LEFT$ (N$,1)
    = Q$) GOTO 1290
140 PRINT : PRINT D$"BLOAD"N$",A$6000"
141 N = ( PEEK (TA + 2) + 256 * PEEK (TA + 3) - 2) / 2:SZ
    = PEEK (TA)
142 IF N = SZ THEN HOME : VTAB 22: PRINT "Die Tabelle ist
    voll !": FOR I = 1 TO 2500: NEXT : GOTO 1040
143 HOME : VTAB 22: PRINT N - SZ" Shape(s) haben noch
    Platz !"
144 FOR I = 1 TO 2500: NEXT : GOTO 1510
145 GOSUB 2730
146 FF = 2: ONERR GOTO 2760
147 HOME : VTAB 22: INPUT "Anzahl der Shapes: ";N$:N =
    VAL (N$): ON (N$ = Q$) GOTO 1290: ON (N < 1 OR N >
    255) GOTO 1470
148 TB = 2 * N + 2
149 FOR I = TA TO TA + TB: POKE I,0: NEXT
150 POKE TA + 2,TB - 256 * INT (TB / 256): POKE TA + 3,
    INT (TB / 256)
151 SB = PEEK (TA + SZ * 2 + 2) + 256 * PEEK (TA + SZ * 2
    + 3) + TA
152 FM = 5: HOME : VTAB 22: INPUT "Shape ('?'=CATALOG):
    ";N$: ON ( LEFT$ (N$,1) = "?") GOTO 2650: ON ( LEFT$
    (N$,1) = Q$) GOTO 1290
153 FF = 3: ONERR GOTO 2760
154 PRINT : PRINT D$"BLOAD"N$",A$4000"
155 POKE 232,0: POKE 233,64: XDRAW 1 AT 93,96
156 HOME : VTAB 22: PRINT "Dieses Shape ? (J/N): ";: GET
    E$: IF E$ < > "J" THEN HGR : GOTO 1520
157 I = 16387
158 I = I + 1:J = PEEK (I): POKE SB + I - 16388,J:: IF J
    = 0 THEN 1600
159 GOTO 1580
160 SZ = SZ + 1:SE = SB + I - 16387 - TA: POKE TA,SZ: IF
    SZ = N THEN HOME : VTAB 22: PRINT "Die Tabelle ist
    voll !": FOR I = 1 TO 2500: NEXT : GOTO 1640
161 POKE TA + 2 * SZ + 2,SE - 256 * INT (SE / 256): POKE
    TA + 2 * SZ + 3, INT (SE / 256)
162 HOME : VTAB 22: PRINT "Weiter ? (J/N): ";: GET E$: IF
    E$ < > "N" THEN HGR : GOTO 1510
163 FF = 4: ONERR GOTO 2760
164 HOME : VTAB 21: PRINT "Name: ( "N1$" ) = <RETURN>"
165 VTAB 22: INPUT "      ";N$: IF N$ = "" THEN N$ = N1$:
    GOTO 1690
166 PRINT "Wird als ST."N$" abgespeichert."
167 PRINT : PRINT D$"BSAVE ST."N$",A$6000,L"SE
168 GOTO 1700
169 PRINT : PRINT D$"BSAVE"N$",A$6000,L"SE
170 GOTO 1040
171 REM ** Shapetabelle editieren **
```

```
172Ø TA = 16384:VT = 6:HT = 7:FM = 3:HH = Ø:V = Ø:W = Ø:
      POKE - 163Ø3,Ø
173Ø HOME : VTAB 6: HTAB 1Ø: PRINT "Die Tabelle
      verlängern"
174Ø VTAB 8: HTAB 1Ø: PRINT "Shape- Folge verändern"
175Ø VTAB 1Ø: HTAB 1Ø: PRINT "Hauptprogramm und Ende"
176Ø GOSUB 117Ø
177Ø ON (VT - 4) / 2 GOTO 179Ø,178Ø,1Ø4Ø
178Ø HH = 1: HGR
179Ø FF = 5: ONERR GOTO 276Ø
18ØØ HOME : VTAB 22: INPUT "Tabelle ('?'=CATALOG):
      ";N$:N1$ = N$
181Ø ON ( LEFT$ (N$,1) = "?") GOTO 265Ø: ON ( LEFT$ (N$,1)
      = Q$) GOTO 172Ø
182Ø PRINT : PRINT D$"BLOAD"N$",A$4ØØØ"
183Ø SZ = PEEK (TA):I = TA + PEEK (TA + SZ * 2) + 256 *
      PEEK (TA + SZ * 2 + 1)
184Ø I = I + 1:J = PEEK (I): IF J < > Ø THEN 184Ø
185Ø SL = I - TA + 1
186Ø IF HH THEN 199Ø
187Ø FF = 6: ONERR GOTO 276Ø
188Ø HOME : VTAB 22: INPUT "Wieviel Shapes hinzu: ";N
189Ø IF SZ + N > 255 THEN HOME : VTAB 22: PRINT "Insgesamt
      nicht mehr als 255 Shapes !": FOR I = 1 TO 25ØØ: NEXT
      : GOTO 188Ø
19ØØ POKE TA - 2 * N,SZ: POKE TA - 2 * N + 1,Ø: POKE TA,Ø
191Ø FOR I = TA + 2 TO TA + 2 * SZ STEP 2
192Ø J = PEEK (I) + 256 * PEEK (I + 1) + 2 * N
193Ø POKE I - 2 * N,J - 256 * INT (J / 256): POKE I,Ø
194Ø POKE I + 1 - 2 * N, INT (J / 256): POKE I + 1,Ø
195Ø NEXT
196Ø TA = TA - 2 * N:I = TA + PEEK (TA + SZ * 2) + 256 *
      PEEK (TA + SZ * 2 + 1)
197Ø I = I + 1:J = PEEK (I): IF J < > Ø THEN 197Ø
198Ø I = I - TA + 1: POKE TA + SZ * 2 + 2,I - 256 * INT (I
      / 256): POKE TA + SZ * 2 + 3, INT (I / 256): GOTO
      213Ø
199Ø POKE 233,64: POKE 232,Ø: FOR I = 1 TO 2
2ØØØ XDRAW 1 AT 93 * I,96:N1 = 1: IF PEEK (TA) = 1 THEN
      VTAB 22: PRINT "Die Tabelle hat nur ein Shape!": FOR
      I = 1 TO 15ØØ: NEXT : GOTO 172Ø
2Ø1Ø HOME : VTAB 21: PRINT "<- Wahl ->, <RETURN> =
      Annahme":N = 1
2Ø2Ø VTAB 22: HTAB 1: PRINT "Shape-Nr.: "N1"  ";: GET E$
2Ø3Ø IF E$ < > CHR$ (8) AND E$ < > CHR$ (21) AND E$ < >
      CHR$ (13) THEN 2Ø2Ø
2Ø4Ø IF E$ = CHR$ (13) THEN N(I) = N: NEXT : GOTO 2Ø9Ø
2Ø5Ø N = N + (E$ = CHR$ (21)) - (E$ = CHR$ (8))
2Ø6Ø IF N < 1 THEN N = PEEK (TA)
2Ø7Ø IF N > PEEK (TA) THEN N = 1
2Ø8Ø XDRAW N1 AT 93 * I,96: XDRAW N AT 93 * I,96:N1 = N:
      GOTO 2Ø2Ø
2Ø9Ø W1 = PEEK (TA + 2 * N(1)):W2 = PEEK (TA + 2 * N(1) +
      1):W3 = PEEK (TA + 2 * N(2)):W4 = PEEK (TA + 2 * N(2)
      + 1)
21ØØ POKE TA + 2 * N(1),W3: POKE TA + 2 * N(1) + 1,W4
211Ø POKE TA + 2 * N(2),W1: POKE TA + 2 * N(2) + 1,W2
212Ø HOME : VTAB 22: PRINT "Nochmal ? (J/N): ";: GET E$:
      IF E$ < > "N" THEN HGR : GOTO 199Ø
213Ø FF = 7: ONERR GOTO 276Ø
214Ø HOME : VTAB 21: PRINT "Name: ( "N1$" ) = <RETURN>"
215Ø VTAB 22: INPUT "       ";N$: IF N$ = "" THEN N$ = N1$:
      GOTO 219Ø
216Ø PRINT "Wird als ST."N$" abgespeichert."
217Ø PRINT : PRINT D$"BSAVE ST."N$",A"TA",L"SL + 2 * N *
      (HH = Ø)
218Ø GOTO 22ØØ
219Ø PRINT : PRINT D$"BSAVE"N$",A"TA",L"SL + 2 * N * (HH =
      Ø)
22ØØ HH = Ø: GOTO 1Ø4Ø
221Ø REM ** Einzelshapes erstellen **
222Ø GOSUB 273Ø
223Ø TA = 24576:N = 1:FM = 4
224Ø HOME : VTAB 22: INPUT "Tabelle ('?'=CATALOG):
      ";N$:N1$ = N$
225Ø ON ( LEFT$ (N$,1) = "?") GOTO 265Ø: ON ( LEFT$ (N$,1)
      = Q$) GOTO 1Ø4Ø
226Ø FF = 8: ONERR GOTO 276Ø
227Ø PRINT : PRINT D$"BLOAD"N$",A$6ØØØ"
228Ø POKE 232,Ø: POKE 233,96: POKE 16384,1: POKE 16385,Ø:
      POKE 16386,4: POKE 16387,Ø
229Ø XDRAW 1 AT 93,96:N1 = 1: IF PEEK (TA) < > 1 THEN 232Ø
23ØØ HOME : VTAB 22: PRINT "Dieses Shape ? (J/N): ";: GET
      E$: IF E$ = "J" THEN 24ØØ
231Ø GOTO 224Ø
232Ø HOME : VTAB 21: PRINT "<- Wahl ->, <RETURN> =
      Annahme":N = 1
233Ø VTAB 22: HTAB 1: PRINT "Shape-Nr.: "N1"  ";: GET E$
234Ø IF E$ < > CHR$ (8) AND E$ < > CHR$ (21) AND E$ < >
      CHR$ (13) THEN 233Ø
235Ø IF E$ = CHR$ (13) THEN 24ØØ
236Ø N = N + (E$ = CHR$ (21)) - (E$ = CHR$ (8))
237Ø IF N < 1 THEN N = PEEK (TA)
238Ø IF N > PEEK (TA) THEN N = 1
239Ø XDRAW N1 AT 93,96: XDRAW N AT 93,96:N1 = N: GOTO 233Ø
24ØØ S1 = TA + N * 2:S2 = S1 + 1:S3 = S2 + 1:S4 = S3 + 1
241Ø SB = TA + PEEK (S1) + 256 * PEEK (S2):I = SB - 1
242Ø IF N < > PEEK (TA) THEN 246Ø
243Ø I = I + 1
244Ø J = PEEK (I): IF J = Ø THEN SE = I + 1: GOTO 247Ø
245Ø GOTO 243Ø
246Ø SE = TA + PEEK (S3) + 256 *PEEK (S4)
247Ø FOR I = SB TO SE - 1: POKE 16388 + I - SB, PEEK (I):
      NEXT
248Ø POKE 233,64
249Ø XDRAW 1 AT 186,96
25ØØ HOME : VTAB 22: PRINT "Shape auf Diskette schreiben ?
      (J/N): ";: GET E$: IF E$ < > "J" THEN 254Ø
251Ø HOME : VTAB 22: INPUT "Name: ";N$: PRINT "Wird als
      SH."N$" abgespeichert."
252Ø FF = 9: ONERR GOTO 276Ø
253Ø PRINT : PRINT D$"BSAVE SH."N$",A$4ØØØ,L"SE - SB + 4
254Ø HOME : VTAB 22: PRINT "Nochmal ? (J/N): ";: GET E$:
      IF E$ < > "J" THEN 1Ø4Ø
255Ø HOME : VTAB 22: HGR : PRINT "Neue Tabelle ? (J/N):
      ";: GET E$: IF E$ < > "J" AND E$ < > "N" THEN 255Ø
256Ø IF E$ = "J" THEN HGR : GOTO 222Ø
257Ø I = Ø: HGR : GOTO 228Ø
258Ø REM ** Einzelshapes editieren **
259Ø FF = 1Ø: ONERR GOTO 276Ø
26ØØ PRINT : PRINT D$"RUN SHAPE.EDITOR"
261Ø REM ** Shape.Creator aufrufen **
262Ø FF = 1Ø: ONERR GOTO 276Ø
263Ø PRINT : PRINT D$"RUN SHAPE.CREATE"
264Ø REM ** Catalog Drive 1 **
265Ø HOME : POKE - 163Ø3,Ø
266Ø PRINT : PRINT D$"CATALOG"
267Ø GET E$: PRINT E$: HGR
268Ø ON FM GOTO 1Ø4Ø,137Ø,177Ø,224Ø,152Ø
269Ø REM ** Programmende **
27ØØ POKE 216,Ø: TEXT : HOME
271Ø PRINT : END : REM Oder: PRINT D$"RUN HELLO"
272Ø REM ** Bildschirm loeschen **
273Ø HOME : VTAB 22: HGR2 : HGR : HCOLOR= 3: ROT= Ø:
      SCALE= 1:I = Ø:J = Ø
274Ø RETURN
275Ø REM ** Fehlerbehandlung **
276Ø HOME : VTAB 22: IF PEEK (222) = 6 THEN PRINT "Diese
      Tabelle ist nicht auf der Disk !": GOTO 281Ø
277Ø IF PEEK (222) = 9 THEN PRINT "Die Diskette ist voll
      !": GOTO 281Ø
278Ø IF PEEK (222) = 8 THEN PRINT "Diskette defekt oder
      Laufwerk offen !": GOTO 281Ø
279Ø IF PEEK (222) = 4 OR PEEK (222) = 1Ø THEN PRINT
      "Disketten,- File-Schreibschutz !": GOTO 281Ø
28ØØ PRINT "Eingabefehler !"
281Ø FOR I = 1 TO 2ØØØ: NEXT : POKE 216,Ø: ON FF GOTO
      138Ø,146Ø,152Ø,164Ø,18ØØ,188Ø,214Ø,224Ø,251Ø,1Ø4Ø
```

SHAPE.EDITOR

```
1ØØØ REM ** Shape.Editor **
1Ø1Ø REM ** Wolfgang Landgraf 2Ø.Ø4.86 **
1Ø2Ø D$ = CHR$ (4):Q$ = "!"
1Ø3Ø IF PEEK (1Ø4) < > 64 THEN POKE 1Ø3,1: POKE 1Ø4,64:
      POKE 16384,Ø: PRINT : PRINT CHR$ (4)"RUN SHAPE.EDITOR"
1Ø4Ø IF PEEK (768) < > 1Ø THEN PRINT : PRINT CHR$
      (4)"BLOAD ST.EDIT,A$3ØØ"
1Ø5Ø REM ** Auswahl des Shapes **
1Ø6Ø TA = 24576: DIM B%(21,21)
1Ø7Ø HOME : VTAB 22: HGR : HCOLOR= 3: ROT= Ø: SCALE= 1:N =
      1
1Ø8Ø INPUT "Tabelle ('?' = Catalog): ";N$: ON ( LEFT$
      (N$,1) = "?") GOTO 32ØØ: ON ( LEFT$ (N$,1) = Q$) GOTO
      236Ø
1Ø9Ø FF = 1: ONERR GOTO 314Ø
11ØØ PRINT : PRINT CHR$ (4)"BLOAD"N$",A$6ØØØ"
111Ø POKE 233,96: POKE 232,Ø:HS = Ø
112Ø XDRAW 1 AT 14Ø,96:N1 = 1: IF PEEK (TA) < > 1 THEN
      115Ø
113Ø HOME : VTAB 22: PRINT "Dieses Shape ? (J/N): ";: GET
      E$: IF E$ = "J" THEN 124Ø
114Ø GOTO 1Ø7Ø
115Ø HOME : VTAB 21: PRINT "<- Wahl ->,<RET>=
      Annahme,<ESC>=Abbruch":N = 1
116Ø VTAB 22: HTAB 1: PRINT "Shape-Nr.: "N1"  ";: GET E$
117Ø IF E$ < > CHR$ (8) AND E$ < > CHR$ (21) AND E$ < >
      CHR$ (27) AND E$ < > CHR$ (13) THEN 116Ø
```

```
118Ø IF E$ = CHR$ (13) THEN N1 = N: GOTO 124Ø
119Ø IF E$ = CHR$ (27) THEN 1Ø7Ø
12ØØ N = N + (E$ = CHR$ (21)) - (E$ = CHR$ (8))
121Ø IF N < 1 THEN N = PEEK (TA)
122Ø IF N > PEEK (TA) THEN N = 1
123Ø XDRAW N1 AT 14Ø,96: XDRAW N AT 14Ø,96:N1 = N: GOTO
     116Ø
124Ø GOSUB 297Ø: RESTORE
125Ø GOSUB 238Ø
126Ø POKE 233,3
127Ø REM ** Einlesen der Daten aus gezeichnetem Shape **
128Ø Y = 3:Y1 = Ø
129Ø FOR J = 8863 TO 16Ø31 STEP 1Ø24:Y = Y + 7:Y1 = Y1 +
     1:M = 7: FOR I = Ø + 2 * HS TO 2 - 2 * HS STEP ( - HS
     * 2) + 1
13ØØ A = PEEK (J + I)
131Ø GOSUB 27ØØ
132Ø NEXT : NEXT
133Ø FOR J = 8991 TO 16159 STEP 1Ø24:Y = Y + 7:Y1 = Y1 +
     1:M = 7: FOR I = Ø + 2 * HS TO 2 - 2 * HS STEP ( - HS
     * 2) + 1
134Ø A = PEEK (J + I)
135Ø GOSUB 27ØØ
136Ø NEXT : NEXT
137Ø FOR J = 9119 TO 13215 STEP 1Ø24:Y = Y + 7:Y1 = Y1 +
     1:M = 7: FOR I = Ø + 2 * HS TO 2 - 2 * HS STEP ( - HS
     * 2) + 1
138Ø A = PEEK (J + I)
139Ø GOSUB 27ØØ
14ØØ NEXT : NEXT
141Ø GOSUB 238Ø
142Ø REM ** Grenzen des Shapes berechnen **
143Ø HOME : VTAB 22: PRINT "Shape wird berechnet, bitte
     warten ...":X1 = Ø:Y1 = Ø:B = Ø
144Ø Y1 = Y1 + 1
145Ø FOR X1 = 1 TO 21:B = B + B%(X1,Y1): NEXT
146Ø IF B = Ø THEN 144Ø
147Ø HO = Y1:Y1 = - 1:B = Ø
148Ø Y1 = Y1 + 1:Y2 = 21 - Y1
149Ø FOR X1 = 1 TO 21:B = B + B%(X1,Y2): NEXT
15ØØ IF B = Ø THEN 148Ø
151Ø HU = Y2:X1 = Ø:B = Ø
152Ø X1 = X1 + 1
153Ø FOR Y1 = 1 TO 21:B = B + B%(X1,Y1): NEXT
154Ø IF B = Ø THEN 152Ø
155Ø HL = X1:X1 = - 1:B = Ø
156Ø X1 = X1 + 1:X2 = 21 - X1
157Ø FOR Y1 = 1 TO 21:B = B + B%(X2,Y1): NEXT
158Ø IF B = Ø THEN 156Ø
159Ø HR = X2:U = 2Ø51:C = Ø:I = Ø:J = Ø
16ØØ REM ** Vektoren bis zur oberen linken Ecke poken **
161Ø IF SY > HO THEN 169Ø
162Ø IF SY = HO THEN 173Ø
163Ø C = HO - SY
164Ø I = I + 1: IF C = Ø THEN I = I - 1
165Ø IF C > = 3 THEN POKE U + I,146:C = C - 3: GOTO 164Ø
166Ø IF C > = 2 THEN POKE U + I,144:C = C - 2: GOTO 164Ø
167Ø IF C > = 1 THEN POKE U + I,13Ø
168Ø GOTO 173Ø
169Ø C = SY - HO
17ØØ I = I + 1: IF C = Ø THEN I = I - 1
171Ø IF C > = 2 THEN POKE U + I,216: POKE U + I + 1,72:I =
     I + 1:C = C - 2: GOTO 17ØØ
172Ø IF C > = 1 THEN POKE U + I,216: POKE U + I + 1,73:
     POKE U + I + 2,2Ø3:I = I + 2
173Ø IF HL > SX THEN 181Ø
174Ø IF HL = SX THEN 187Ø
175Ø C = SX - HL
176Ø I = I + 1
177Ø IF C > = 3 THEN POKE U + I,219:C = C - 3: GOTO 176Ø
178Ø IF C > = 2 THEN POKE U + I,219: POKE U + I + 1,2Ø1:I
     = I + 1:C = C - 2: GOTO 176Ø
179Ø IF C > = 1 THEN POKE U + I,91
18ØØ GOTO 187Ø
181Ø C = HL - SX
182Ø I = I + 1
183Ø IF C > = 3 THEN POKE U + I,73:C = C - 3: GOTO 182Ø
184Ø IF C > = 2 THEN POKE U + I,73: POKE U + I + 1,91:I =
     I + 1:C = C - 2: GOTO 182Ø
185Ø IF C > = 1 THEN POKE U + I,75
186Ø REM ** Startbytes des Shapes poken, dann Anfang des
     Bildes bestimmen **
187Ø POKE 2Ø48,1: POKE 2Ø49,Ø: POKE 2Ø5Ø,4: POKE 2Ø51,Ø
188Ø J = J + 1
189Ø I = PEEK (2Ø51 + J): IF I < > Ø THEN 188Ø
19ØØ U = 2Ø51 + J:J = Ø:C = Ø
191Ø REM ** Vektoren des Bildes poken **
192Ø FOR I = HO TO HU:A = (A = Ø)
193Ø IF I = 22 THEN 223Ø
194Ø IF A = Ø THEN 2Ø5Ø
195Ø FOR J = HL TO HR
196Ø B = B%(J,I):C = C + 1
197Ø IF J = HR THEN 215Ø
198Ø IF NOT B THEN 2Ø2Ø
199Ø IF C = 3 THEN U = U + 1:C = 1: GOTO 2Ø1Ø
2ØØØ IF C = 2 THEN POKE U, PEEK (U) + 4Ø: NEXT
2Ø1Ø IF C = 1 THEN POKE U, PEEK (U) + 5: NEXT
2Ø2Ø IF C = 3 THEN POKE U, PEEK (U) + 64:U = U + 1:C = Ø:
     NEXT
2Ø3Ø IF C = 2 THEN POKE U, PEEK (U) + 8: NEXT
2Ø4Ø IF C = 1 THEN POKE U, PEEK (U) + 1: NEXT
2Ø5Ø FOR J = HR TO HL STEP - 1
2Ø6Ø B = B%(J,I):C = C + 1
2Ø7Ø IF J = HL THEN 215Ø
2Ø8Ø IF NOT B THEN 212Ø
2Ø9Ø IF C = 3 THEN U = U + 1:C = 1: GOTO 211Ø
21ØØ IF C = 2 THEN POKE U, PEEK (U) + 56: NEXT
2111 IF C = 1 THEN POKE U, PEEK (U) + 7: NEXT
2112 IF C = 3 THEN POKE U, PEEK (U) + 192:U = U + 1:C = Ø:
     NEXT
2113 IF C = 2 THEN POKE U, PEEK (U) + 24: NEXT
2114 IF C = 1 THEN POKE U, PEEK (U) + 3: NEXT
2115 IF NOT B THEN 219Ø
2116 IF C = 3 THEN U = U + 1:C = 1: GOTO 218Ø
2117 IF C = 2 THEN POKE U, PEEK (U) + 48: GOTO 222Ø
2118 IF C = 1 THEN POKE U, PEEK (U) + 6: GOTO 222Ø
2119 IF C = 3 THEN POKE U, PEEK (U) + 128:U = U + 1:C = Ø
22ØØ IF C = 2 THEN POKE U, PEEK (U) + 16
2211 IF C = 1 THEN POKE U, PEEK (U) + 2
2222 NEXT I
2223 IF C > Ø THEN U = U + 1
2224 POKE U,Ø
2225 REM ** Shape auf Diskette, Neustart ? **
2226 HCOLOR= Ø: FOR I = 216 TO 238: HPLOT I,89 TO I,111:
     NEXT : HCOLOR= 3
2227 POKE 233,8: XDRAW 1 AT 227,1ØØ: POKE 233,3: FOR I = 1
     TO 16: XDRAW 8 AT 227,1ØØ: FOR J = 1 TO 25Ø: NEXT :
     NEXT
2228 HOME : VTAB 22: PRINT "Shape auf Diskette schreiben ?
     (J/N): ";: GET E$: IF E$ < > "J" THEN 232Ø
2229 HOME : VTAB 22: INPUT "Name: SH.";N$: PRINT "Wird als
     SH."N$" abgespeichert."
23ØØ FF = 2: ONERR GOTO 314Ø
2311 PRINT : PRINT CHR$ (4)"BSAVE SH."N$",A2Ø48,L"U - 2Ø47
2321 HOME : VTAB 22: PRINT "Nochmal ? (J/N): ";: GET E$:
     IF E$ < > "J" THEN HOME : TEXT : GOTO 236Ø
2331 HOME : VTAB 22: HGR : PRINT "Neue Tabelle ? (J/N):
     ";: GET E$: IF E$ < > "J" AND E$ < > "N" THEN 233Ø
2341 IF E$ = "J" THEN 1Ø7Ø
2351 N = 1:HS = 1: GOTO 111Ø
2361 POKE 216,Ø: PRINT : PRINT CHR$ (4)"RUN TABLE.EDITOR"
2371 REM ** Bewegungen Cursor und Shape **
2381 FH = (FH = Ø): IF NOT FH THEN VTAB 21: HTAB 4: PRINT
     "tartpunkt setzen"
2391 IF FH THEN XA = 227:YA = 5Ø: POKE 233,96: GOTO 241Ø
24ØØ XA = 87:YA = 8Ø:N = 7:X1 = 11:Y1 = 11:SX = X1:SY =
     Y1: POKE 233,3: XDRAW N AT XA,YA: XDRAW 4 AT XA,YA:
     CX= XA:CY = YA
2411 VTAB 22: HTAB 36: GET E$
2421 IF E$ < > "J" AND E$ < > "K" AND E$ < > "I" AND E$ <
     > "M" AND E$ < > "S" AND E$ < > "P" AND E$ < > "L"
     AND E$ < > CHR$ (27) AND E$ < > CHR$ (13) THEN 241Ø
2431 XN = XA + ((E$ = "K") + 6 * (FH = Ø AND E$ = "K")) -
     ((E$ = "J") + 6 * (FH = Ø AND E$ = "J"))
2441 YN = YA + ((E$ = "M") + 6 * (FH = Ø AND E$ = "M")) -
     ((E$ = "I") + 6 * (FH = Ø AND E$ = "I"))
2451 IF NOT FH THEN X1 = X1 + (E$ = "K") - (E$ = "J")
2461 IF NOT FH THEN Y1 = Y1 + (E$ = "M") - (E$ = "I")
2471 IF FH AND XN > 26Ø THEN XN = 26Ø: GOTO 241Ø
2481 IF FH AND XN < 19Ø THEN XN = 19Ø: GOTO 241Ø
2491 IF FH AND YN > 1ØØ THEN YN = 1ØØ: GOTO 241Ø
2501 IF FH AND YN < 2Ø THEN YN = 2Ø: GOTO 2410
2511 IF FH AND E$ = "S" THEN HS = (HS = Ø): IF HS THEN
     VTAB 21: HTAB 4: PRINT "= Normal": GOTO 241Ø
2521 IF FH AND NOT HS AND E$ = "S" THEN VTAB 21: HTAB 4:
     PRINT "piegeln ": GOTO 241Ø
2531 IF FH THEN 259Ø
2541 IF E$ = CHR$ (27) THEN N = N1:HS = Ø: POKE 233,96:
     GOTO 124Ø
2551 IF XN > 157 THEN XN = 157:X1 = 21: GOTO 241Ø
2561 IF XN < 17 THEN XN = 17:X1 = 1: GOTO 241Ø
2571 IF YN > 15Ø THEN YN = 15Ø:Y1 = 21: GOTO 241Ø
2581 IF YN < 1Ø THEN YN = 1Ø:Y1 = 1: GOTO 241Ø
2591 IF FH AND E$ = CHR$ (27) THEN FH = Ø: HGR : GOTO 111Ø
26ØØ IF E$ = CHR$ (13) THEN 268Ø
2611 IF NOT FH AND E$ = "P" OR E$ = "L" OR E$ = "S" THEN
     263Ø
2621 XDRAW N AT XA,YA: XDRAW N AT XN,YN:XA = XN:YA = YN:
     GOTO 241Ø
2631 XDRAW N AT XA,YA
```

FACHBUCHTIP Hüthig

Das Buch zum Apple-Writer II/IIe

Hans Gabriel

1986, 155 S., kart., DM 35,—
ISBN 3-7785-1234-X

„Das Buch zum Apple Writer II/IIe" wendet sich an alle, die dieses Textverarbeitungssystem schon einsetzen oder einsetzen wollen.
In einzelnen, in sich geschlossenen Kapiteln erlernt der Leser alle Funktionen und erzielt schnelle Erfolgserlebnisse.
Im ersten Kapitel werden typische Arbeitsstellungen der Textverarbeitung und ihre systematische Bearbeitung vorgestellt. Das zweite Kapitel stellt in logischen Funktionsgruppen die Befehle vor, mit denen der Applewriter während der Textarbeit direkt gesteuert werden kann. Kapitel 3 zeigt die Programmierung des Appiewriters in der Text-Kommando-Sprache TKS (WPL). Dazu werden Beispiele analysiert. Auf der Begleitdiskette zum Buch, die separat bezogen werden kann, sind darüber hinaus umfangreiche Schwerpunkterklärungen enthalten, die über die Help-Funktion vom Benutzer abgerufen werden können. Eine kleine Adreßdatenbank mit den dazugehörigen Dienstprogrammen zur Pflege der Daten und zu ihrem Einsatz in Einzel- und Serienbriefen befindet sich ebenfalls auf Diskette.
„Das Buch zum Apple Writer II/IIe" behandelt sowohl die alte Programmversion für den Apple IIplus als auch die neue Ausgabe, die 128-kByte auf dem Apple IIe oder Apple IIc unterstützt.

BESTELLCOUPON

Bitte ausfüllen und an Hüthig Vertriebsservice, Postfach 102869 · 6900 Heidelberg schicken.

Buchtitel

Name

Straße

Ort

Unterschrift

```
264Ø IF E$ = "P" AND B%(X1,Y1) = Ø THEN XDRAW N AT XA,YA:
     XDRAW 6 AT XA,YA:XN = XA:YN = YA:B%(X1,Y1) = 1:
     HCOLOR= 3: HPLOT XN / 7 + 215,YN / 7 + 89: GOTO 241Ø
265Ø IF E$ = "L" AND B%(X1,Y1) = 1 THEN XDRAW N AT XA,YA:
     XDRAW 6 AT XA,YA:XN = XA:YN = YA:B%(X1,Y1) = Ø:
     HCOLOR= Ø: HPLOT XN / 7 + 215,YN / 7 + 89: HCOLOR= 3:
     GOTO 241Ø
266Ø IF E$ = "S" THEN XDRAW 4 AT XA,YA:SX = X1:SY = Y1:
     XDRAW 4 AT CX,CY: XDRAW N AT XA,YA:XN = XA:YN = YA:CX
     = XA:CY = YA: GOTO 241Ø
267Ø XDRAW N AT XA,YA: GOTO 241Ø
268Ø RETURN
269Ø REM ** Gelesene Bytes in Feld B(X1,Y1) eintragen **
27ØØ IF A = Ø THEN FOR K = 1 TO 7:A(K) = Ø: NEXT : GOTO
     285Ø
271Ø IF A > = 64 THEN A(7 - 6 * HS) = 1:A = A - 64: GOTO
     273Ø
272Ø A(7 - 6 * HS) = Ø
273Ø IF A > = 32 THEN A(6 - 4 * HS) = 1:A = A - 32: GOTO
     275Ø
274Ø A(6 - 4 * HS) = Ø
275Ø IF A > = 16 THEN A(5 - 2 * HS) = 1:A = A - 16: GOTO
     277Ø
276Ø A(5 - 2 * HS) = Ø
277Ø IF A > = 8 THEN A(4) = 1:A = A - 8: GOTO 279Ø
278Ø A(4) = Ø
279Ø IF A > = 4 THEN A(3 + 2 * HS) = 1:A = A - 4: GOTO
     281Ø
28ØØ A(3 + 2 * HS) = Ø
281Ø IF A > = 2 THEN A(2 + 4 * HS) = 1:A = A - 2: GOTO
     283Ø
282Ø A(2 + 4 * HS) = Ø
283Ø IF A > = 1 THEN A(1 + 6 * HS) = 1:A = A - 1: GOTO
     285Ø
284Ø A(1 + 6 * HS) = Ø
285Ø H = I
286Ø IF I = Ø AND HS THEN I = 2: GOTO 288Ø
287Ø IF I = 2 AND HS THEN I = Ø
288Ø FOR K = 1 TO 7
289Ø X = M * K + I * 49 + 1Ø:X1 = K + I * 7
29ØØ IF A(K) = 1 THEN XDRAW 6 AT X,Y: HCOLOR= 3: HPLOT X /
     7 + 215,Y / 7 + 89: GOTO 292Ø
291Ø HCOLOR= Ø: HPLOT X / 7 + 215,Y / 7 + 89: HCOLOR= 3
292Ø B%(X1,Y1) = A(K)
293Ø NEXT
294Ø I = H
295Ø RETURN
296Ø REM ** Bildaufbau **
297Ø HOME : HGR : FOR I = 2Ø48 TO 235Ø: POKE I,Ø: NEXT :
     POKE 233,3
298Ø FOR I = 13 TO 16Ø STEP 7: HPLOT I,6 TO I,153: NEXT
299Ø FOR I = 6 TO 153 STEP 7: HPLOT 13,I TO 16Ø,I: NEXT
3ØØØ HPLOT 216,39 TO 238,39 TO 238,61 TO 216,61 TO 216,39
3Ø1Ø DRAW 9 AT 22Ø,37
3Ø2Ø HPLOT 216,89 TO 238,89 TO 238,111 TO 216,111 TO
     216,89
3Ø3Ø DRAW 1Ø AT 22Ø,87
3Ø4Ø FOR I = 1 TO 32: READ A,B,C: DRAW A AT B,C: NEXT
3Ø5Ø DATA 1,8,1Ø,5,8,38,1,5,73,3,1Ø,73,1,5,1Ø8,5,1Ø,1Ø8,2,5,
     143,3,1Ø,143
3Ø6Ø DATA 1,17,2,5,45,2,1,77,2,3,82,2,1,112,2,5,117,2,2,147,
     2,3,152,2
3Ø7Ø DATA 1,166,1Ø,5,166,38,1,163,73,3,168,73,1,163,1Ø8,5,
     168,1Ø8,2,163,143,3,168,143
3Ø8Ø DATA 1,17,157,5,45,157,1,77,157,3,82,157,1,112,157,5,
     117,157,2,147,157,3,152,157
3Ø9Ø POKE 233,96: XDRAW N AT 227,5Ø: POKE 233,3
31ØØ VTAB 21: HTAB 1: PRINT "<S>piegeln": VTAB 21: HTAB
     36: PRINT "I"
311Ø VTAB 22: HTAB 1: PRINT "<P>unkt, <L>oeschen
     Cursor ->": VTAB 22: HTAB 35: PRINT "J": VTAB 22:
     HTAB 37: PRINT "K"
312Ø VTAB 23: HTAB 1: PRINT "<RET> Editier-Ende": VTAB 23:
     HTAB 36: PRINT "M": RETURN
313Ø REM ** Fehlerbehandlung **
314Ø HOME : VTAB 22: IF PEEK (222) = 6 THEN PRINT "Diese
     Tabelle ist nicht auf der Disk !": GOTO 319Ø
315Ø IF PEEK (222) = 9 THEN PRINT "Die Diskette ist voll
     !": GOTO 319Ø
316Ø IF PEEK (222) = 8 THEN PRINT "Diskette defekt oder
     Laufwerk offen !": GOTO 319Ø
317Ø IF PEEK (222) = 4 OR PEEK (222) = 1Ø THEN PRINT
     "Disketten,- File-Schreibschutz !": GOTO 319Ø
318Ø PRINT "Eingabefehler !"
319Ø FOR I = 1 TO 2ØØØ: NEXT : POKE 216,Ø: ON FF GOTO
     1Ø7Ø,229Ø
32ØØ TEXT : HOME : PRINT : PRINT CHR$ (4)"CATALOG": GET
     E$: PRINT : GOTO 1Ø7Ø
```

SHAPE.CREATE

```
1ØØØ REM ** Shape.Create **
1Ø1Ø REM ** Wolfgang Landgraf, Juni 86 **
1Ø2Ø REM ** Start des Programmes bei $4ØØØ **
1Ø3Ø D$ = CHR$ (4):Q$ = "!"
1Ø4Ø IF PEEK (1Ø4) < > 64 THEN POKE 1Ø3,1: POKE 1Ø4,64:
     POKE 16384,Ø: PRINT D$"RUN SHAPE.CREATE"
1Ø5Ø TEXT:HOME
1Ø6Ø IF PEEK (768) < > 1Ø THEN PRINT D$"BLOAD ST.EDIT"
1Ø7Ø FOR I = Ø TO 39: PRINT "-";: NEXT
1Ø8Ø PRINT "              SHAPE.CREATOR"
1Ø9Ø FOR I = Ø TO 39: PRINT "-";: NEXT :
11ØØ POKE 34,4
111Ø REM ** Wahl der Tabelle **
112Ø POKE 232,Ø: POKE 233,3:TA = 24576
113Ø VTAB 7: PRINT "<N>eue Tabelle": PRINT "<L>ade
     Tabelle"
114Ø VTAB 1Ø: HTAB 1: PRINT "Wahl ... ";: GET E$
115Ø IF E$ < > "L" AND E$ < > "N" THEN 114Ø
116Ø IF E$ = "N" THEN 127Ø
117Ø VTAB 12: HTAB 1
118Ø INPUT "Tabelle ('?' = Catalog): ";N$: ON ( LEFT$
     (N$,1) = "?") GOTO 3ØØØ: ON ( LEFT$ (N$,1) = Q$) GOTO
     113Ø
119Ø FF = 1: ONERR GOTO 293Ø
12ØØ PRINT CHR$ (4)"BLOAD"N$",A$6ØØØ":N1$ = N$
121Ø N = ( PEEK (TA + 2) + 256 * PEEK (TA + 3) - 2) / 2
122Ø SN = PEEK (TA): PRINT : IF SN = N THEN PRINT "Tabelle
     voll": FOR I = 1 TO 1ØØØ: NEXT : GOTO 1Ø5Ø
123Ø PRINT "Die Tabelle kann ";N;" Shapes haben !": PRINT
124Ø PRINT "Jetzt hat sie ";SN;" Shape(s) !": PRINT
125Ø FOR I = 1 TO 2ØØØ: NEXT
126Ø GOTO 133Ø
127Ø VTAB 12: HTAB 1: INPUT "Anzahl der Shapes in der
     Tabelle :";E$:N = VAL (E$): IF N < 1 OR N > 255 THEN
     127Ø
128Ø TN = 2 * N + 2
129Ø FOR I = TA TO TA + TN: POKE I,Ø: NEXT I
13ØØ POKE TA + 2,TN - 256 * INT (TN / 256)
131Ø POKE TA + 3, INT (TN / 256)
132Ø REM ** Startbedingungen bestimmen **
133Ø XM = 16Ø:YM = 153
134Ø DIM B%(21,21)
135Ø GOSUB 278Ø
136Ø FF = 3: ONERR GOTO 293Ø
137Ø U = TA + PEEK (TA + SN * 2 + 2) + 256 * PEEK (TA + SN
     * 2 + 3)
138Ø FOR I = U TO U + 3Ø2: POKE I,Ø: NEXT
139Ø XA = 87:YA = 8Ø:X1 = 11:Y1 = 11:SX = X1:SY = Y1:
     XDRAW 7 AT XA,YA: XDRAW 4 AT XA,YA:CX = XA:CY = YA
14ØØ REM ** Tastatureingabe **
141Ø VTAB 22: HTAB 36: GET E$
142Ø IF E$ < > "J" AND E$ < > "K" AND E$ < > "I" AND E$ <
     > "M" AND E$ < > "S" AND E$ < > "P" AND E$ < > "L"
     AND E$ < > CHR$ (27) AND E$ < > CHR$ (13) THEN 141Ø
143Ø XN = XA + (E$ = "K") * 7 - (E$ = "J") * 7
144Ø YN = YA + (E$ = "M") * 7 - (E$ = "I") * 7
145Ø X1 = X1 + (E$ = "K") - (E$ = "J")
146Ø Y1 = Y1 + (E$ = "M") - (E$ = "I")
147Ø IF XN > XM - 3 THEN XN = XM - 3:X1 = 21: GOTO 141Ø
148Ø IF XN < 17 THEN XN = 17:X1 = 1: GOTO 141Ø
149Ø IF YN > YM - 3 THEN YN = YM - 3:Y1 = 21: GOTO 141Ø
15ØØ IF YN < 1Ø THEN YN = 1Ø:Y1 = 1: GOTO 141Ø
151Ø IF E$ = CHR$ (13) THEN XDRAW 7 AT XA,YA: XDRAW 4 AT
     CX,CY: GOTO 16ØØ
152Ø IF E$ = "P" OR E$ = "L" OR E$ = "S" THEN 154Ø
153Ø XDRAW 7 AT XA,YA: XDRAW 7 AT XN,YN:XA = XN:YA = YN:
     GOTO 141Ø
154Ø XDRAW 7 AT XA,YA
155Ø IF E$ = "P" AND B%(X1,Y1) = Ø THEN XDRAW 7 AT XA,YA:
     XDRAW 6 AT XA,YA:XN = XA:YN = YA:B%(X1,Y1) = 1: FOR I
     = Ø TO 3 STEP 3: HCOLOR= I: HPLOT XM + 15 + X1,7 + Y1
     + 34 * (I = 3): NEXT : GOTO 141Ø
156Ø IF E$ = "L" AND B%(X1,Y1) = 1 THEN XDRAW 7 AT XA,YA:
     XDRAW 6 AT XA,YA:XN = XA:YN = YA:B%(X1,Y1) = Ø: FOR I
     = 3 TO Ø STEP - 3: HCOLOR= I: HPLOT XM + 15 + X1,7 +
     Y1 + 34 * (I = Ø): NEXT : GOTO 141Ø
157Ø IF E$ = "S" THEN XDRAW 4 AT XA,YA:SX = X1:SY = Y1:
     XDRAW 4 AT CX,CY: XDRAW 7 AT XA,YA:XN = XA:YN = YA: CX
     = XA:CY = YA: GOTO 141Ø
158Ø XDRAW 7 AT XA,YA: GOTO 141Ø
159Ø REM ** Eckpunkte festlegen **
16ØØ HOME : VTAB 22: PRINT "Shape wird berechnet, bitte
     warten ...":X1 = Ø:Y1 = Ø:B = Ø
161Ø Y1 = Y1 + 1
162Ø FOR X1 = 1 TO 21:B = B + B%(X1,Y1): NEXT
```

```
163Ø IF B = Ø THEN 161Ø
164Ø HO = Y1:Y1 = - 1:B = Ø
165Ø Y1 = Y1 + 1:Y2 = 21 - Y1
166Ø FOR X1 = 1 TO 21:B = B + B%(X1,Y2): NEXT
167Ø IF B = Ø THEN 165Ø
168Ø HU = Y2:X1 = Ø:B = Ø
169Ø X1 = X1 + 1
17ØØ FOR Y1 = 1 TO 21:B = B + B%(X1,Y1): NEXT
171Ø IF B = Ø THEN 169Ø
172Ø HL = X1:X1 = - 1:B = Ø
173Ø X1 = X1 + 1:X2 = 21 - X1
174Ø FOR Y1 = 1 TO 21:B = B + B%(X2,Y1): NEXT
175Ø IF B = Ø THEN 173Ø
176Ø HR = X2:C = Ø:I = Ø:J = Ø:U = U - 1
177Ø REM ** Vektoren bis zur oberen linken Ecke poken **
178Ø IF SY > HO THEN 186Ø
179Ø IF SY = HO THEN 19ØØ
18ØØ C = HO - SY
181Ø I = I + 1: IF C = Ø THEN I = I - 1
182Ø IF C > = 3 THEN POKE U + I,146:C = C - 3: GOTO 181Ø
183Ø IF C > = 2 THEN POKE U + I,144:C = C - 2: GOTO 181Ø
184Ø IF C > = 1 THEN POKE U + I,13Ø
185Ø GOTO 19ØØ
186Ø C = SY - HO
187Ø I = I + 1: IF C = Ø THEN I = I - 1
188Ø IF C > = 2 THEN POKE U + I,216: POKE U + I + 1,72:I =
    I + 1:C = C - 2: GOTO 187Ø
189Ø IF C > = 1 THEN POKE U + I,216: POKE U + I + 1,73:
    POKE U + I + 2,2Ø3:I = I + 2
19ØØ IF HL > SX THEN 198Ø
191Ø IF HL = SX THEN 2Ø3Ø
192Ø C = SX - HL
193Ø I = I + 1
194Ø IF C > = 3 THEN POKE U + I,219:C = C - 3: GOTO 193Ø
195Ø IF C > = 2 THEN POKE U + I,219: POKE U + I + 1,2Ø1:I
    = I + 1:C = C - 2: GOTO 193Ø
196Ø IF C > = 1 THEN POKE U + I,91
197Ø GOTO 2Ø3Ø
198Ø C = HL - SX
199Ø I = I + 1
2ØØØ IF C > = 3 THEN POKE U + I,73:C = C - 3: GOTO 199Ø
2Ø1Ø IF C > = 2 THEN POKE U + I,73: POKE U + I + 1,91:I =
    I + 1:C = C - 2: GOTO 199Ø
2Ø2Ø IF C > = 1 THEN POKE U + I,75
2Ø3Ø I = Ø:J = Ø
2Ø4Ø J = J + 1:I = PEEK (U + J): IF I < > Ø THEN 2Ø4Ø
2Ø5Ø U = U + J:J = Ø:C = Ø
2Ø6Ø FOR I = HO TO HU:A = (A = Ø)
2Ø7Ø IF I = 22 THEN 237Ø
2Ø8Ø IF A = Ø THEN 219Ø
2Ø9Ø FOR J = HL TO HR
21ØØ B = B%(J,I):C = C + 1
211Ø IF J = HR THEN 229Ø
212Ø IF NOT B THEN 216Ø
213Ø IF C = 3 THEN U = U + 1:C = 1: GOTO 215Ø
214Ø IF C = 2 THEN POKE U, PEEK (U) + 4Ø: NEXT
215Ø IF C = 1 THEN POKE U, PEEK (U) + 5: NEXT
216Ø IF C = 3 THEN POKE U, PEEK (U) + 64:U = U + 1:C = Ø:
    NEXT
217Ø IF C = 2 THEN POKE U, PEEK (U) + 8: NEXT
218Ø IF C = 1 THEN POKE U, PEEK (U) + 1: NEXT
219Ø FOR J = HR TO HL STEP - 1
22ØØ B = B%(J,I):C = C + 1
221Ø IF J = HL THEN 229Ø
222Ø IF NOT B THEN 226Ø
223Ø IF C = 3 THEN U = U + 1:C = 1: GOTO 225Ø
224Ø IF C = 2 THEN POKE U, PEEK (U) + 56: NEXT
225Ø IF C = 1 THEN POKE U, PEEK (U) + 7: NEXT
226Ø IF C = 3 THEN POKE U, PEEK (U) + 192:U = U + 1:C = Ø:
    NEXT
227Ø IF C = 2 THEN POKE U, PEEK (U) + 24: NEXT
228Ø IF C = 1 THEN POKE U, PEEK (U) + 3: NEXT
229Ø IF NOT B THEN 233Ø
23ØØ IF C = 3 THEN U = U + 1:C = 1: GOTO 232Ø
231Ø IF C = 2 THEN POKE U, PEEK (U) + 48: GOTO 236Ø
232Ø IF C = 1 THEN POKE U, PEEK (U) + 6: GOTO 236Ø
233Ø IF C = 3 THEN POKE U, PEEK (U) + 128:U = U + 1:C = Ø
234Ø IF C = 2 THEN POKE U, PEEK (U) + 16
235Ø IF C = 1 THEN POKE U, PEEK (U) + 2
236Ø NEXT I
237Ø IF C > Ø THEN U = U + 1
238Ø POKE U,Ø
239Ø REM ** Anzeigen des Shapes; Frage, ob anhängen **
24ØØ POKE TA,SN + 1: POKE 233,96: XDRAW SN + 1 AT XM + 15
    + SX,75 + SY: POKE 233,3: POKE TA,SN: FOR I = 1 TO
    16: XDRAW 8 AT XM + 15 + SX,75 + SY: FOR J = 1 TO 5Ø:
    NEXT : NEXT
241Ø HOME : VTAB 22: PRINT "Save Shape-Nr. ";SN + 1;"
    (J/N)";: GET E$
242Ø HCOLOR= Ø: FOR I = XM + 13 TO XM + 43: HPLOT I,65 TO
    I,1ØØ: NEXT : HCOLOR= Ø:A = Ø
243Ø IF E$ < > "J" AND E$ < > "N" THEN 241Ø
244Ø U = U + 1
245Ø IF E$ = "J" THEN SN = SN + 1: IF SN < N THEN TN = U -
    TA: POKE TA + 2 * SN + 2,TN - 256 * INT (TN / 256):
    POKE TA + 2 * SN + 3, INT (TN / 256)
246Ø POKE TA,SN
247Ø REM ** Anzeige Auswahlmenu **
248Ø HOME : VTAB 21: PRINT "(N)eues Shape zeichnen"
249Ø VTAB 22: PRINT "(Z)eigen der Shapes"
25ØØ VTAB 23: PRINT "(T)able auf Diskette ";: GET E$
251Ø ON (E$ = "T") GOTO 268Ø: ON (E$ = "Z") GOTO 257Ø
252Ø IF SN = N THEN HOME : VTAB 22: PRINT "Die Tabelle ist
    voll !": FOR I = 1 TO 2ØØØ: NEXT : GOTO 248Ø
253Ø HOME : VTAB 22: PRINT "Rahmen löschen (J/N): ";: GET
    E$
254Ø IF E$ = "N" THEN GOSUB 289Ø: GOTO 137Ø
255Ø IF E$ = "J" THEN HOME : VTAB 22: PRINT "Raster wird
    gelöscht !": FOR I = 1 TO 21: FOR J = 1 TO 21:B%(I,J)
    = Ø: NEXT : NEXT : GOTO 135Ø
256Ø REM ** Zeigen der bisher definierten Shapes **
257Ø IF PEEK (TA) = Ø THEN HOME : VTAB 22: PRINT "Es ist
    kein Shape definiert !": FOR I = 1 TO 2ØØØ: NEXT :
    GOTO 248Ø
258Ø HOME : VTAB 22: PRINT "<- , -> <RET> = Ende":N1 = 1:M
    = 1: POKE 233,96: XDRAW N1 AT XM + SX + 15,SY + 1ØØ
259Ø VTAB 23: HTAB 1: PRINT "Shape-Nr.: "N1"  ";: GET E$
26ØØ IF E$ < > CHR$ (8) AND E$ < > CHR$ (21) AND E$ < >
    CHR$ (13) THEN 259Ø
261Ø IF E$ = CHR$ (13) THEN XDRAW N1 AT XM + SX + 15,SY +
    1ØØ: POKE 233,3: GOTO 248Ø
262Ø M = M + (E$ = CHR$ (21)) - (E$ = CHR$ (8))
263Ø IF M < 1 THEN M = SN
264Ø IF M > SN THEN M = 1
265Ø XDRAW N1 AT XM + SX + 15,SY + 1ØØ: XDRAW M AT XM + SX
    + 15,SY + 1ØØ:N1 = M
266Ø GOTO 259Ø
267Ø REM ** Tabelle auf Diskette **
268Ø FF = 2: ONERR GOTO 293Ø
269Ø HOME : VTAB 22: PRINT "Name: ( "N1$" ) = <RETURN>"
27ØØ VTAB 23: INPUT "       ";N$: IF N$ = "" THEN N$ = N1$:
    GOTO 274Ø
271Ø PRINT "Wird als ST."N$" abgespeichert."
272Ø PRINT D$"BSAVE ST."N$",A$6ØØØ,L"U - TA + 1
273Ø N1$ = "ST." + N$: GOTO 275Ø
274Ø PRINT D$"BSAVE"N$",A$6ØØØ,L"U - TA + 1
275Ø HOME : VTAB 22: PRINT "Weiter (J/N): ";: GET E$: ON
    (E$ = "J") GOTO 248Ø: ON (E$ = "N") GOTO 276Ø: GOTO
    275Ø
276Ø POKE 216,Ø: TEXT : HOME : PRINT D$"RUN TABLE.EDITOR"
277Ø REM ** Bildschirmaufbau **
278Ø HGR : HCOLOR= 3: ROT= Ø: SCALE= 1
279Ø FOR I = 13 TO XM STEP 7: HPLOT I,6 TO I,YM: NEXT
28ØØ FOR I = 6 TO YM STEP 7: HPLOT 13,I TO XM,I: NEXT
281Ø FOR I = 1 TO 32: READ A,B,C: DRAW A AT B,C: NEXT
282Ø DATA 1,8,1Ø,5,8,38,1,5,73,3,1Ø,73,1,5,1Ø8,5,1Ø,1Ø8,2,
    5,143,3,1Ø,143
283Ø DATA 1,17,2,5,45,2,1,77,2,3,82,2,1,112,2,5,117,2,2,147,
    2,3,152,2
284Ø DATA 1,166,1Ø,5,166,38,1,163,73,3,168,73,1,163,1Ø8,5,
    168,1Ø8,2,163,143,3,168,143
285Ø DATA 1,17,157,5,45,157,1,77,157,3,82,157,1,112,157,5,
    117,157,2,147,157,3,152,157
286Ø RESTORE :A = Ø
287Ø FOR I = 6 TO 3Ø: HPLOT XM + 14,I TO XM + 38,I: NEXT
288Ø HPLOT XM + 14,4Ø TO XM + 38,4Ø TO XM + 38,64 TO XM +
    14,64 TO XM + 14,4Ø
289Ø HOME : VTAB 21: HTAB 1: PRINT "<S>tartpunkt setzen":
    VTAB 21: HTAB 36: PRINT "I"
29ØØ VTAB 22: HTAB 1: PRINT "<P>unkt, <L>oeschen
    Cursor ->": VTAB 22: HTAB 35: PRINT "J": VTAB 22:
    HTAB 37: PRINT "K"
291Ø VTAB 23: HTAB 1: PRINT "<RET> Editier-Ende": VTAB 23:
    HTAB 36: PRINT "M": RETURN
292Ø REM ** Fehlerbehandlung **
293Ø HOME : VTAB 22: IF PEEK (222) = 6 THEN PRINT "Diese
    Tabelle ist nicht auf der Disk !": GOTO 297Ø
294Ø IF PEEK (222) = 9 THEN PRINT "Die Diskette ist voll
    !": GOTO 297Ø
295Ø IF PEEK (222) = 4 OR PEEK (222) = 1Ø THEN PRINT
    "Disketten,- File-Schreibschutz !": GOTO 297Ø
296Ø PRINT "Eingabefehler !"
297Ø FOR I = 1 TO 2ØØØ: NEXT : POKE 216,Ø: ON FF GOTO
    117Ø,268Ø,298Ø
298Ø HOME : VTAB 22: PRINT "Wenigstens ein Punkt muss
    gesetzt sein !": FOR I = 1 TO 2ØØØ: NEXT : GOSUB
    289Ø: GOTO 136Ø
299Ø REM ** Catalog der Diskette **
3ØØØ PRINT : PRINT D$"CATALOG": GET E$: HOME : GOTO 117Ø
```

Der Trace-Fehler im BASIC.SYSTEM

oder wie man Katastrophen vermeidet

von Arne Schäpers

Cogito ergo erro
Ich denke, also irre ich mich
(frei nach R.Descartes)

Nachdem sich das BASIC.SYSTEM seit nunmehr rund zwei Jahren auf dem Markt befindet und damit von Tausenden im praktischen Einsatz getestet wurde, hat nun ein Autor der amerikanischen Zeitschrift „Call A.p.p.l.e." einen Programmierfehler ans Tageslicht gebracht, der dort jedoch nur angedeutet, aber nicht behoben wird. Das Erstaunliche an diesem Fehler: Es handelt sich nicht um eine exotische Kleinigkeit wie z.B. die CATALOG- und $48-Bugs von DOS 3.3, über die seitenlange Artikel geschrieben worden sind, sondern um einen kompletten Systemabsturz, der bei der Arbeit mit Applesoft jederzeit auftreten kann und unter Umständen das Programm zerstört. Die in diesem Beitrag vorgeschlagene Korrektur ist deshalb für jeden Applesoft-Programmierer eine unbedingte Notwendigkeit.

1. Auftreten und Auswirkungen

Das BASIC.SYSTEM benutzt die TRACE-Funktion von Applesoft zur ständigen Überprüfung des Programmablaufs:

- Beim Start des BASIC.SYSTEMs wird das TRACE-Flag von Applesoft gesetzt und bleibt von da an ständig aktiv. Der Befehl NOTRACE wird vom BASIC.SYSTEM unwirksam gemacht.
- Applesoft gibt bei der Ausführung eines Programms jeweils ein TRACE-Zeichen aus, wenn eine der beiden folgenden Bedingungen erfüllt ist:

– Der nächste Befehl befindet sich am Anfang einer Programmzeile.
– Innerhalb einer Programmzeile wurde ein Doppelpunkt („:") als Trennzeichen zwischen zwei Befehlen erkannt.

- Die Ausgabe des TRACE-Zeichens führt zu einem Sprung in das BASIC.SYSTEM, wo dieser „nächste Befehl" überprüft wird. FLASH schaltet z.B. die interne TRACE-Erkennung um. PRINT bewirkt, daß die nächste Ausgabe auf „Control-D" geprüft wird usw.
- Die Überwachungsroutine arbeitet nur dann korrekt, wenn der „nächste Befehl" am Anfang einer Zeile bzw. nach einem Doppelpunkt zulässig ist.

Unter „unzulässig" ist jeder Befehl als Applesoft-Token zu verstehen, der nur in Kombination mit anderen Befehlen ausgeführt werden kann und nicht am Anfang einer Zeile stehen darf, also:
– TAB(, SPC(← nur nach einem PRINT-Befehl erlaubt;
– TO, STEP ← nur nach einem FOR-Befehl erlaubt;
– FN, AT ← nur nach DEF bzw. HLIN/VLIN erlaubt;
– AND, NOT, OR ← logische Operatoren;
– +, -, *, /, ↑, >, <, = ← arithmetische und relationale Operatoren;
– SGN, INT, ABS, ... ← sämtliche Funktionen
Wird eines dieser reservierten Wörter in einem Variablennamen verwendet, reagiert Applesoft normalerweise mit einem „?SYNTAX ERROR" und bricht die Programmausführung ab. Unter dem BASIC.SYSTEM führt derselbe Fehler nicht zu einer Meldung, sondern zu einem Systemabsturz. Beispiele:

10 NOTENWERT = 4
Diese Zeile wird von Applesoft als 10 NOT ENWERT = 4 gelistet. Ihre Ausführung erzeugt unter der Version 1.1 des BASIC.SYSTEMs die Ausgabe von #10, danach hilft nur noch ein RESET weiter. Unter der Version 1.0 hat eine Zeile wie
10 POSITION = 100
dieselbe Wirkung.

10 ORDNUNG = X
Diese Zeile wird von Applesoft als 10 OR DNUNG = X gelistet. Unter der Version 1.1 bewirkt ihre Ausführung, daß das Volume-Directory der zuletzt benutzten Diskette in einer Endlosschleife immer wieder gelesen wird. Unter der Version 1.0 wird #10 ausgegeben, danach hängt das System.

Unter der Version 1.1 führt eine Zeile, die mit einem Pluszeichen beginnt, bei drei von fünf Versuchen zum Verlust des kompletten Programms. Dasselbe ist unter der Version 1.0 für den Befehl USR der Fall. Abhängig davon, ob das System mit einem 6502 oder 65C02 ausgerüstet ist, ergeben sich unterschiedliche Auswirkungen mancher Befehle. Der 65C02 führt nicht-definierte Opcodes als NOPs aus, der 6502 kommt dabei völlig durcheinander.

2. Detaillierte Herleitung

Um den Ablauf eines Applesoft-Programms nicht mehr als unbedingt nötig zu verlangsamen, ist die Überwachungsroutine des BASIC.SYSTEMs auf maximale Geschwindigkeit optimiert worden und arbeitet mit einem selbstmodifizierenden Sprung. **Disassembler-Listing 1** gibt die relevanten Stellen der Version 1.1 wieder,

Disassembler-Listing 2 die Sprungtabelle. Der Ablauf für die Version 1.0 ist exakt derselbe, nur daß die absoluten Adressen unterschiedlich sind:

– Auslesen des „nächsten Befehls" von Applesoft. Dieser Befehl ist als „Token" kodiert, wird also durch ein Byte dargestellt, dessen Wert zwischen $80 und $FF liegt.

– Der Wert des Tokens wird zur direkten Adressierung einer Tabelle verwendet. Wenn das dort gelesene Byte einen Wert größer/gleich $80 hat, folgt ein direkter Sprung zu Applesoft, ansonsten wird es als „Sprungoffset" eingesetzt.

– Ausführung des soeben modifizierten Sprungs. Für PRINT und IF ergibt sich ein „Sprungoffset" von $00 und damit die Fortsetzung des Programms auf der Adresse $9EFE, für LIST ein Sprung zu $9F11 usw.

– Je nach Art des Befehls werden verschiedene Flags des BASIC.SYSTEMs gesetzt. Die TRACE-Kontrollroutine endet in jedem Fall mit einem Sprung in den Applesoft-Interpreter hinein, wo der Befehl schließlich ausgeführt wird.

Sämtliche „zulässigen" Befehle von Applesoft haben Tokens im Wertebereich von $80..$BF. Die Sprungtabelle hat deshalb eine Länge von $40 Bytes. Da eine Begrenzung des Token-Wertes auf den Bereich $80..$BF fehlt, führen „unzulässige" Befehle über das Tabellenende hinaus und lesen Werte aus der nachfolgenden Tabelle, die die Startindexe für den Vergleich mit der internen Kommandotabelle (CMDTAB) enthält. Die Version 1.1 des BASIC.SYSTEM kennt das Kommando BYE, das in der Version 1.0 nicht enthalten ist. Aus diesem Grund sind die Startindexe unterschiedlich – und damit auch die Wirkung „unzulässiger" Befehle. Der modifizierte Sprung führt also „wild" in den folgenden Code der TRACE-Überwachung hinein – in den meisten Fällen zwischen Opcodes und die dazugehörigen Operanden. So ergibt sich z.B. für „OR" in der Version 1.1 ein Sprungoffset von $04 und die Befehlsfolge LDX $03AD,Y/CLV/STA $BE38.., jede nachfolgende Ausgabe führt zu einem Sprung in die CATALOG-Routine hinein.

3. TRACE.KORREKTUR

Da eine Verlängerung der Sprungtabelle um $40 Bytes aus naheliegenden Gründen ausscheidet, muß die TRACE-Kontrolle um einen Test auf „Token größer $BF" und eine entsprechende Verzweigung erweitert werden. Zwei Möglichkeiten bieten sich dazu an:

Disassembler-Listing 1

```
9EC6-   A0 00       LDY   #$00
9EC8-   B1 B8       LDA   ($B8),Y       ; Ist das nächste Statement
9ECA-   30 25       BMI   $9EF1           ein Token? Sprung wenn ja
9ECC-   F0 20       BEQ   $9EEE         ; <00>: Zeilenende in Applesoft
   ......
                                        ; ("automatisches" FRE bzw. Test)
   ......
9EE8-   A0 00       LDY   #$00          ; weiter in Applesoft:
9EEA-   B1 B8       LDA   ($B8),Y       ; Simulation von CHRGOT
9EEC-   C9 3A       CMP   #$3A          ; ":"?
9EEE-   4C 20 D8    JMP   $D820         ; Applesoft: GOCMD, NEWSTT (vgl. $9F2F)

Nächstes Byte im Applesoft-Programmtext ist Token
9EF1-   85 33       STA   $33           ; -> "Prompt"
9EF3-   A8          TAY                 ; Vergleich mit Tabelle, Adressierung
9EF4-   B9 99 B7    LDA   $B799,Y         über das Token ($80..$BF)
9EF7-   30 F5       BMI   $9EEE         ; ok - keine Aktion
9EF9-   8D FD 9E    STA   $9EFD         ; Tabellenwert => Sprungoffset
9EFC-   D0 **       BNE   $9***         ; dieser Sprung wird modifiziert!
9EFE-   85 33       STA   $33           ; PRINT und IF-Kommando: $33 = 00
9F00-   8D 4C BE    STA   CHRLAST       ; <- kein Abfangen doppelter <CR>s
9F03-   AD 03 B8    LDA   $B803         ; VSYSOUT auf STATE 4/8 ($9DA3):
9F06-   8D 38 BE    STA   VSYSOUT         Wenn als nächstes ein Ctrl-D aus-
9F09-   AD 04 B8    LDA   $B804           gedruckt wird -> State 8, ansonsten
9F0C-   8D 39 BE    STA   VSYSOUT+1       zurück zu State 4
9F0F-   D0 1D       BNE   $9F2E         ; "always": NEXT STATEMENT

9F11-   A9 01       LDA   #$01          ; LIST-Befehl: $33 = 01 - nur relevant
-------------------------------------------------------------------------
9F13-   85 33       STA   $33             für Ausgabe in eine OUT-Datei
-------------------------------------------------------------------------
9F15-   D0 17       BNE   $9F2E         ; Ausführung Applesoft-Statement

9F17-   A9 02       LDA   #$02          ; CALL-Befehl: $33 = 02 - nur relevant
9F19-   85 33       STA   $33             für Ausgabe in eine OUT-Datei
9F1B-   D0 11       BNE   $9F2E         ; Ausführung Applesoft-Statement

9F1D-   98          TYA                 ; LET-Befehl: Test Stringspeicherplatz,
9F1E-   4C CE 9E    JMP   $9ECE           danach Ausführung des Statements

9F21-   8C 41 BE    STY   DTRACE        ; TRACE-Befehl: Setzt DTRACE ON
9F24-   D0 04       BNE   $9F2A          und Applesoft TRACE-Flag ON

9F26-   8D 41 BE    STA   DTRACE        ; NOTRACE: DTRACE aus (Acc < $7F)
9F29-   88          DEY                 ; "NOTRACE": $9C, "TRACE": $9B
9F2A-   A9 4A       LDA   #$4A          ; Applesoft TRACE aus und danach
9F2C-   85 F2       STA   $F2           ; Ausführung des Befehls "TRACE"(!)

-Ausführung des Applesoft-Statements
-------------------------------------------------------------------------
9F2E-   98          TYA                 ; Token-Nummer bzw. "TRACE"
9F2F-   4C 20 D8    JMP   $D820         ; AS-Routinen GOCMD, danach NEWSTT
                                          Kein Return! (Stack v. $9EBE gesetzt)
-------------------------------------------------------------------------
9F32-   A9 00       LDA   #$00          ; RESUME-Kommando: setzt (222) auf 00.
9F34-   85 DE       STA   $DE             Das dürfte ein Bug Fix für
9F36-   98          TYA                   Applesoft sein.
9F37-   D0 B3       BNE   $9EEC         ; "always": Ausführung RESUME

-Sprung hierher, wenn Applesoft TRACE an
9F39-   20 00 9A    JSR   $9A00         ; SET REALIO
9F3C-   A9 A3       LDA   #$A3          ; "#"
9F3E-   20 ED FD    JSR   $FDED         ; "#" drucken
9F41-   A6 75       LDX   $75           ; $75/76: (LINNUM): Zeilennummer
9F43-   A5 76       LDA   $76             in Applesoft
9F45-   20 24 ED    JSR   $ED24         ; Applesoft: "PRINT A-X dezimal"
9F48-   A9 A0       LDA   #$A0
9F4A-   20 ED FD    JSR   $FDED         ; und ein Leerzeichen danach
9F4D-   20 8D 9A    JSR   $9A8D         ; SET DOSIO
9F50-   38          SEC                 ; erst danach wird das nächste
9F51-   4C C6 9E    JMP   $9EC6           Statement getestet
9F54-   A9 E3       LDA   #$E3          ; FLASH-Befehl: Applesoft druckt
9F56-   D0 02       BNE   $9F5A           danach $E3 (FLASHING "#") bei TRACE!

9F58-   A9 A3       LDA   #$A3          ; NORMAL und INVERSE: Applesoft druckt
9F5A-   8D 61 9F    STA   $9F61           wieder "#" bei TRACE
9F5D-   98          TYA
9F5E-   D0 8C       BNE   $9EEC         ; EXEC STATEMENT
```

● Ein echter „Patch" durch Ersetzen von drei Bytes durch einen „JMP" zu einer Routine, die sich in einem noch unbenutzten Speicherbereich des BASIC.SYSTEM befindet. Diese Routine holt die durch den „JMP" ersetzten Befehle nach, nimmt die Prüfung auf „größer $BF" vor und springt sodann wieder zur TRACE-Kontrolle zurück. Obwohl innerhalb des Variablenbereichs des BASIC.SYSTEMs genügend Platz vorhanden wäre und die entsprechenden Bereiche für die Versionen 1.0 und 1.1 dieselbe Adreßlage haben, wurde diese Lösung aus zwei Gründen verworfen:

– Das Startprogramm des BASIC.SYSTEM kopiert keine Daten in diesen Bereich hinein. Es wäre also eine Zusatzroutine erforderlich, die für die Verschiebung der TRACE-Erweiterung an ihren richtigen Platz sorgt. Die Startprogramme der Versionen 1.0 und 1.1 unterscheiden sich stark voneinander, die Zusatzroutine müßte also versionsabhängig geschrieben werden.

– Der Zeitbedarf der TRACE-Kontrolle würde durch die Sprünge für jeden Applesoft-Befehl um rund 15 Mikrosekunden erhöht, die Ausführungsgeschwindigkeit eines Applesoft-Programms damit um rund 1% gesenkt.

● Optimierung des Codes der TRACE-Kontrolle in einer Form, die vier Bytes freien Speicherplatz innerhalb der Routine selbst schafft und so den Einbau des Tests ermöglicht. Dieser Lösungsweg ist zwar im Ansatz erheblich komplizierter, hat aber abgesehen von einer minimalen Geschwindigkeitseinbuße (vier Mikrosekunden) den Vorteil, daß er versionsunabhängig geschrieben werden kann. Erfreulicherweise sind die TRACE-Kontrollen beider Versionen des BASIC.SYSTEMs bis auf eine verschobene Adreßlage absolut identisch.

Disassembler-Listing 2

```
B819- 8Ø 81 82 83 84 85 86 87          ; Applesoft-Tokens für die TRACE-
B821- 88 89 8A 8B 19 8D 8E 8F            Routine. Indiziert über das Token
B829- 9Ø 91 92 93 94 95 96 97            selber. MSB gesetzt: keine Aktion.
B831- 98 99 9A 23 28 5A 5A 56            MSB gelöscht: Sprung-Offset
B839- AØ A1 A2 A3 A4 A5 34 A7            innerhalb von VSYSOUT State 4
B841- A8 A9 1F AB AC ØØ AE AF            "Spezielle" Tokens:
B849- BØ B1 B2 B3 B4 B5 B6 B7            CALL/TRACE/NOTRACE/NORMAL/INVERSE
B851- B8 B9 ØØ BB 13 BD BE BF            FLASH/RESUME/LET/IF/PRINT/LIST

B859- 47 5F 18 3Ø 15 BD 3D 28          ; Indexe für den Vergleich mit CMDTAB,
B861- ØD 41 2Ø 5Ø 84 ØD Ø4 46            Tabellen-Absuche von hinten, d.h.
B869- 6A 4C 5C 35 A5 7Ø 2D 12            Start mit B95Ø. Wenn MSB gesetzt,
B871- 65 39 41 88 1C B5 D8               dann Kommandowort-Länge minus 1

-CMDTAB - Tabelle mit den einzelnen Kommandoworten
B878- 42 53 41 56 45 52 49 46  59 42 4C 4F 41 44 45 4C    BSAVERIFYBLOADEL
B888- 45 54 45 42 59 45 43 41  54 41 4C 4F 47 4F 5Ø 45    ETEBYECATALOGOPE
```

Zur Ausführung

Das Applesoft-Programm TRACE.KORREKTUR nimmt die Korrektur in vier voneinander getrennten Schritten vor. Die angegebenen Adressen gelten für die Version 1.1 des BASIC.SYSTEMs:

1) Der Programmcode im Bereich $9EF4 bis $9F31 wird um 2 bzw. 4 Bytes in Richtung höherer Adressen verschoben. Die im Disassembler-Listing 1 durch Trennstriche markierten Befehle werden dadurch überschrieben, der Bereich $9EF3..$9EF6 wird frei.

2) Die Prüfung auf „größer $BF" wird in den freigewordenen Bereich geschrieben, die Behandlung für LIST/CALL und TRACE/NOTRACE wird speicherplatzmäßig optimiert. Die Ausführung dieser vier Befehle benötigt danach eine Mikrosekunde mehr an Zeit und vier Bytes weniger Speicherplatz.

3) Sämtliche relativen Sprünge, die sich auf die verschobenen Programmteile beziehen, werden entsprechend angepaßt.

4) Die zur Ausführung von LET, CALL, TRACE und NOTRACE gehörigen Programmteile haben sich relativ zu dem selbstmodifizierenden Sprung um jeweils zwei Bytes verschoben. Für NORMAL, INVERSE, FLASH und RESUME ergibt sich eine Verschiebung um vier Bytes. Die Sprungtabelle wird für diese Befehle angepaßt.

Literatur

A.Schäpers, Das BASIC.SYSTEM. Lesebuch, Analyse, Tips und Tricks
Heidelberg 1986

Kurzhinweise

1. Zweck:
Korrektur des Trace-Fehlers im BASIC.SYSTEM
2. Konfiguration:
II+/e/c; ProDOS-BASIC.SYSTEM 1.0 oder 1.1
3. Test:
a) ProDOS und BASIC.SYSTEM starten
b) LOAD TRACE.KORREKTUR
c) Für BASIC.SYSTEM Version 1.1: keine Änderung
d) Für BASIC.SYSTEM Version 1.0: Zeilen 330 und 340 anpassen
e) RUN
f) Diskette mit BASIC.SYSTEM einlegen
g) <Return> drücken. Nach rund 10 Sekunden wird die korrigierte Version des BASIC.SYSTEM wieder auf die Diskette zurückgeschrieben und automatisch gestartet.
4. Sammeldisk:
TRACE.KORREKTUR
5. Sonstiges:
Programm TRACE.KORREKTUR muß zunächst mit CONVERT auf Ihre ProDOS-Arbeitsdiskette konvertiert werden.

A.U.G.E.-Mailboxen

Die Apple User Group Europe e.V. (A.U.G.E.) hat uns die Adressen der A.U.G.E.-Mailboxen zur Verfügung gestellt. Peeker-Leser können sich außer in der A.U.G.E.-Box überall als Gäste einloggen.
Datex-P-Mailbox:
A.U.G.E. Box: NUA 45 4298 41010,AUGE
INFSYS: NUA 44 6151 90343
DAILY SERVE: NUA 44 7261 40323
Lokale Mailboxen:
Berlin: - 030/8034656
Darmstadt: INFSYS-DA 06151/784158
Hamburg: Rappelkiste 040/373865
Köln: INFSYS-K 02203/34456
MA/HD: RNI 06203/45496
Münster: Micky 0251/619054
Münster: Maus 0251/522790
Wiesbaden/Ffm: CIA-CS 069/494201
Ruhrgebiet: A.U.S. 0203/705827
Saarbrücken: PFM 0711/3700978

TRACE.KORREKTUR

```
100 REM ORDNUNG = ORDNUNG (nicht KATASTROPHE)
110 REM Beseitigung eines fatalen Fehlers im
120 REM BASIC.SYSTEM (Versionen 1.0 und 1.1)
130 :
140 REM  AS, September '86
150 :
160 HIMEM: 38400
170 :
180 TEXT : HOME : PRINT "TRACE-FEHLERKORREKTUR"
190 VTAB 4: PRINT "Diskette mit BASIC.SYSTEM": PRINT
200 PRINT "einlegen und:": PRINT
210 INPUT "<RETURN>=Weiter, <CONTROL-C>=Ende ";X$
220 PRINT
230 PRINT  CHR$ (4);"UNLOCK BASIC.SYSTEM"
240 PRINT  CHR$ (4);"BLOAD BASIC.SYSTEM,TSYS,A$2000"
250 BYTES =  PEEK (48857) + 256 *  PEEK (48858): REM
    "ENDFILE"
260 :
270 REM ***********************************
280 :
290 REM Basisadressen für die Veränderungen, die
    gegebenen
300 REM Werte gelten für die Version 1.1.
310 REM Die Variable ADDR enthält immer einen relativen
    Wert!
320 :
330 BP  = 10480: REM ** VERSION 1.1 ** (V1.0: 10527)
340 TAB = 16921: REM ** VERSION 1.1 ** (V1.0: 17011)
350 :
360 IF  PEEK (BP) <  > 216 THEN  PRINT "Falsche
    Version!": STOP
370 IF  PEEK (BP + 3) = 201 THEN  PRINT "bereits
    korrigiert.": STOP
380 Z = 0: FOR X = 1 TO 40: READ Y:Z = Z + Y: NEXT
390 RESTORE : IF Z <  > 1809 THEN  PRINT "DATA-Fehler!":
    STOP
400 :
410 REM Durch Verschiebung werden vor dem veränderlichen
420 REM Sprung vier Byte Freiraum für den Test auf
430 REM "größer $BF" geschaffen.
440 :
450 FOR Y = 1 TO 2: READ AD,ED,OFFSET
460 : FOR X = ED TO AD STEP  - 1
470 ::Z =  PEEK (BP + X)
480 :: POKE BP + X + OFFSET,Z
490 : NEXT
500 NEXT
510 :
520 DATA 37,61,2,04,34,4
530 :
540 REM Einbau des Tests sowie der
    Speicherplatz-Optimierung
550 REM der Behandlung von LIST/CALL und TRACE/NOTRACE
560 :
570 FOR Y = 1 TO 3: READ ADDR,ANZAHL
580 : FOR X = 0 TO ANZAHL - 1
590 :: READ CODE: POKE BP + ADDR + X,CODE
600 : NEXT
610 NEXT
620 :
630 DATA 03,5: REM "CMP #$C0/TYA/BCS.."
640 DATA 201,192,168,176,246
650 DATA 40,1,02: REM LIST/CALL
660 DATA 64,2,208,04: REM TRACE/NOTRACE
670 :
680 REM Korrektur der relativen und absoluten Referenzen
690 REM innerhalb der verschobenen Programmteile
700 :
710 FOR X = 1 TO 4: READ ADDR,KWERT
720 :Z =  PEEK (BP + ADDR) + KWERT
730 IF Z < 256 THEN 760
740 ::Z = Z - 256: REM danach Übertrag
750 :: POKE BP + ADDR + 1, PEEK (BP + ADDR + 1) + 1
760 : POKE BP + ADDR,Z
770 NEXT
780 :
790 DATA 12,-6,14,+4,36,+4,46,+6
800 :
810 REM Korrektur der Werte in der "Token-Tabelle" für
820 REM die Befehle CALL, LET, TRACE und NOTRACE
830 REM sowie NORMAL, FLASH, INVERSE und RESUME
840 :
850 FOR Y = 1 TO 2: READ ANZAHL,KWERT
860 : FOR X = 1 TO ANZAHL: READ ADDR
870 :: POKE TAB + ADDR,  PEEK (TAB + ADDR) + KWERT
880 : NEXT
890 NEXT
900 :
910 DATA 4,-2,12,27,28,42: REM CALL/LET...
920 DATA 4,-4,29,30,31,38: REM NORMAL/FLASH...
930 :
940 REM ***********************************
950 :
960 PRINT "OK, wird zurückgeschrieben..."
970 PRINT  CHR$ (4);"BSAVE
    BASIC.SYSTEM,TSYS,A$2000,L";BYTES
980 PRINT  CHR$ (4);"LOCK BASIC.SYSTEM"
990 PRINT : PRINT "'Booting'...": PRINT  CHR$
    (4);"-BASIC.SYSTEM"
```

Programmverwaltung

Ein Übungsprogramm für Applesoft-Anfänger

von Nils Arndt

Dieses aus zwei Modulen bestehende Programm soll Ihnen helfen, eine übersichtliche Liste über Ihre Programme anzufertigen. Sie ersparen sich so langes Suchen nach einer bestimmten Datei, besonders wenn Sie viele Disketten besitzen. Ein kurzer Blick auf die mit dem Drukker erstellte und möglicherweise mit dem Modul „Alphasort" alphabetisch sortierte Liste genügt in Zukunft, um die Diskette herauszufinden, auf der sich das Programm befindet. Aber auch über die Programmart, die Programmlänge und die Bedienungsart gibt sie Auskunft. Die Daten von bis zu 324 Programmen werden als Textfile auf Diskette geschrieben.
Das Verwaltungsprogramm soll professionellen Dateiprogrammen keine Konkurrenz machen, sondern ist als nutzbringendes Übungsprogramm für Applesoft-Anfänger gedacht, die eigene Erweiterungen, z.B. BASIC.MASKE aus Peeker, Heft 12/86, einbauen können.

1. PROGRAMMVERWALT

1.1. Dateieingabe

Nach dem Start der Programmverwaltung mit RUN PROGRAMMVERWALT (von DOS 3.3 oder ProDOS) wählen Sie für die Eingabe die entsprechende Option im Menü. Geben Sie dann nacheinander den Programmnamen, die Programmart, die Programmlänge, die Nummer der Diskette, auf der das Programm gespeichert ist, und schließlich die Bedienungsart ein.
Diese fünf Angaben halte ich persönlich für geeignet, um schnell über ein Programm zu informieren. Sie können jedoch von Ihnen geändert werden. Dazu müssen Sie in den Zeilen 455 (Listenüberschrift), 565-595 (Eingabe) und 640-660 (Ausgabe auf dem Bildschirm) Veränderungen vornehmen. Gegebenenfalls ist es auch erforderlich, die Zwischenräume zwischen den einzelnen Angaben in den Zeilen 270-305 neu berechnen zu lassen.

Die Punkte hinter dem Cursor geben das Maximum der Zeichen an, die Sie bei der Eingabe verwenden können. Sie sollten es nicht überschreiten, da dies sonst Ursache für eine unübersichtliche Liste wäre. So darf der Programmname lediglich 23 Zeichen umfassen. Da er oft zu abstrakt ist, um über das Programm Auskunft zu geben, können Sie mit einem Schlagwort (höchstens 17 Zeichen) die Programmart definieren (z.B. Kopierprogramm, Grafik-Utility, Adventure, Compiler).

Die Bedienungsart, die bis zu drei Zeichen lang sein darf, ist abgekürzt anzugeben. Sie sparen so Zeit und Speicherplatz. „J" könnte zum Beispiel für Joystick, „K" für Keyboard, „P" für Paddle und „M" für Mouse stehen. Falls mehrere Bedienungsarten für ein Programm zutreffen, ist es zweckmäßig, sie nebeneinander (ohne Kommata), etwa in der Form „JPK", darzustellen.

Wenn Sie noch während der Eingabe eine Korrektur durchführen möchten, schreiben Sie bitte anstelle des Programmnamens **&V**. Sie werden daraufhin feststellen, daß die Programmnummer um eins abgenommen hat. Geben Sie nun die Daten nochmals ein.

Die Eingabe wird entweder nach 324 Programmen automatisch beendet oder dadurch, daß Sie anstelle des Programmnamens **&E** schreiben. Nachdem Sie den Dateinamen bestimmt haben, werden die Daten auf Diskette gespeichert.
Falls Sie eine bereits angelegte Datei später fortsetzen möchten, wählen Sie bitte im Menü die entsprechende Option.

1.2. Korrektur

Die Korrektur von bereits auf die Diskette geschriebenen Daten erfolgt durch die Menü-Option 3. Geben Sie die laufende Nummer des Datensatzes ein, die Sie z.B. der mit dem Drucker erstellten Liste entnehmen können. Sie haben nun Gelegenheit, Daten zu korrigieren oder zu löschen.

1.3. Bildschirm- und Druckerausgabe

Die Ausgabe der Daten auf dem Bildschirm erfolgt in derselben Form wie die Eingabe, nämlich untereinander, wobei immer die Daten von zwei Programmen gleichzeitig erscheinen.
Die Ausgabe der Daten mit dem Drucker geschieht der Übersichtlichkeit halber nebeneinander. Eine Zeile besteht aus 80 Zeichen. Der Programmname und die Programmart werden linksbündig, die laufende Programmnummer, die Programmlänge, die Diskettenummer und die Bedienungsart dagegen rechtsbündig ausgegeben. Eine in zwei Päckchen aufgeteilte DIN-A4-Seite besteht aus Daten von 54 Programmen. Insgesamt werden auf eine Seite 59 Zeilen gedruckt (2 * 27 Zeilen Programmdaten, eine Leerzeile und 2 * 2 Zeilen für die Überschrift der Päckchen). Wenn Sie mehr bzw. weniger Zeilen auf eine Seite gedruckt haben möchten, müssen Sie die Programmzeilen 260 und 265 ändern.

2. ALPHASORT

Nach dem Start des Sortierprogramms mit RUN ALPHASORT wird die gewünschte Datei von Diskette eingelesen, wenn sie mit dem Programm „Programmverwaltung" erstellt worden ist. Anschließend ordnet es die Programmnamen alphabetisch, was bei 324 Namen etwa zweieinhalb Minuten in Anspruch nimmt. Danach werden die Daten unter dem gleichen

Dateinamen wieder auf Diskette geschrieben. Das Sortieren geschieht folgendermaßen: „Alphasort" prüft nacheinander die Anfangsbuchstaben der Programmnamen und zwar zuerst auf den ersten, dann auf den zweiten, dritten ... Buchstaben des Alphabets. Den Programmnamen, die mit dem ersten, zweiten, dritten ... Buchstaben beginnen, und den zugehörigen anderen Programmdaten werden neue Feldvariablen zugewiesen. Je kleiner die Stelle des Anfangsbuchstabens des Dateinamens im Alphabet ist, desto eher erhalten die betreffenden Programmdaten neue Variablen, so daß die Programmnamen letztendlich alphabetisch geordnet sind.

Kurzhinweise

1. Zweck:
Programmverwaltung

2. Konfiguration:
II+/e/c in 40 Z/Z; DOS 3.3, Diversi-DOS oder ProDOS
3. Test:
RUN PROGRAMMVERWALT
4. Sammeldisk:
PROGRAMMVERWALT (Ein/Ausgabe-Modul)
ALPHASORT (Sortier-Modul)
PROGRAMMDATEI (5 Übungseinträge)

ALPHASORT

```
12Ø REM *** ALPHASORT ***
17Ø DIM B$(324),F$(324),G$(324),H$(324),A$(324),
    I$(324),K$(324),L$(324),E$(324),J$(324)
18Ø HOME :J = 1:Z = 1
19Ø INPUT "DATEINAME: ";N$: PRINT
2ØØ D$ =  CHR$ (4)
21Ø REM  *** LESEN DER DATEI ***
22Ø PRINT D$"OPEN ";N$
23Ø PRINT D$"READ ";N$
24Ø INPUT K: FOR A = 1 TO K: INPUT
    A$(A),K$(A),L$(A),E$(A),J$(A): NEXT
25Ø PRINT D$"CLOSE ";N$
26Ø HOME : PRINT "DIE PROGRAMMNAMEN WERDEN ALPHABETISCH
    GEORDNET (BITTE WARTEN! 324 NAMEN       = 2 1/2
    MINUTEN)."
27Ø REM  *** SORTIEREN ***
28Ø FOR A = 1 TO K
29Ø IF  LEFT$ (A$(A),1) =  CHR$ (64 + J) THEN I$(Z) =
    A$(A):B$(Z) = K$(A):F$(Z) = L$(A):G$(Z) = E$(A):H$(Z)
    = J$(A):Z = Z + 1
3ØØ IF Z = K + 1 THEN 33Ø
31Ø NEXT
32Ø J = J + 1: IF J < 27 THEN 28Ø
33Ø HOME : PRINT "DIE DATEI WIRD AUF DISKETTE
    GESCHRIEBEN!"
34Ø REM  *** SCHREIBEN DER DATEI ***
35Ø PRINT D$"OPEN ";N$
36Ø PRINT D$"WRITE ";N$
37Ø PRINT K: FOR I = 1 TO K: PRINT I$(I): PRINT B$(I):
    PRINT F$(I): PRINT G$(I): PRINT H$(I): NEXT
38Ø PRINT D$"CLOSE ";N$
```

PROGRAMMVERWALT

```
11Ø REM *** PROGRAMMVERWALTUNG ***
135 X = 1:K = 324
14Ø DIM N$(324),AR$(324),L$(324),D$(324),J$(324)
145 REM *** MENUE ***
15Ø TEXT : HOME : GOSUB 4Ø5
155 PRINT "*";: INVERSE : PRINT "
    PROGRAMMVERWALTUNG          ";: NORMAL : PRINT "*"
16Ø PRINT "*";: INVERSE : PRINT "     ";: FLASH : PRINT
    "GESCHRIEBEN VON NILS ARNDT";: INVERSE : PRINT "
    ";: NORMAL : PRINT "*"
165 GOSUB 4Ø5: POKE 34,4
17Ø VTAB 7:  PRINT "<1> PROGRAMMEINGABE"
175 VTAB 9:  PRINT "<2> PROGRAMMEINGABE FORTSETZEN"
18Ø VTAB 11: PRINT "<3> KORREKTUR"
185 VTAB 13: PRINT "<4> AUSGABE MIT DEM DRUCKER"
19Ø VTAB 15: PRINT "<5> AUSGABE AUF DEM BILDSCHIRM"
195 VTAB 17: PRINT "<6> ENDE"
2ØØ VTAB 23: HTAB 1: PRINT "WAEHLEN SIE BITTE (1-6): ";:
    GET A$
2Ø5 ON  VAL (A$) GOTO 22Ø,325,345,24Ø,61Ø,395
21Ø GOTO 2ØØ
215 REM *** PROGRAMMEINGABE ***
22Ø X = 1:K = 324
225 HOME : GOSUB 41Ø: GOSUB 42Ø: HOME : GOSUB 435: FOR I
    = X TO K: GOSUB 56Ø: GOSUB 435: NEXT I
23Ø HOME : PRINT :K = I - 1: GOSUB 44Ø: GOSUB 445: GOSUB
    42Ø: GOTO 15Ø
235 REM  *** PROGRAMMAUSGABE (DRUCKER) ***
24Ø HOME : PRINT : GOSUB 45Ø: IF A$ = "J" THEN 25Ø
245 PRINT : PRINT : GOSUB 44Ø: GOSUB 465: GOSUB 515
25Ø HOME : PRINT : PRINT "MACHEN SIE BITTE DEN DRUCKER
    STARTKLAR, UND DRUECKEN SIE EINE TASTE...";: GET A$:
    PRINT : HOME : PRINT  CHR$ (4)"PR#1": PRINT : GOSUB
    455
255 FOR I = 1 TO K
26Ø A = (I - 55) / 54: IF I > 1 AND A =  INT (A) THEN
    PRINT  CHR$ (4)"PR#Ø": HTAB 1: INVERSE : PRINT "LEGEN
    SIE BITTE NEUES PAPIER EIN, UND   DRUECKEN SIE EINE
    TASTE...";: GET A$: PRINT : NORMAL : PRINT  CHR$
    (4)"PR#1": PRINT : GOSUB 455
265 A = (I - 28) / 54: IF A =  INT (A) THEN  PRINT : GOSUB
    455
27Ø HTAB 1:W =  LEN ( STR$ (I)): IF W = 3 THEN 28Ø
275 FOR U = 1 TO (3 - W): PRINT " ";: NEXT U
28Ø PRINT I;
285 HTAB 6: PRINT N$(I);
29Ø HTAB 3Ø: PRINT AR$(I);
295 W =  LEN (AR$(I)):N = 19 - W + 3 -  LEN (L$(I)): GOSUB
    415: PRINT L$(I);
3ØØ N = 1Ø -  LEN (D$(I)): GOSUB 415: PRINT D$(I);
3Ø5 N = 13 -  LEN (J$(I)): GOSUB 415: PRINT J$(I)
31Ø NEXT I
315 PRINT : PRINT  CHR$ (4)"PR#Ø": HOME : GOSUB 42Ø: GOTO
    15Ø
32Ø REM  *** PROGRAMMEINGABE FORTSETZEN ***
325 HOME : PRINT : GOSUB 45Ø: IF A$ = "J" THEN 335
33Ø PRINT : PRINT : GOSUB 44Ø: GOSUB 465: GOSUB 515
335 X = K + 1:K = 324: GOTO 225
34Ø REM  *** KORREKTUR ***
345 HOME : PRINT : GOSUB 45Ø: IF A$ = "J" THEN  PRINT :
    GOTO 355
35Ø PRINT : PRINT : GOSUB 44Ø: GOSUB 465: GOSUB 515
355 PRINT : PRINT "WOLLEN SIE DATEN LOESCHEN ODER
    VERBESSERN (L/V)? ";: GET A$
36Ø PRINT : PRINT : INPUT "WELCHE NUMMER HAT DAS
    PROGRAMM? ";I
365 IF A$ = "L" THEN  PRINT : PRINT "BITTE WARTEN (MAX.
    1Ø SEK.)!": FOR J = I TO K:N$(J) = N$(J + 1):AR$(J) =
    AR$(J + 1):L$(J) = L$(J + 1):D$(J) = D$(J + 1):J$(J)
    = J$(J + 1): NEXT :K = K - 1: PRINT : PRINT "DIE
    PROGRAMMDATEN WURDEN GELOESCHT!": GOTO 375
37Ø HOME : GOSUB 435: GOSUB 56Ø: GOSUB 435
375 PRINT : PRINT "MOECHTEN SIE NOCH EINE KORREKTUR
    DURCHFUEHREN (J/N)? ";: GET A$
38Ø IF A$ = "J" THEN  HOME : GOTO 355
385 HOME : PRINT : GOSUB 44Ø: GOSUB 445: HOME : GOSUB
    42Ø: GOTO 15Ø
39Ø REM  *** PROGRAMMENDE ***
395 TEXT : HOME : END
4ØØ REM  *** ZEHN KLEINE UNTERPROGRAMME ***
4Ø5 PRINT "****************************************":
    RETURN
41Ø VTAB 7: PRINT "BENUTZEN SIE <&E> ZUM BEENDEN DER
    EINGABE UND <&V> ZUM VERBESSERN.": RETURN
415 FOR U = 1 TO N: PRINT " ";: NEXT U: RETURN
42Ø PRINT : PRINT : PRINT "DRUECKEN SIE BITTE EINE TASTE
    (<CTRL-C> ---> MENUE)...";: GET A$: PRINT
425 IF A$ =  CHR$ (3) THEN 15Ø
43Ø RETURN
435 PRINT "----------------------------------------":
    RETURN
44Ø HOME : PRINT : INPUT "GEBEN SIE BITTE DEN NAMEN DER
    DATEI     EIN: ";NA$: PRINT : RETURN
445 PRINT : PRINT "DIE DATEI WIRD AUF DISKETTE
    GESCHRIEBEN!": GOSUB 475: RETURN
45Ø PRINT "IST DIE DATEI SCHON GELADEN              WORDEN
    (J/N)? ";: GET A$: PRINT : RETURN
```

PEEKER Börse

Biete Hardware

Apple II+ Nachbau mit Accelerator, Z80B 6Mhz, Ram Floppy 256K, RS232 u. V24, 80Z. Karte, L. Karte, 35 + 160 Track Laufwerk, Controller, IBM Look, für DM 2900,–. Tel. 02303/13144 nach 15 Uhr.

16/32 Bit für Apple II 68000 mit 128K von IBS (AP20), für DM 600,–. Tel. 02303/13144 nach 15 Uhr

Apple IIe enhanced, Monitor III, Disk II + Contr. (Original)! 80Z/Z + 64K, 128K-Karte, Z80A. VB! Tel. (HH) 040/22 38 48 zw. 15 und 18 Uhr

Apple IIc, mit Monitor, 2. Laufwerk u. CP/M Card, sowie umfangr. Literatur u.v.a. zu verkaufen. Preis VB. Tel. 098/4 30 09 85

ITOH 8510 FF Matrixdrucker Schnittst. par./ser. werksgepr. mit autom. EB-Einzug-Aufsatz NP 2300,– für DM 1100,–. K. Freimann, Andelsbachstr. 2a, 7887 Laufenburg, 07763/16 58 8-12 h.

Apple IIe + Disk II + Mon. + SSC + IMAGEWRITER + Erw. 80Z + 512K-RAMDISK für Pascal + Pascal 1.2 DM 3600,–. Alexander Seggerman, Tel. 06172/30 34 74

Apple IIc mit CP/M-2Floppy-Monit. Scribe Drucker-Maus-Joystick 3 Diskettenboxen usw. VB DM 4000,–. Tel. 06898/67680 ab 20.00 Uhr

Orig. Apple II europlus, 2 Laufwerke, Monitor, 80 Zeichen, Z80-Karte, sep. Tastatur, serielle u. parallele Schnittstelle, viel Software, Bücher Literatur VB DM 1500,–. Tel. 040/7 39 99 05

Apple IIc + Monitor + Imagewriter + Maus + Appleworks + Merlin-Pro + Elite + Flight-Simulator + Kyan-Pascal + Bücher im Wert von DM 800,– + alle Peeker nur kompl. VB DM 3000,– Tel. 0911/42 87 12

MAC f. Studenten (sowie einige Apple-Prod.) zu Sonderpreisen!! Datent. Schoeben U., Bahnhofstr. 12 ½, 8900 Augsburg, Tel. 0821/15 23 77 G

IBS-Space 84 IBM-Geh. ACS-Tastatur 2 Taxan Disk m. Contr. Z80B, STAR-Gemini 10xm. Interf. VB 1950,– Siemens Disk + Contr. 250,– MS-Fortran 80 300,– Tel.: 02932/24120

Apple III 256K Monitor ext. LW Lit. Software, Profile, Drucker LQ 2000, VB DM 6500,– Tel. 08507/665 abends

II+ komp., 64K, Z80, 80Z, 2 Teac-LW 55F (160 + 35Spur), ect. Tastatur, Monitor, BS.: DOS + CP/M + Lit + ca. 100 Disks (SW umsonst) NP ca. DM 4800,– für DM 2800,–. Tel. 0631/66914

Suche Hardware

Suche Image-Writer/EPSON FX80 i. Apple IIe-Interface Tel. 02472/1680

Biete Software

Apple IIe 128K, IIc Software DMP-Charger 2.2 Neuzeichen – DM 100,–, Reportworks AWD-Auswerter – DM 300,–, SUPERCALC 3A TABKALK+GRAPH – DM 500,–. Einkaufskont. Bethel, Tel. 0521/144 38 15

Kyan-Pascal: Include-File-Tool mit über 20 Files 20,– (Schein) Info/Liste für DM 0,60 in Bfm. R. Winzen, Daimlerstr. 70, 4040 Neuss 1

Ärgern Sie sich nicht auch, daß Sie mit AppleWorks keine Serienbriefe schreiben können? Dann brauchen Sie AppleCircular! Es verknüpft Text und Daten und schreibt so persönliche Einladungen. Werbebriefe usw. Programm und kostenl. Info bei: Johannes Pellenz, Kiefernweg 4, 6550 Bad Kreuznach. Tel. 0671/63173

* **DISKETTEN** *
* 5¼", 48tpi, **DM 0,99,** 2D *
* 3½", 135tpi, **DM 3,19** IDD *
* 3" Schneider **DM 5,85** *
* auch andere, bes. Garantie *
* Allg. Austro-AG, Ringstr. 10 *
* D-8057 Eching, *
* Tel. 08133/6116 G *

Apple II: DFÜ-Kermit, Pascal satt, Public Domain in DOS u. CP/M. Je Volume DM 15,–. Bahnhofssimulation, Sprache, Schulprogramme. Gratisinfo: Fa. Waltraud Muhle Waldwinkel 3, 2105 Seevetal 3.

Gewerbliche Anzeigen sind mit G gekennzeichnet.

Suche Software

ACHTUNG! Wer hat das Grafik- programm Gutenberg 85 (= FONTRIX) für den Apple IIe in verbesserter Form (speziell angepaßt für den IIe und eine Harddisk, außerdem als ProDOS-Version anstelle DOS 3.3 und Double-Hires anstatt Hires)? Stefan Richter, Kalte Marter 3, 8650 Kulmbach

Tausch

ATARI ST Public Domain Softwaretausch. Liste an: Matthias Meyer, Postfach 191, CH-9001 St. Gallen.

Verschiedenes

APPLE REPARATUREN
(auch compatible M-boards, z.B. **Atlas, Arca, CES, Datastar, Dipa, Lasar, Mewa, PC-48 + 64, Plato, Radix, o. ae.) sowie Zusatzkarten und Disk-Drives** führt unser Spezialistenteam mit mehr als 5-jähriger Kunden- und Reparatur-Dienst-Erfahrung, garantiert zuverlässig und besonders kostengünstig aus.
Bitte genaue Fehlerangabe sowie Tel. Nr. für evtl. Rückfragen nicht vergessen.
Auf Wunsch Kostenvoranschlag.
aaa-electronic gmbh
Habsburgerstr. 134, 7800 Freiburg, Tel. 0761/276864, Tx. 772642aaad

Apple II+ Motherboard-Reperatur DM 100,– Tel. Mo-Do 0511/40 – 6604 Rüter

Original Peeker-Hefte 1/84-12/86 komplett DM 75,– Tel. 06121/54530

Für Ihre Unterlagen

Abonnement bestellt

am: _____

Vertrauensgarantie:
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag GmbH, Postfach 10 28 69, 6900 Heidelberg innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

Peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

Peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

Peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER

Abo-Karte

Ja, ich möchte **Peeker** abonnieren.
Liefern Sie mir **Peeker** ab Ausgabe . . . . . zum Jahresbezugspreis von z.Zt. DM 75,– (Inland) inkl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt z.Zt. DM 75,– plus DM 20,– Versandspesen.

X _____
Datum 1. Unterschrift

Bitte lesen!

Vertrauensgarantie: Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag GmbH, Postfach 10 28 69, 6900 Heidelberg innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

X _____
Datum 2. Unterschrift

Verlagshinweis: Das Abonnement verlängert sich zu den jeweils gültigen Bedingungen um ein Jahr, wenn es nicht 2 Monate vor Jahresende schriftlich gekündigt wird.

Wir können nur Bestellungen mit zwei Unterschriften bearbeiten.

PEEKER

Buch-Karte

Bitte senden Sie mir gegen Rechnung folgende Bücher:

- ☐ Bühler, Applesoft-BASIC, 3-7785-1094-0, DM 38,–
- ☐ Eggerich, dBase II, Bd. 1, 3-7785-1147-5, DM 39,80
- ☐ Eggerich, dBase II, Bd. 2, 3-7785-0987-X, DM 39,80
- ☐ Eggerich, dBase II, Bd. 3, 3-7785-0988-8, DM 39,80
- ☐ Gabriel, Applewriter, 3-7785-1234-X, DM 35,–
- ☐ Hagenmüller, Microsoft-BASIC, Bd. 1, 3-7785-1038-X, DM 38,–
- ☐ Juhnke/Redlin, Apple Pascal, Bd. 1, 3-7785-1246-3, DM 42,–
- ☐ Kehrel, Apple Assembler lernen, Bd. 1, 3-7785-1151-3, DM 38,–
- ☐ Kehrel, Apple Assembler lernen, Bd. 2, 3-7785-1170-X, DM 38,–
- ☐ Schäpers, ProDOS Analyse, 3-7785-1134-3, DM 68,–
- ☐ Schäpers, Bewegte Apple-Graphik, 3-7785-1150-5, DM 58,–
- ☐ Stiehl, Apple DOS 3.3, 3-7785-1297-8, DM 28,–
- ☐ Stiehl, Apple ProDOS, Bd. 1, 3-7785-1098-3, DM 28,–
- ☐ Stiehl, Apple ProDOS, Bd. 2, 3-7785-1036-3, DM 30,–
- ☐ Stiehl, Apple Assembler, 3-7785-1047-9, DM 34,–
- ☐ Wassermann, Apple IIc Handbuch, 3-7785-1157-2, DM 35,–

_____
Datum Unterschrift

PEEKER

Software-Karte

Bitte senden Sie mir gegen Rechnung folgende Disketten:

- ☐ Peeker-Sammeldiskette, einzeln
  Disk# _____, Disk# _____
  Disk# _____, Disk# _____
  Preis je Disk DM 28,– (einzeln)
- ☐ Peeker-Sammeldiskette, im Fortsetzungsbezug ab Disk# _____
  (Mindestbezug 6 Disketten)
  Preis je Disk DM 20,–
- ☐ Apple DOS 3.3, Begleitdisk., DM 28,–
- ☐ ProDOS, Band 1, Begleitdisk., DM 28,–
- ☐ ProDOS, Band 2, Begleitdisk., DM 28,–
- ☐ Apple Assembler, Begleitdisk., DM 28,–
- ☐ ProDOS-Editor 1.0, Programm, DM 98,–
- ☐ MMU 2.0, Programm, DM 98,–
- ☐ INPUT 2.0, Programm, DM 98,–
- ☐ DB-Meister, Programm, DM 290,–
- ☐ Superquick, Programm, DM 48,–
- ☐ Turtle Graphics, Programm, DM 98,–
- ☐ Disk 40, Programm, DM 48,–
- ☐ Kyan-Pascal 2.0, Programm, DM 170,–
- ☐ Fast-Writer, DOS 3.3, DM 128,–
- ☐ Fast-Writer, ProDOS, DM 128,–
- ☐ Double-Hires-Tools für Applesoft, DM 28,–
- ☐ Double-Hires-Tools für Kyan, DM 28,–
- ☐ Kyan-Toolkit Nr. _____, DM _____

_____
Datum Unterschrift

Abo-Karte

Name

Firma

Straße

PLZ/Ort

Ich wünsche jährliche Berechnung durch:
☐ Verlagsrechnung ☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank/PschA

Bankleitzahl

Kto.-Nr.

Bitte freimachen

POSTKARTE

Peeker
Leserservice

Dr. Alfred Hüthig Verlag GmbH

Postfach 10 28 69

6900 Heidelberg

Buch-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Bitte freimachen

POSTKARTE

Peeker
Buchabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Bitte freimachen

POSTKARTE

Peeker
Softwareabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1021-5

„Input 2.0" liegt wahlweise in der Bank 1 oder Bank 2 der Language Card und wird durch einen kurzen Driver in den unteren 48K aufgerufen.
Für jedes Feld der Bildschirmmaske lassen sich u. a. definieren: Feldlänge (bis zu 255 Zeichen) – Vtab – Htab – Datentyp (insgesamt 8 Typen) – Scrollflag (starre oder dynamische Maske) – Ctrlflag – Füllflag – Löschflag – Bildschirmflag (40- oder 80-Z-Darstellung). Innerhalb eines Eingabefeldes besteht jeder denkbare Redigierkomfort (Insert, Delete, Rubout, Restore usw.).
Gerätevoraussetzung: Apple IIe oder IIc; ferner Apple II+ im 40-Zeichenmodus

MMU 2.0

Memory Managements Utilities

für die Apple IIe 64K-Karte DOS 3.3 (und ProDOS)

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7787-1023-1

Insgesamt enthält die neue „MMU 2.0"-Diskette über 25 Programme, die neue Einsatzmöglichkeiten für die Extended 80 Column Card (erweiterte 80-Z-Karte = 64K-Karte für den Apple IIe) erschließen. Ein Teil der Programme laufen auch auf dem Apple II Plus, doch ist „MMU 2.0" primär für 64K-Karte-Besitzer gedacht.

Gerätevoraussetzung: Apple IIe mit 64K-Karte oder IIc

DISK 40

Disketten-Organisationsprogramm für DOS-3.3-35-40 Spuren

von Hermann Seibold und Dipl.-Ing. Udo Marin, 1986, Programmdiskette mit Anleitung, DM 48,–

Durch eine einfach zu bedienende Menüführung können DOS-3.3-Disketten umfangreich bearbeitet oder kopiert werden.
– Tabellarische Ausgabe der Diskettenbelegung
– Ordnen des Catalogs
– „Undelete"n von versehentlich gelöschten Dateien
– Vergleichen von Disketten, Dateien oder DOS-Spuren
– Kopieren von Disketten, Dateien oder DOS-Spuren
– Formatieren von Daten-Disketten
– Erweitern auf 40 Spuren bei bestehenden 35-Spur-Disketten
– Ändern des Boot-Programms
– File-Editor zum Editieren von Disketten-Dateien
– Komfortabler Sektor-Editor für Hex- und ASCII-Darstellung
–VTOC-Editor, z. B. zur Freigabe der DOS-Spuren

Hüthig Software Service, Postfach 10 28 69, D-6900 Heidelberg

```
455 PRINT "NR.: PROGRAMMNAME:            PROGRAMMART:
    LAENGE:  DISK NR.:  BEDIENUNGSART:"
460 PRINT
    "═══════════════════════════════════════
    ═══════════════════════": RETURN
465 PRINT : PRINT "DIE DATEI WIRD VON DISKETTE GELESEN!":
    RETURN
470 REM  *** DIE DATEI WIRD AUF DISK GESCHRIEBEN ***
475 PRINT  CHR$ (4)"OPEN ";NA$
480 PRINT  CHR$ (4)"WRITE ";NA$
485 PRINT K
490 FOR I = 1 TO K
495 PRINT N$(I): PRINT AR$(I): PRINT L$(I): PRINT D$(I):
    PRINT J$(I)
500 NEXT I
505 PRINT  CHR$ (4)"CLOSE ";NA$: RETURN
510 REM  *** DATEI WIRD VON DISK GELESEN ***
515 PRINT  CHR$ (4)"OPEN ";NA$
520 PRINT  CHR$ (4)"READ ";NA$
525 INPUT K
530 FOR I = 1 TO K
535 INPUT N$(I),AR$(I),L$(I),D$(I),J$(I)
540 NEXT I
545 PRINT  CHR$ (4)"CLOSE ";NA$
550 RETURN
555 REM  *** PROGRAMMEINGABE ***
560 PRINT "PROGRAMM NR. : ";I
565 PRINT "PROGRAMMNAME : ....................";: HTAB
    16: INPUT "";N$(I)
570 IF N$(I) = "&E" THEN 230
575 IF N$(I) = "&V" THEN I = I - 2: RETURN
580 PRINT "PROGRAMMART  : ...............";: HTAB 16:
    INPUT "";AR$(I)
585 PRINT "LAENGE       : .....";: HTAB 16: INPUT
    "";L$(I)
590 PRINT "DISK NR.     : ......";: HTAB 16: INPUT
    "";D$(I)
595 PRINT "BEDIENUNGSART: ...";: HTAB 16: INPUT "";J$(I)
600 RETURN
605 REM  *** PROGRAMMAUSGABE (BILDSCHIRM) ***
610 HOME : PRINT : GOSUB 450: IF A$ = "J" THEN 620
615 PRINT : PRINT : GOSUB 440: GOSUB 465: GOSUB 515
620 HOME : GOSUB 420: HOME
625 FOR I = 1 TO K
630 GOSUB 435
635 PRINT "PROGRAMM NR. : ";I
640 PRINT "PROGRAMMNAME : ";N$(I)
645 PRINT "PROGRAMMART  : ";AR$(I)
650 PRINT "LAENGE       : ";L$(I)
655 PRINT "DISK NR.:    : ";D$(I)
660 PRINT "BEDIENUNGSART: ";J$(I)
665 A = I / 2: IF A =  INT (A) THEN  GOSUB 435: GOSUB 420:
    HOME
670 NEXT I
675 IF A =  INT (A) THEN 150
680 GOSUB 435: GOSUB 420: GOTO 150
```

ABU – Ampersand-BASIC-Utility

von Jörg Bettray

1. Was ist ABU?

ABU ist eine Erweiterung des Applesoft-BASIC um mehrere Grafikbefehle und um drei erweiterte BASIC-Befehle. Was diese Befehle bewirken und wie sie angewendet werden, wird im BASIC-Programm ABU. DEMO demonstriert.

2. Die neuen Befehle

In der folgenden Beschreibung der neuen Befehle bezeichnen X, X1, Y, Y1 usw. beliebige numerische Ausdrücke, ansonsten werden Parameter klein geschrieben. <pagenum> steht für die Nummer der Grafikseite (1 oder 2), <color> steht für die Farbnummer (0..8). Spitze und runde Klammern dürfen nicht als Befehl mit eingegeben werden. Parameter in runden Klammern sind optional, d.h. sie können bei der Eingabe des Befehls wegfallen. Bei jedem Befehl wird erst die korrekte Syntax, dann die Wirkung angegeben.

& HELP:
HELP löscht den Bildschirm und gibt alle Befehle von ABU (Ampersand BASIC Utilities) und deren Syntax aus.

& PAGE <pagenum>:
PAGE setzt die internen Zeiger auf die angegebene Grafikseite. Sämtliche Befehle von ABU und die Applesoft-Befehle (AS-Befehle) greifen auf diese Zeiger zurück.

& HIRES:
HIRES schaltet auf die ausgewählte Grafikseite um, ohne sie zu löschen. Dabei wird die Routine FULL (s.u.) aufgerufen, so daß in jedem Fall die volle Grafikseite angezeigt wird. Möchte man in den unteren 4 Zeilen Text anzeigen, muß man anschließend die Routine MIX aufrufen.

& FULL:
FULL zeigt die volle Grafikseite an, d.h. die unteren 4 Textzeilen verschwinden vom Bildschirm. Dieser Befehl funktioniert allerdings nur bei eingeschalteter Grafik.

& MIX:
MIX zeigt Grafik und 4 Zeilen Text an. Auch dieser Befehl funktioniert nur bei eingeschalteter Grafik. Hat man allerdings die zweite Grafikseite gewählt, werden auch die 4 Zeilen Text der zweiten Textseite angezeigt. Da diese bei dezimal 2048 ($0800) anfängt und hier normalerweise auch das BASIC-Programm liegt, sind normalerweise nur blinkende und inverse Zeichen zu sehen. Da alle Ausgaberoutinen des AS-Interpreters nur auf die erste Textseite zugreifen, lohnt es sich nicht, den Befehl MIX in Verbindung mit der zweiten Grafikseite zu benutzen.

& BACKCOL <color>:
BACKCOL setzt die programminternen Zeiger auf die ausgewählte Hintergrundfarbe.

& CLS:
CLS löscht die Grafikseite, die durch PAGE ausgewählt wurde. Dabei werden keine anderen Veränderungen vorgenommen, d.h. es wird weder auf die zu löschende Grafikseite umgeschaltet noch die MIX/FULL-Einstellung verändert. Man kann also auch eine Grafikseite verdeckt löschen und dann erst mittels HIRES auf die Grafikseite umschalten.
Dieser Befehl entspricht nicht dem AS-Befehl HGR, da der Bildschirm mit der durch BACKCOL ausgewählten Hintergrundfarbe gefüllt wird. Bei der Hintergrundfarbe Schwarz entspricht dies dem Grafiklöschen. Wird BACKCOL 8 ausgewählt, wird der Bildschirm bitmäßig invertiert. Diese Art wurde deshalb gewählt, weil die farbmäßige Invertierung nur auf Farbmonitoren zu sehen wäre. Auf einem monochromen Monitor würden sämtliche Punkte nur um einen halben Punkt in der Horizontalen verschoben. Die bitmäßige Invertierung hingegen ist auf beiden Arten von Monitoren zu sehen. Wer trotzdem die farbmäßige Invertierung vorzieht, kann sich bei der Beschreibung des Programmes darüber informieren, wie man diese Routine ändert.

& COL <color>:
COL bestimmt die Zeichenfarbe für sämtliche folgenden Zeichenbefehle, auch für die normalen AS-Befehle. COL unterscheidet sich allerdings von dem AS-Befehl HCOLOR= durch die zusätzliche Farbe INVERS (color 8). Wird mit dieser Farbe gezeichnet, so werden die gesetzten Punkte auf dem Bildschirm bitweise invertiert. Alle AS-Befehle ignorieren diese Farbeinstellung und zeichnen statt dessen mit der Farbe Weiß.

& PSET <x>,<y>:
PSET setzt einen Punkt mit den angegebenen Koordinaten in die Grafikseite. Die-

ser Befehl entspricht dem AS-Befehl HPLOT <x>,<y>; der ABU-Befehl kennt die zusätzliche Zeichenfarbe INVERS.

& LINE (<x1>,<y1>) TO <x2>,<y2> (... TO <xN>,<yN>):
LINE verbindet die angebenen Koordinaten mit Linien. Auch hier gilt die zusätzliche Zeichenfarbe INVERS. Es können beliebig viele Koordinatenpaare angegeben werden, allerdings wird die Anzahl von der maximalen Länge einer BASIC-Zeile begrenzt. Man kann aber in der nächsten Zeile weiterschreiben, indem man das erste Koordinatenpaar wegläßt und direkt nach dem „LINE" das „TO" schreibt. Dann wird als Startpunkt für die erste Linie der zuletzt gezeichnete Punkt genommen. Dieser kann von allen Zeichenbefehlen aus ABU oder von allen AS-Zeichenbefehlen stammen.

& REC <x1>,<y1>,<x2>,<y2>:
REC zeichnet ein Rechteck. Die beiden Koordinatenpaare geben zwei diagonal gegenüberliegende Ecken des Rechtecks an. Welche Ecken angegeben werden und in welcher Reihenfolge, ist gleichgültig. Auch dieser Befehl kennt die Zeichenfarbe INVERS.

& BOX <x1>,<y1>,<x2>,<y2>:
BOX zeichnet ein ausgefülltes Rechteck, dessen Ecken durch die angegebenen Koordinaten bestimmt werden. Bei der Zeichenfarbe INVERS wird der angegebene Bereich invertiert. Damit ist es im Gegensatz zum Befehl CLS möglich, Teile des Bildschirms zu invertieren.

& CIRCLE <x>,<y>,<x-radius> (, (<y-radius>) (,<step>)):
CIRCLE zeichnet eine Ellipse um den Mittelpunkt, der durch das erste Koordinatenpaar bestimmt wird. Parameter 3 und 4 bestimmen über den X- und Y-Radius, wie groß die Ellipse wird. Diese beiden Parameter können im Bereich zwischen 0 und 300 liegen. Teile der Ellipse, die über den Bildschirmrand hinausgehen, werden nicht gezeichnet. Als fünfter Parameter kann ein Stepwert bzw. eine Schrittweite angegeben werden. Je größer dieser Wert ist, um so weniger Punkte des Keises werden berechnet, d.h. der Kreis wird ekkiger. Ein Stepwert von Null wird automatisch zu Eins. Der größte Stepwert ist 255; allerdings erscheint dann auf dem Bildschirm eine Raute. Fehlt der Stepwert, wird er automatisch auf Eins gesetzt. Auch der Y-Radius kann wegfallen. Dann werden X- und Y-Radius als gleich angesehen, und es erscheint ein Kreis auf dem Bildschirm. Soll der Y-Radius entfallen, aber trotzdem ein Stepwert angegeben werden, werden zwei Kommata hintereinander gesetzt (Bsp.: & CIRCLE 139,96,50,,10). Auch bei diesem Befehl kann mit der Farbe INVERS gezeichnet werden, dann treten allerdings Ungenauigkeiten auf: Die Ellipse wird dann an den vier Punkten, die die X- und Y-Achse schneiden, nicht geschlossen. Das liegt daran, daß die Ellipse sich aus vier Teilen zusammensetzt, die an jenen vier Punkten zusammenstoßen. Dadurch werden die Punkte doppelt gezeichnet, was bei der Farbe INVERS dazu führt, daß sie wieder gelöscht werden.

& FILL <x>,<y>:
FILL füllt eine geschlossene Fläche mit der gewünschten Zeichenfarbe aus. Die angegebenen Koordinatenpunkte gelten dabei als Startpunkt. Ist die Fläche nicht geschlossen, läuft der gesamte Bildschirm voll. Mit INVERS wird nicht gefüllt, die Farbe wird automatisch zu Weiß. Bei allen anderen Farben außer Schwarz und Weiß kann es zu Problemen kommen. Farbige Linien sind auf einem monochromen Monitor in Richtung der X-Achse gestrichelt. Daher sind sie auch in der HGR-Bitmap des Apple, nach der sich die Fill-Routine richtet, nicht geschlossen. Die Routine erkennt deshalb farbige Linien nicht als Begrenzung an.

& TEST <x>,<y>,<varname>:
TEST fragt ab, ob ein Punkt mit den angegebenen Koordinaten gesetzt ist oder nicht. Entsprechend wird in der Variablen, die als dritter Parameter angegeben werden muß, eine Eins (Punkt auf Koordinaten gesetzt) oder eine Null (...nicht gesetzt) abgelegt. Diese Variable muß numerisch sein, kann also auch vom Typ Integer sein. Auch hier richtet sich die Test-Routine nach der Bitmap, in der nicht jeder farbige Punkt gesetzt wird (siehe unter FILL). Es kann also passieren, daß man einen Punkt setzt und die Test-Routine trotzdem den Wert Null zurück gibt.

& INPUT ("Text";) <varname>:
INPUT gibt an der momentanen Cursorposition den Text aus, der dem INPUT-Befehl in Anführungsstrichen folgt, und erwartet dann vom Benutzer eine Eingabe. Ist die Variable hinter dem Semikolon eine Stringvariable, so erwartet die Input-Routine einen beliebigen Text vom Benutzer. Dieser Text darf sämtliche Zeichen enthalten, die mit der Tastatur einzugeben sind, außer Ctrl-C und Return. Return schließt die Eingabe ab. Wurde als erstes Zeichen ein Ctrl-C eingegeben, so bricht das Programm mit „BREAK IN ..." ab. Wird als erstes Zeichen Return gedrückt, so wird die Variable nicht verändert, d.h. der alte Wert bleibt erhalten. Ist die Variable hinter dem Semikolon eine numerische Variable, kann der Benutzer einen beliebigen numerischen Ausdruck eingeben. Dieser Ausdruck wird automatisch interpretiert und das Ergebnis in der Variable abgelegt. Ctrl-C und Return wirken wie bei der Eingabe von Texten. Folgt der Variablenname direkt dem INPUT, wird der Text und das Semikolon weggelassen. In diesem Fall wird ein Fragezeichen ausgegeben.

& ERR:
ERR verhindert ein Überlaufen des Stacks nach dem Auftreten von Fehlern im BASIC-Programm. Arbeitet man mit „ONERR GOTO ..." und springt von der Fehlerbehandlung wieder ins Programm zurück, so wird nach einiger Zeit das Programm mit einer Fehlermeldung unterbrochen, wenn zuviele Fehler auftreten. Um dies zu verhindern, sollte man als erstes in der Fehlerbehandlungs-Routine mit & ERR dieses Überlaufen des Stacks verhindern. Dadurch können beliebig viele Fehler auftreten, ohne daß der BASIC-Interpreter aussteigt.

& GOTO <zeilennummer>:
GOTO entspricht dem gleichnamigen AS-Befehl, die Zeilennummer kann hier aber ein beliebiger numerischer Ausdruck sein. Dieser wird interpretiert und das Ergebnis als die anzuspringende Zeilennummer angesehen. Existiert diese nicht, so erfolgt auch hier die Fehlermeldung „UNDEF'D STATEMENT ERROR IN ...".

& GOSUB <zeilennummer>:
GOSUB entspricht dem gleichnamigen AS-Befehl, für die Zeilennummer gilt das unter GOTO Gesagte.

& RESTORE <zeilennummer>:
RESTORE setzt den Datazeiger auf die angegebene Zeile. Befindet sich in dieser Zeile kein DATA-Statement, so wird automatisch das nächste gesucht und der Datazeiger auf die gefundene Zeile gesetzt. Gibt es ab dieser Zeile kein DATA-Statement mehr, wird beim nächsten Leseversuch durch READ eine Fehlermeldung ausgegeben. Für die Zeilennummer gilt das gleiche wie bei GOTO.

3. Das Programm

Das Programm wird durch BRUN ABU oder – wenn es sich bereits im Speicher befindet – durch CALL 32768 (= $8000) gestartet, und die Variablen werden auf ihre Startwerte gesetzt. Da die Routinen über das Ampersand-Zeichen „&" aufgerufen werden, muß auch der Ampersand-Vektor in Page 3 auf die Auswert-Routine

von ABU gelegt werden. (Zur Auswert-Routine siehe AGE-Programm von Klaus Schäfer, Peeker 1/86, S. 30ff.) Auch HI-MEM wird von ABU neu gesetzt, um zu verhindern, daß ABU durch Variablen überschrieben wird. Aus diesem Grund sollte ABU gestartet werden, bevor irgendwelche Variablen benutzt oder dimensioniert werden. Danach gibt es die Kontrolle wieder an den BASIC-Interpreter.

Jedesmal, wenn das Ampersand-Zeichen im BASIC-Programm auftaucht, verzweigt der Interpreter zur Auswert-Routine von ABU. Diese vergleicht den Namen hinter dem Ampersand-Zeichen mit den Namen in der Tabelle. Stimmt er überein, so wird die Startadresse dieser Routine aus der zweiten Tabelle geholt, auf dem Stack abgelegt und dann durch ein RTS angesprungen. Wird der Name nicht gefunden, so wird ein „SYNTAX ERROR IN ..." ausgegeben.
Die einzelnen Routinen, die angesprungen werden können, werden im folgenden beschrieben:

PAGE:
PAGE holt mittels der ROM-Routine GETBYTE die gewünschte Pagenummer. Nach einem Quantity-Check, d.h. einer Bereichsprüfung, wird die Nummer in der Variablen PAGENUM abgelegt. Gleichzeitig wird der entsprechende Wert in der Zeropage-Variablen HPAG abgelegt. Dieser Wert wird aus einer Tabelle geholt, die im Anschluß an dieser Routine im Speicher steht.
HIRES:
HIRES betätigt nur die Softswitches in der $C0-Page. Dabei wird entsprechend der Variablen PAGENUM entweder HGR-Seite 1 oder 2 eingeschaltet.
FULL:
FULL betätigt nur den Softswitch $C052, der auf die gesamte Grafikseite umschaltet. Dieser ist allerdings nur bei eingeschalteter Grafik sinnvoll.
MIX:
MIX betätigt analog zu FULL nur den Softswitch $C053, der den Befehl FULL wieder rückgängig macht.
BACKCOL:
BACKCOL liest die folgende Zahl mittels GETBYTE ein und setzt nach einem Quantity-Check die Variable BCKCOLOR auf diesen Wert.
CLS:
CLS überprüft die Variable BCKCOLOR. Liegt ihr Wert zwischen null und sieben, wird eine ROM-Routine angesprungen, die den Bildschirm mit dieser Farbe füllt. Bei acht wird zu einer eigenen Routine gesprungen, die den Bildschirm invertiert. Wer statt der bitmäßigen Invertierung die farbmäßige vorzieht, muß das EOR #$7F in ein EOR #$80 umwandeln.
COL:
COL liest den Farbwert mittels GETBYTE ein und setzt nach einem Quantity-Check zwei Variablen. In COLOR wird die Farbnummer abgelegt. COLOR1 ist ein Flag, in dem gespeichert wird, ob die Farbe INVERS ausgewählt wurde. Anschließend wird die Farbe durch die ROM-Routine SETHCOL gesetzt.

PSET:
PSET liest die Koordinaten mittels der ROM-Routine GETKOORD ein. Nach Überprüfung der Variablen COLOR1 wird entweder die ROM-Routine HPLOT oder die eigene Routine XPLOT aufgerufen.

LINE:
LINE überprüft zunächst das nächste Zeichen im BASIC-Text. Ist es das „TO"-Token, so wird der erste Teil der Line-Routine übersprungen. Ansonsten wird das erste Koordinatenpaar geholt und der Grafikcursor auf die entsprechende Stelle gesetzt. Dann wird ein Syntax-Check auf das TO-Token durchgeführt und das nächste Koordinatenpaar geholt. Entsprechend der Variablen COLOR1 wird entweder die ROM-Routine HPLOTTO angesprungen, die die Linie zeichnet, oder die eigene Routine XPLOTTO. Diese Fallunterscheidung wird auch in allen anderen Zeichen-Routinen vorgenommen, sie wird aber nicht mehr beschrieben. Nun wird überprüft, ob noch ein weiteres Koordinatenpaar folgt. Ist dies der Fall, wird der letzte Schritt wiederholt, sonst wird die Kontrolle wieder an den BASIC-Interpreter zurückgegeben.
REC:
REC holt beide Koordinaten mittels GETKOORD und verbindet diese anschließend zu einem Rechteck.
BOX:
BOX holt beide Koordinatenpaare mittels GETKOORD und sortiert die beiden Y-Werte der Größe nach. Anschließend werden in einer Schleife sämtliche Y-Werte zwischen diesen beiden Koordinaten durchlaufen, und es wird jeweils von der einen X-Position zur anderen eine Linie gezogen.
CIRCLE:
CIRCLE holt sich zuerst alle Parameter, die übergeben werden. Dabei wird auch beachtet, daß zwei Parameter entfallen können. Die Circle-Routine holt sich in einer Schleife, in der die Winkel zwischen 0 und 90 durchlaufen werden, die Sinus- und Cosinus-Werte aus einer Tabelle und multipliziert sie mit dem Radius. Auf diese Weise wird ein Viertelkreis berechnet, die drei anderen Viertelkreise werden gespiegelt. Die Tabelle enthält die Cosinuswerte für die Winkel zwischen 0 und 90 Grad. Diese Werte wurden mit 128 multipliziert und die Nachkommastellen abgestrichen. Für die Sinuswerte wird auf die Tabelle von hinten zugegriffen.
TEST:
TEST holt sich die Koordinaten mittels GETBYTE, setzt den Grafikcursor auf die angegebene Koordinate und überprüft das entsprechende Bit. Anschließend wird der Variablenname ausgewertet und der Wert des Bits in dieser abgespeichert.
FILL:
FILL durchläuft alle X-Positionen in einer Reihe bis zu einer Begrenzungslinie. Dabei merkt sich die Routine alle noch nicht ausgefüllten Flächen ober- und unterhalb dieser Reihe. Gibt es keine freien Flächen mehr, so springt die Routine wieder zum BASIC-Interpreter.
INPUT:
INPUT testet das nächste Zeichen im BASIC-Text. Ist es ein Anführungszeichen, wird der Text in den Anführungszeichen mittels ROM-Routinen ausgegeben, ansonsten wird ein Fragezeichen ausgegeben. Anschließend wird zur ROM-Routine INLINE gesprungen, die alle Zeichen annimmt. Nun wird der Variablenname und der eingegebene String ausgewertet und das Ergebnis in der angegebenen numerischen oder String-Variablen abgelegt.
ERR:
ERR initialisiert nur den Stack neu, so daß kein Überlauf auftreten kann. Dabei muß allerdings darauf geachtet werden, daß nicht die letzte Rücksprungadresse vom Stack genommen wird, da sonst nicht zum BASIC-Interpreter zurückgesprungen werden kann.
GOTO:
GOTO holt die Zeilennummer mittels FRMEVL, legt das Integer-Ergebnis in $50 und $51 ab und springt anschließend zum regulären GOTO-Befehl.
GOSUB:
GOSUB simuliert den normalen GOSUB-Befehl, d.h. die Ablage der Rücksprungadresse und des Gosub-Tokens auf dem Stack, holt die Zeilennummer mittels FRMEVL und springt anschließend zum normalen GOTO-Befehl.
RESTORE:
RESTORE holt die Zeilennummer mittels FRMEVL, sucht die Zeilennummer im Speicher und legt den gefundenen Zeiger in dem Datazeiger in der Zeropage ab. Steht dieser Zeiger allerdings nicht auf einem Data-Statement, so wird der nächste Doppelpunkt oder das Ende der Zeile gesucht und der Zeiger darauf gesetzt. Der READ-Befehl sucht selber nach dem nächsten DATA-Statement, nur muß der Datazeiger dazu auf einem Doppelpunkt oder auf dem Ende einer Zeile stehen.

Kurzhinweise

1. Zweck:
Befehlserweiterung des Applesoft-BASIC durch Ampersand-Befehle.
2. Konfiguration:
Apple II+/IIe/IIc; DOS 3.3 (kein ProDOS wegen HIMEM-Änderung).
3. Test:
RUN ABU.DEMO
4. Sammeldisk:
ABU.DEMO
T.ABU (Big-Mac-Quelltext, im Heft nicht gelistet)
ABU (Objektcode)
5. Sonstiges:
Beachten Sie, daß Sie ABU.DEMO oder ähnliche Programme, die mit &GOTO oder &GOSUB arbeiten, nicht mit normalen Renumber-Programmen umnumerieren können.

ABU.DEMO

```
100 PRINT CHR$ (4)"BRUN ABU"
105 PAUSE = 545
110 & BACKCOL0
115 & CLS: & HIRES: & COL8
120 FOR I = 1 TO 50
125 X1 = INT ( RND (1) * 280)
130 Y1 = INT ( RND (1) * 192)
135 X2 = INT ( RND (1) * 50)
140 Y2 = INT ( RND (1) * 50)
145 XS = SGN (X1 - 55)
150 YS = SGN (Y1 - 55)
155 X2 = X1 - X2 * XS
160 Y2 = Y1 - Y2 * YS
165 & BOXX1,Y1,X2,Y2
170 FOR O = 1 TO 6
175 & RECX1,Y1,X2,Y2
180 NEXT O,I
185 & GOSUB PAUSE
190 & BACKCOL8
195 FOR I = 1 TO 5
200 & CLS
205 NEXT I
210 & GOSUB PAUSE
215 & BACKCOL0: & COL3
220 & CLS
225 D = 3.141 / 180
230 FOR I = 0 TO 360 STEP 10
235 & CIRCLE50 + I / 2,96 -
    SIN (D * I) * 46,49,,10
240 NEXT I
245 & GOSUB PAUSE
250 & CLS
255 FOR I = 0 TO 30
260 X = INT ( RND (1) * 219) + 30
265 Y = INT ( RND (1) * 130) + 30
270 XR = INT ( RND (1) * 30) + 1
275 YR = INT ( RND (1) * 30) + 1
280 ST = INT ( RND (1) * 22)
285 & CIRCLEX,Y,XR,YR,ST
290 NEXT I
295 & GOSUB PAUSE
300 & COL8
305 FOR I = 0 TO 50
310 X1 = INT ( RND (1) * 280)
315 X2 = INT ( RND (1) * 280)
320 Y1 = INT ( RND (1) * 192)
325 Y2 = INT ( RND (1) * 192)
330 & LINEX1,Y1 TO X2,Y2
335 FOR WA = 0 TO 50: NEXT WA
340 & LINEX1,Y1 TO X2,Y2
345 NEXT I
350 & GOSUB PAUSE
355 & CLS: & COL3
360 FOR I = 45 TO 95 STEP 10
365 & CIRCLE139,96,I,,3
370 NEXT I
375 & LINE139,1 TO 139,191
380 & LINE44,96 TO 234,96
385 FOR I = 1 TO 3
390 & FILL167 + I * 20,98
395 & FILL112 - I * 20,94
400 IF I = 3 THEN 415
405 & FILL141,59 - I * 20
410 & FILL137,133 + I * 20
415 NEXT I
420 & GOSUB PAUSE
425 & BACKCOL3: & COL0
430 & CLS
435 FOR X = 139 TO 0 STEP - 4
440 & LINE139,191 TO X,0
445 & LINE140,191 TO 279 - X,0
450 NEXT X
455 FOR Y = 0 TO 188 STEP 4
460 & LINE140,191 TO 279,Y
465 & LINE139,191 TO 0,Y
470 NEXT Y
475 & GOSUB PAUSE
480 TEXT
485 HOME
490 PRINT "Geben Sie einen num.
    Ausdruck ein:"
495 & INPUT "ABU >";AU
500 PRINT : PRINT "Das Ergebnis
    lautet: ";AU
505 & GOSUB PAUSE
510 HOME
515 PRINT "Geben Sie einen bel.
    Text ein:"
520 & INPUT "ABU >";TE$
525 PRINT : PRINT "Sie gaben ein: ";
    TE$
530 & GOSUB PAUSE
535 & HELP
540 END
545 CALL - 198: REM Pause-Routine
550 FOR I = 1 TO 1000
555 IF PEEK ( - 16384) > 127
    THEN GET G$: GOTO 565
560 NEXT I
565 RETURN
```

Peeker-Sammeldisk #25

DOS-3.3-Diskette; Heft 1/1987
Einzelbezug DM 28,–
Fortsetzungsbezug DM 20,–
(1) Zweck; (2) Heft/Seite; (3) Gerätekonfiguration; (4) Betriebssystem; (5) Programmstart; (6) Sonstiges

A 025 TABLE.EDITOR
A 028 SHAPE.EDITOR
A 027 SHAPE.CREATE
B 002 ST. EDIT
A 003 UEBERSICHT
(1) Shape-Entwicklungspaket; (2) 1/87, S. 6; (3) Apple II+/e/c; (4) DOS 3.3 oder ProDOS; (5) RUN UEBERSICHT oder direkt RUN TABLE.EDITOR usw.; (6) Als Shapes zum Ausprobieren des Programms kann man neben ST.EDIT auch die Mousory-Shapes von Sammeldisk #19 verwenden.

A 011 TRACE.KORREKTUR
(1) BASIC.SYSTEM-Patch-Programm zur Beseitigung des Trace-Bugs; (2) 1/87, S. 20; (3) Apple II+/e/c; (4) BASIC.SYSTEM 1.0 oder 1.1; (5) siehe Aufsatz; (6) TRACE. KORREKTUR muß zunächst mit CONVERT oder DOSTOPRO von der Sammeldisk auf Ihre ProDOS-Arbeitsdisk konvertiert werden.

A 017 PROGRAMMVERWALT
A 005 ALPHASORT
T 002 PROGRAMMDATEI
(1) Verwaltung von Programmen bzw. Programmnamen; (2) 1/87, S. 24; (3) Apple II+/e/c in 40 Z/Z; (4) DOS 3.3 oder ProDOS; (5) RUN PROGRAMMVERWALT

A 007 ABU.DEMO
T 074 T.ABU
B 013 ABU
(1) Ampersand-Befehlserweitungen (&Circle, &Box, &Goto, &Restore usw.); (2) 1/87, S. 30; (3) Apple II+/e/c; (4) DOS 3.3 (kein ProDOS!); (5) RUN ABU.DEMO

T 007 CAT.P
T 060 CAT.I
(1) Catalog-Befehle für Kyan-Psacal; (2) 1/87, S. 36; (3) Apple II+/e/c; (4) Kyan-Pascal ab Version 2.0; (5) CAT von Kix-Menü; (6) CAT.P und CAT.I müssen zunächst mit CONVERT oder DOSTOPRO auf Ihre Kyan-Arbeitsdisk konvertiert werden. Danach CAT.P compilieren.

T 018 DISM1.TEXT
T 069 DISM2.TEXT
T 078 DISM3.TEXT
T 046 DISM4.TEXT
(1) P-Code-Disassembler für UCSD; (2) 10/85, S. 44-46 (Beschreibung aller 3 Utilities); (3) Apple II+/e/c; (4) Apple-Pascal 1.1 oder 1.2; (5) Die drei großen Pascal-Utilities sind damit auf die Sammeldisketten #22 (Disk-Editor), #24 (Library-Programm) und #25 (Disassembler) verteilt. Ferner wird das UTILITY-Programm von Sammeldisk #23 benötigt.

Hüthig Software Service
Postfach 10 28 69
6900 Heidelberg 1

BSAVE ABU, A$8000, L$0BA8

```
$8000: A9 00 A0 80 85 73 84 74
$8008: A9 4C A0 31 A2 80 8D F5
$8010: 03 8C F6 03 8E F7 03 A9
$8018: 01 8D 01 8F A9 20 85 E6
$8020: A2 03 8E 02 8F 20 EC F6
$8028: A9 00 8D 04 8F 8D 03 8F
$8030: 60 A2 FF A9 00 8D 00 8F
$8038: A0 FF E8 C8 BD 6F 80 C9
$8040: 20 F0 16 C9 FF F0 23 D1
$8048: B8 F0 EF E8 BD 6F 80 C9
$8050: 20 D0 F8 EE 00 8F 4C 38
$8058: 80 AD 00 8F 0A AA BD C9
$8060: 80 48 BD C8 80 48 20 98
$8068: D9 60 A2 10 4C 12 D4 50
$8070: 41 47 45 20 48 49 52 45
$8078: 53 20 46 55 4C 4C 20 4D
$8080: 49 58 20 42 41 43 4B 43
$8088: 4F 4C 20 43 4C 53 20 43
$8090: 4F 4C 20 50 53 45 54 20
$8098: 4C 49 4E 45 20 52 45 43
$80A0: 20 42 4F 58 20 43 49 52
$80A8: 43 4C 45 20 54 45 53 54
$80B0: 20 46 49 4C 4C 20 AB 20
$80B8: B0 20 AE 20 84 20 45 52
$80C0: 52 20 48 45 4C 50 20 FF
$80C8: EF 80 0A 81 1A 81 1E 81
$80D0: 22 81 32 81 65 81 8F 81
$80D8: A7 81 51 82 11 83 71 83
$80E0: 53 87 BA 87 41 89 4A 89
$80E8: 67 89 AA 89 2A 8A 34 8A
$80F0: 20 F8 E6 E0 01 B0 05 A2
$80F8: 35 20 12 D4 E0 03 B0 F7
$8100: 8E 01 8F BD 08 81 85 E6
$8108: 60 20 40 AD 50 C0 AD 57
$8110: C0 AD 52 C0 AE 01 8F BD
$8118: 53 C0 60 2C 52 C0 60 2C
$8120: 53 C0 60 20 F8 E6 E0 09
$8128: 90 05 A2 35 20 12 D4 8E
$8130: 04 8F 60 AE 04 8F E0 08
$8138: F0 11 20 EC F6 A5 E4 85
$8140: 1C 20 F6 F3 AE 02 8F 20
$8148: EC F6 60 A9 00 85 06 A5
$8150: E6 85 07 A2 20 A0 00 B1
$8158: 06 49 7F 91 06 C8 D0 F7
$8160: E6 07 CA D0 F0 60 20 F8
$8168: E6 E0 09 90 05 A2 35 4C
$8170: 12 D4 E0 08 F0 0C 8E 02
$8178: 8F 20 EC F6 A9 00 8D 03
$8180: 8F 60 A2 03 8E 02 8F 20
$8188: EC F6 A9 80 8D 03 8F 60
$8190: 20 B9 F6 2C 03 8F 30 04
$8198: 20 57 F4 60 20 11 F4 A5
$81A0: 30 29 7F 51 26 91 26 60
$81A8: 20 B7 00 C9 C1 F0 06 20
$81B0: B9 F6 20 11 F4 A9 C1 20
$81B8: C0 DE 20 B9 F6 8C 06 8F
$81C0: A8 8A AE 06 8F 2C 03 8F
$81C8: 30 0B 20 3A F5 20 B7 00
$81D0: C9 C1 F0 E1 60 20 DB 81
$81D8: 4C CD 81 48 38 E5 E0 48
$81E0: 8A E5 E1 85 D3 B0 0A 68
$81E8: 49 FF 69 01 48 A9 00 E5
$81F0: D3 85 D1 85 D5 68 85 D0
$81F8: 85 D4 68 85 E0 86 E1 98
$8200: 18 E5 E2 90 04 49 FF 69
$8208: FE 85 D2 84 E2 66 D3 38
$8210: E5 D0 AA A9 FF E5 D1 85
$8218: 1D A4 E5 B0 05 0A 20 65
$8220: F4 38 A5 D4 65 D2 85 D4
$8228: A5 D5 E9 00 85 D5 A5 30
$8230: 29 7F 51 26 91 26 E8 D0
$8238: 04 E6 1D F0 14 A5 D3 B0
$8240: DC 20 D3 F4 18 A5 D4 65
$8248: D0 85 D4 A5 D5 65 D1 50
$8250: DB 60 20 B9 F6 8E 07 8F
$8258: 8C 08 8F 8D 09 8F A9 2C
$8260: 20 C0 DE 20 B9 F6 8E 0A
$8268: 8F 8C 0B 8F 8D 0C 8F 2C
$8270: 03 8F 30 3D AE 07 8F AC
$8278: 08 8F AD 09 8F 20 11 F4
$8280: AD 07 8F AE 08 8F AC 0C
$8288: 8F 20 3A F5 AD 0A 8F AE
$8290: 0B 8F AC 0C 8F 20 3A F5
$8298: AD 0A 8F AE 0B 8F AC 09
$82A0: 8F 20 3A F5 AD 07 8F AE
$82A8: 08 8F AC 09 8F 20 3A F5
$82B0: 60 AE 07 8F AC 08 8F AD
$82B8: 09 8F 20 9C 81 AD 07 8F
$82C0: AE 08 8F AC 0C 8F 20 DB
$82C8: 81 AE 07 8F AC 08 8F AD
$82D0: 0C 8F 20 9C 81 AD 0A 8F
$82D8: AE 0B 8F AC 0C 8F 20 DB
$82E0: 81 AE 0A 8F AC 0B 8F AD
$82E8: 0C 8F 20 9C 81 AD 0A 8F
$82F0: AE 0B 8F AC 09 8F 20 DB
$82F8: 81 AE 0A 8F AC 0B 8F AD
$8300: 09 8F 20 9C 81 AD 07 8F
$8308: AE 08 8F AC 09 8F 20 DB
$8310: 81 60 20 B9 F6 8E 07 8F
$8318: 8C 08 8F 8D 09 8F A9 2C
$8320: 20 C0 DE 20 B9 F6 8E 0A
$8328: 8F 8C 0B 8F 8D 0C 8F CD
$8330: 09 8F B0 0F 8D 06 8F AD
$8338: 09 8F 8D 0C 8F AD 06 8F
$8340: 8D 09 8F AE 07 8F AC 08
$8348: 8F AD 09 8F 20 11 F4 AD
$8350: 0A 8F AE 0B 8F AC 09 8F
$8358: 2C 03 8F 30 06 20 3A F5
$8360: 4C 66 83 20 DB 81 EE 09
$8368: 8F AD 0C 8F CD 09 8F B0
$8370: D2 60 20 B9 F6 8E 28 8F
$8378: 8C 29 8F 8D 2A 8F A9 01
$8380: 8D 05 8F A9 2C 20 C0 DE
$8388: 20 67 DD 20 52 E7 8C 12
$8390: 8F 8D 13 8F 8C 14 8F 8D
$8398: 15 8F C9 01 90 0D C9 02
$83A0: B0 04 C0 2D 90 05 A2 35
$83A8: 4C 12 D4 20 B7 00 C9 2C
$83B0: D0 38 20 B1 00 C9 2C F0
$83B8: 24 20 67 DD 20 52 E7 8C
$83C0: 14 8F 8D 15 8F C9 01 90
$83C8: 0D C9 02 B0 04 C0 2D 90
$83D0: 05 A2 35 4C 12 D4 20 B7
$83D8: 00 C9 2C D0 0D 20 B1 00
$83E0: 20 F8 E6 E0 00 F0 03 8E
$83E8: 05 8F 20 EE 83 60 A9 FF
$83F0: 8D 16 8F 8D 17 8F 8D 18
$83F8: 8F 8D 19 8F EE 14 8F D0
$8400: 03 EE 15 8F EE 12 8F D0
$8408: 03 EE 13 8F A9 00 8D 11
$8410: 8F 20 29 84 AD 11 8F 18
$8418: 6D 05 8F 8D 11 8F 90 F1
$8420: A9 FF 8D 11 8F 20 29 84
$8428: 60 AD 16 8F 8D 1A 8F AD
$8430: 17 8F 8D 1B 8F AD 18 8F
$8438: 8D 1C 8F AD 19 8F 8D 1D
$8440: 8F A9 00 8D 16 8F 8D 17
$8448: 8F 8D 18 8F 8D 19 8F AC
$8450: 11 8F B9 54 86 85 AD A9
$8458: 00 85 AE AD 12 8F 85 64
$8460: AD 13 8F 85 65 20 B8 E2
$8468: 8E 1E 8F 8D 1F 8F 4E 1F
$8470: 8F 6E 1E 8F 4E 1F 8F 6E
$8478: 1E 8F 4E 1F 8F 6E 1E 8F
$8480: 4E 1F 8F 6E 1E 8F 4E 1F
$8488: 8F 6E 1E 8F 4E 1F 8F 6E
$8490: 1E 8F 4E 1F 8F 6E 1E 8F
$8498: A9 FF 38 ED 11 8F A8 B9
$84A0: 54 86 85 AD A9 00 85 AE
$84A8: AD 14 8F 85 64 AD 15 8F
$84B0: 85 65 20 B8 E2 8E 20 8F
$84B8: 8D 21 8F 4E 21 8F 6E 20
$84C0: 8F 4E 21 8F 6E 20 8F 4E
$84C8: 21 8F 6E 20 8F 4E 21 8F
$84D0: 6E 20 8F 4E 21 8F 6E 20
$84D8: 8F 4E 21 8F 6E 20 8F 4E
$84E0: 21 8F 6E 20 8F 18 AD 1E
$84E8: 8F 6D 28 8F 8D 07 8F AD
$84F0: 1F 8F 6D 29 8F 8D 08 8F
$84F8: 38 AD 28 8F ED 1E 8F 8D
$8500: 0A 8F AD 29 8F ED 1F 8F
$8508: 8D 0B 8F 18 AD 20 8F 6D
$8510: 2A 8F 8D 0D 8F AD 21 8F
$8518: 69 00 8D 0E 8F 38 AD 2A
$8520: 8F ED 20 8F 8D 0F 8F A9
$8528: 00 ED 21 8F 8D 10 8F AD
$8530: 08 8F C9 01 90 10 C9 02
$8538: B0 07 AD 07 8F C9 18 90
$8540: 05 A9 FF 8D 16 8F AD 0B
$8548: 8F C9 80 90 05 A9 FF 8D
$8550: 17 8F AD 0E 8F D0 07 AD
$8558: 0D 8F C9 C0 90 05 A9 FF
$8560: 8D 18 8F AD 10 8F C9 80
$8568: 90 05 A9 FF 8D 19 8F AD
$8570: 16 8F D0 49 AD 1A 8F D0
$8578: 44 AD 18 8F D0 1D AD 1C
$8580: 8F D0 18 AE 07 8F AC 08
$8588: 8F AD 0D 8F 20 3A 86 AD
$8590: 22 8F AE 23 8F AC 26 8F
$8598: 20 47 86 AD 19 8F D0 49
$85A0: AD 1D 8F D0 44 AE 07 8F
$85A8: AC 08 8F AD 0F 8F 20 3A
$85B0: 86 AD 22 8F AE 23 8F AC
$85B8: 27 8F 20 47 86 AD 17 8F
$85C0: D0 53 AD 1B 8F D0 4E AD
$85C8: 19 8F D0 1D AD 1D 8F D0
$85D0: 18 AE 0A 8F AC 0B 8F AD
$85D8: 0F 8F 20 3A 86 AD 24 8F
$85E0: AE 25 8F AC 27 8F 20 47
$85E8: 86 AD 17 8F D0 27 AD 1B
$85F0: 8F D0 22 AD 18 8F D0 1D
$85F8: AD 1C 8F D0 18 AE 0A 8F
$8600: AC 0B 8F AD 0D 8F 20 3A
$8608: 86 AD 24 8F AE 25 8F AC
$8610: 26 8F 20 47 86 AD 07 8F
$8618: 8D 22 8F AD 08 8F 8D 23
$8620: 8F AD 0D 8F 8D 26 8F AD
$8628: 0A 8F 8D 24 8F AD 0B 8F
$8630: 8D 25 8F AD 0F 8F 8D 27
$8638: 8F 60 2C 03 8F 30 04 20
$8640: 11 F4 60 20 9C 81 60 2C
$8648: 03 8F 30 04 20 3A F5 60
$8650: 20 DB 81 60 7F 7F 7F 7F
$8658: 7F 7F 7F 7F 7F 7F 7F 7F
$8660: 7F 7F 7F 7F 7F 7F 7F 7F
$8668: 7F 7E 7E 7E 7E 7E 7E 7E
$8670: 7E 7D 7D 7D 7D 7D 7D 7D
$8678: 7C 7C 7C 7C 7C 7B 7B 7B
$8680: 7B 7B 7A 7A 7A 7A 79 79
$8688: 79 79 78 78 78 78 77 77
$8690: 77 77 76 76 76 75 75 75
$8698: 74 74 74 73 73 73 72 72
$86A0: 72 71 71 71 70 70 70 6F
$86A8: 6F 6E 6E 6E 6D 6D 6C 6C
$86B0: 6B 6B 6B 6A 6A 69 69 68
$86B8: 68 68 67 67 66 66 65 65
$86C0: 64 64 63 63 62 62 61 61
$86C8: 60 60 5F 5F 5E 5E 5D 5C
$86D0: 5C 5B 5B 5A 5A 59 59 58
$86D8: 57 57 56 56 55 55 54 53
$86E0: 53 52 52 51 50 50 4F 4F
$86E8: 4E 4D 4D 4C 4B 4B 4A 49
$86F0: 49 48 48 47 46 46 45 44
$86F8: 44 43 42 42 41 40 40 3F
$8700: 3E 3D 3D 3C 3B 3B 3A 39
$8708: 39 38 37 36 36 35 34 34
$8710: 33 32 31 31 30 2F 2F 2E
$8718: 2D 2C 2C 2B 2A 29 29 28
$8720: 27 26 26 25 24 23 23 22
$8728: 21 20 20 1F 1E 1D 1C 1C
$8730: 1B 1A 19 19 18 17 16 16
$8738: 15 14 13 12 12 11 10 0F
$8740: 0E 0E 0D 0C 0B 0B 0A 09
$8748: 08 07 07 06 05 04 03 03
$8750: 02 01 00 00 20 B9 F6 8E
$8758: 2B 8F 8C 2C 8F 8D 2D 8F
$8760: A9 2C 20 C0 DE A9 00 85
$8768: 14 20 E3 DF 85 85 84 86
$8770: 24 11 10 05 A2 A3 4C 12
$8778: D4 A5 0F F0 F7 20 A1 87
$8780: AC 37 8F 20 01 E3 A5 12
$8788: F0 0F 20 52 E7 A0 00 A5
$8790: 51 91 85 C8 A5 50 91 85
$8798: 60 A6 85 A4 86 20 2B EB
$87A0: 60 AE 2B 8F AC 2C 8F AD
$87A8: 2D 8F 20 11 F4 A5 30 29
$87B0: 7F 31 26 F0 02 A9 01 8D
$87B8: 37 8F 60 20 B9 F6 8E 34
$87C0: 8F 8C 35 8F 8D 36 8F A9
$87C8: 00 8D 2E 8F AD 34 8F 8D
$87D0: 2B 8F AD 35 8F 8D 2C 8F
$87D8: AD 36 8F 8D 2D 8F 20 A1
$87E0: 87 AD 37 8F 8D 2F 8F AD
$87E8: 36 8F C9 C0 90 03 4C 21
$87F0: 89 AD 34 8F 38 E9 01 8D
$87F8: 2B 8F AD 35 8F E9 00 8D
$8800: 2C 8F AD 36 8F 8D 2D 8F
$8808: 20 A1 87 AD 37 8F CD 2F
$8810: 8F D0 20 AD 35 8F C9 01
$8818: B0 05 AD 34 8F F0 14 AD
$8820: 34 8F 38 E9 01 8D 34 8F
$8828: AD 35 8F E9 00 8D 35 8F
$8830: 4C F1 87 A9 50 8D 30 8F
$8838: 8D 32 8F AD 30 8F 8D 31
$8840: 8F AD 34 8F 8D 2B 8F AD
$8848: 35 8F 8D 2C 8F AD 36 8F
$8850: 18 69 01 8D 2D 8F 20 A1
$8858: 87 AD 37 8F 8D 30 8F CD
$8860: 2F 8F D0 23 AD 31 8F CD
$8868: 2F 8F F0 1B AC 2E 8F AD
$8870: 34 8F 99 00 8C AD 35 8F
$8878: 99 00 8D AD 36 8F 18 69
$8880: 01 99 00 8E EE 2E 8F AD
$8888: 32 8F 8D 33 8F AD 34 8F
$8890: 8D 2B 8F AD 35 8F 8D 2C
$8898: 8F AD 36 8F 38 E9 01 8D
$88A0: 2D 8F 20 A1 87 AD 37 8F
$88A8: 8D 32 8F CD 2F 8F D0 23
$88B0: AD 33 8F CD 2F 8F F0 1B
$88B8: AC 2E 8F AD 34 8F 99 00
$88C0: 8C AD 35 8F 99 00 8D AD
$88C8: 36 8F 38 E9 01 99 00 8E
$88D0: EE 2E 8F AE 34 8F AC 35
$88D8: 8F AD 36 8F 20 57 F4 AD
$88E0: 34 8F 18 69 01 8D 2B 8F
$88E8: AD 35 8F 69 00 8D 2C 8F
$88F0: AD 36 8F 8D 2D 8F 20 A1
$88F8: 87 AD 37 8F CD 2F 8F D0
$8900: 20 AD 35 8F F0 07 AD 34
$8908: 8F C9 17 B0 14 AD 34 8F
$8910: 18 69 01 8D 34 8F AD 35
$8918: 8F 69 00 8D 35 8F 4C 3B
$8920: 88 AD 2E 8F F0 1B CE 2E
$8928: 8F AC 2E 8F B9 00 8C 8D
$8930: 34 8F B9 00 8D 8D 35 8F
$8938: B9 00 8E 8D 36 8F 4C E7
$8940: 87 60 20 67 DD 20 52 E7
$8948: 4C 41 D9 A9 03 20 D6 D3
$8950: A5 B9 48 A5 B8 48 A5 76
$8958: 48 A5 75 48 A9 B0 48 20
$8960: B7 00 20 42 89 4C D2 D7
$8968: 20 67 DD 20 52 E7 20 1A
$8970: D6 A5 9B 18 69 04 85 7D
$8978: A5 9C 69 00 85 7E A5 B8
$8980: 48 A5 B9 48 A5 7D 85 B8
$8988: A5 7E 85 B9 20 B7 00 F0
$8990: 13 C9 3A F0 0F C9 83 F0
$8998: 0B 20 95 D9 A5 B8 85 7D
$89A0: A5 B9 85 7E 68 85 B9 68
$89A8: 85 B8 60 20 06 E3 20 B7
$89B0: 00 C9 22 D0 12 20 81 DE
$89B8: 20 3D DB A9 3B 20 C0 DE
$89C0: A9 80 85 33 4C CB 89 A9
$89C8: BF 85 33 A9 55 85 52 20
$89D0: E3 DF 85 85 84 86 20 30
$89D8: D5 AD 00 02 C9 03 D0 03
$89E0: 20 63 D8 C9 00 D0 01 60
$89E8: 24 11 10 1C A9 00 A0 02
$89F0: A2 00 20 E9 E3 A5 9D 20
$89F8: D5 E3 86 71 84 72 A2 00
$8A00: A0 02 20 E2 E5 4C B3 DA
$8A08: A5 12 48 A5 B8 48 A5 B9
$8A10: 48 A9 00 85 B8 A9 02 85
$8A18: B9 20 59 D5 20 B1 00 20
$8A20: 67 DD 68 85 B9 68 85 B8
$8A28: 4C 62 DA 68 48 A6 DF
$8A30: 9A 48 98 48 60 20 58 FC
$8A38: A2 00 BD 57 8A F0 08 09
$8A40: 80 20 ED FD E8 D0 F3 A2
$8A48: 00 BD 4E 8B F0 08 09 80
$8A50: 20 ED FD E8 D0 F3 60 26
$8A58: 50 41 47 45 20 3C 31 2D
$8A60: 32 3E 8D 26 48 49 52 45
$8A68: 53 8D 26 46 55 4C 4C 8D
$8A70: 26 4D 49 58 8D 26 42 41
$8A78: 43 4B 43 4F 4C 20 3C 30
$8A80: 2D 38 3E 8D 26 43 4C 53
$8A88: 8D 26 43 4F 4C 20 3C 30
$8A90: 2D 38 3E 8D 26 50 53 45
$8A98: 54 20 3C 58 3E 2C 3C 59
$8AA0: 3E 8D 26 4C 49 4E 45 20
$8AA8: 5B 3C 58 31 3E 2C 3C 59
$8AB0: 31 3E 5D 20 54 4F 20 3C
$8AB8: 58 32 3E 2C 3C 59 32 3E
$8AC0: 20 5B 54 4F 20 3C 58 33
$8AC8: 3E 2C 3C 59 33 3E 5D 8D
$8AD0: 26 52 45 43 20 3C 58 31
$8AD8: 3E 2C 3C 59 31 3E 2C 3C
$8AE0: 58 32 3E 2C 3C 59 32 3E
$8AE8: 8D 26 42 4F 58 20 3C 58
$8AF0: 31 3E 2C 3C 59 31 3E 2C
$8AF8: 3C 58 32 3E 2C 3C 59 32
$8B00: 3E 8D 26 43 49 52 43 4C
$8B08: 45 20 3C 58 3E 2C 3C 59
$8B10: 3E 2C 3C 58 52 3E 5B 2C
$8B18: 5B 3C 59 52 3E 5D 5B 2C
$8B20: 3C 53 54 45 5D 3E 5D 5D
$8B28: 8D 26 46 49 4C 4C 20 3C
$8B30: 58 3E 2C 3C 59 3E 8D 26
$8B38: 54 45 53 54 20 3C 58 3E
$8B40: 2C 3C 59 3E 2C 56 41 52
$8B48: 4E 41 4D 45 8D 00 26 47
$8B50: 4F 54 4F 20 3C 41 55 53
$8B58: 44 52 55 43 4B 3E 8D 26
$8B60: 47 4F 53 55 42 20 3C 41
$8B68: 55 53 44 52 55 43 4B 3E
$8B70: 8D 26 52 45 53 54 4F 52
$8B78: 45 20 3C 41 55 53 44 52
$8B80: 55 43 4B 3E 8D 26 49 4E
$8B88: 50 55 54 20 5B 22 54 65
$8B90: 78 74 22 3B 5D 20 56 41
$8B98: 52 4E 41 4D 45 8D 26 45
$8BA0: 52 52 8D 00 00 00 00 00
```

BUCHHALTUNG mit BUCH

Doppelte Buchführung, die auf Selbständige und Kleinunternehmer zugeschnitten ist:

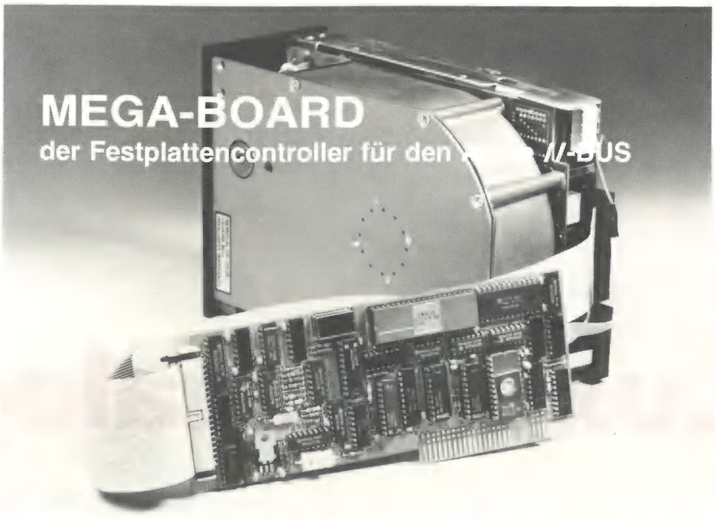
- 14 Tage frei zur Ansicht!
- einfache und sichere Benutzerführung
- Prüfsiegel »anwenderfreundliche Software«
- Betriebsübersicht, G&V Rechnung, Debitoren und Creditoren, Journal, Kontenblätter, Saldenliste
- automatische Umsatzsteuerverbuchung
- das Programm hat eine Lohnsteueraußenprüfung des Finanzamtes in unserem Hause bestanden

»Ein leistungsfähiges, leicht zu bedienendes Werkzeug...« (Testbericht PC-Soft, 7/85)
»...halten wir dieses Finanzbuchhaltungsprogramm für ganz ausgezeichnet.« (Softwaretestjahrbuch '85)

660,— DM für Apple II + IIe/IIc und IBM-PC/XT/AT, Commodore PC 10 etc.

Informationen bei

RÖNTGEN SOFTWARE
Simpert-Krämer-Str. 44,
8909 Edelstetten,
Tel. 08283/1463



Nicht jeder kann und möchte an seinen Apple //-Rechner unser MEGA-CORE oder die MDB 10/20 anschließen. Gerade für diesen Personenkreis bietet sich das MEGA-BOARD als der alternative und preiswerte Einstieg zum komfortablen Festplattenbetrieb an. Das Manual führt Sie zielsicher zu Ihren Wünschen:

- Festplattenbetrieb von 5-64 MBytes
- Betriebssystembereiche frei wählbar
- Booten von der Festplatte
- Betriebssysteme DOS, MS-DOS, CP/M, UCSD-Pascal, ProDOS menuegesteuert im Zugriff.

Zum Lieferumfang gehört der Controller MEGA-BOARD, alle Kabel mit Steckern, Installationssoftware, das ausführliche Manual und auf Wunsch die Festplatte zum Tagespreis.

Ein Produkt von:

FRANK & BRITTING
Elektronik Entwicklungs GmbH
Lange Straße 4, 7529 Forst
Telefon: 07251/103068-69
Telex: 7822452 fub d

Die Harddiskcontroller-Spezialisten

Erster aaa „little big" Ableger
für Apple II/+/e, Basis 108 und compatible

Sicher. – Wir haben uns bei der Entwicklung u. Fertigstellung unserer **Mega Ramcard II**® etwas Zeit gelassen, damit Sie jetzt auch Zeit gewinnen. Und das hat sich gelohnt!
Die Tests, aber auch die mitgelieferten Ramdisktreiber-Programme für folgende Betriebssysteme, beweisen es:

- Apple DOS 3.3
- Diversi DOS 2-C/4-C
- Apple ProDOS 1.0.1, 1.1.1
- Apple Pascal 1.1
  Apple Pascal 1.2/64 K, 1.2/128 K
  Apple Pascal 1.3/64 K, 1.3/128 K
- CP/M 2.2 für
  Microsoft Softcard, CP/M 2.20, 2.23
  Microsoft premium card, CP/M 2.26
  Microsoft Softcard II, CP/M 2.28

Sogar nach einem „Systemabsturz" oder Betriebssystemwechsel bleiben alle Dateien erhalten!
Peripherieslot und Speicherkapazität werden automatisch erkannt, sodaß das lästige Eingeben entfällt.
Mega Ramcard II® ist mit den erphi-AFDC2/AFDC3-Controller vollständig kompatibel, u.v.m.
Demo-Diskette = 4,50 (in Briefmarken) anfordern.
Händleranfragen schriftlich erwünscht.

Mega Ramcard II, 1 MByte bestückt + Software (s.o.) + deutsches Handbuch, II/+/e	878,–
Mega Ramcard II, 256 K-RAM best. (bis 1 MB aufr.) + Software (s.o.) + dt. Hb., II/+/e	498,–
Apple-Works Anpassung	69,–
Disk II-Siemens-Laufwerk im Geh. + Kabel f. orig. + comp. Contr. geeignet, II/+/e	298,–
Disk II-Siemens-Laufw. + Contr. + Kabel, II/+/e	359,–
Erphi-AFDC2-Controller + Autopatch Software + deutsches Handbuch, II/+/e	198,–
Disk II-Philips Laufwerk o. Geh., 2 x 80 Track-640 KB f. AFDC2 o. 3 modifiziert, II/+/e	298,–
Erphi-DuoDisk 1,2 MByte im Gehäuse + Erphi-AFDC3-Controller, II/+/e	898,–

REPARATUREN an Apple + Compatiblen Geräten + Zubehör führt unser Spezialteam garantiert zuverlässig + besonders kostengünstig aus.. **Sprechen Sie mit uns. Kostenvoranschlag auf Wunsch!**

aaa electronic gmbh
Telex 0772642 aaa-d
Habsburgerstraße 134
7800 FREIBURG, Tel. (0761) 276864
Bauelemente – Bausätze – µP's
Meßgeräte – Zubehör – Fachliteratur
Fachgeschäft für Elektronik + Mikrocomputer

Ausgabe und Eingabe mit TYPETERM®

im Slot Ihres

APPLE II/IIe/IIgs

Das bedeutet: Computertextverarbeitung von der Schreibmaschinentastatur! Steckerfertig ohne Umbau.

Die neue CE-550! mit TYPETERM DM 1.398,–

TYPETERM-Interface DM 479,–
für alle BROTHER-Typenradschreibmaschinen ab AX-30 bis EM-811 (auch für Vorgängermodelle!)
Paketpreis z. B.:

EM-501 mit TYPETERM	DM 2136,–
EM-511 mit TYPETERM	DM 2412,–
EM-701 mit TYPETERM	DM 2468,–

TYPETERM – ein starkes Interface für starke Maschinen! Alle Cursor- und Ctl-Befehle. 4k ROM auf der Karte für DOS, PRODOS, CP/M, PASCAL. 2 Zeichensätze verfügbar z. B. deutsch u. ASCII. Alle Features: Hoch-/Tiefstellen, autom. Unterstreichen, var. Zeichen und Zeilenabst., autom. Papierzuführung usw.

TYPETERM – ein Produkt von

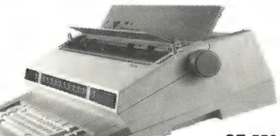
interkom electronic Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 05139-87393

Ausgabe mit TYPETERM® JUNIOR

im Slot Ihres

APPLE II/IIe/IIgs

Paketpreis **DM 899,–**
Schreibmaschine AX-10 mit Interface TYPETERM JUNIOR, steckfertig.



brother
Die Zukunft heute

TYPETERM JUNIOR mit AX-10 – unser besonders günstiges Gespann, ebenfalls steckfertig. Mit TYPETERM JUNIOR kann die AX-10 mehr. Sie wird zum vollwertigen Typenraddrucker für Ihren Apple:

- 3 verschiedene Schriftstärken
- Automatisches Unterstreichen
- 2 Zeichensätze z. B. deutsch u. ASCII
- 2 Zeichenabstände
- 2k ROM auf der Karte für Ausgabe unter DOS, PRODOS, CP/M u. PASCAL.

TYPETERM JUNIOR – ein Produkt von

interkom electronic Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 05139-87393

aaa electronic gmbh
Neue Preise # 1

NEU: Jetzt mit aaa-Computer-Kredit und bequemen monatlichen Raten!

Industrie-Standard IBM-PC/XT + AT-compatible COMPUTER + Zubehör in Industrie-Qualität (X=XT, A=AT, X/A=XT+AT)	LEER/BESTÜCKT
Mainboard XT-256, 8088 CPU, ØK (bis 256K-on board, oder 640 K mit Addr. auf Karte aufr.) + BIOS + 8 Slots, geprüft	79,—/227,—
Mainboard XT-256 mit 256 K-RAM, sonst w.o., gepr.	317,—
Mainboard XT-640, 8088 CPU, ØK (bis 640 K-on board) gepr.	89,—/237,—
Mainboard XT-640 w.o. jedoch 256 K-RAM best. + gepr.	297,—
Mainboard XT-640 w.o. jedoch 640 K – vollbest. und geprüft	407,—
Mainboard TURBO XT-640 -4,7/8 MHz, ØK, best. + gepr.	99,—/276,—
Mainboard AT-1MB, 6/8 MHz, 80286 CPU, 16/24 bit (mit Sockel f. 80287) 512 K-RAM best. (bis 1MB aufrüstb.)+BIOS+Uhr m.Akku+8 Slots,gepr.	1229,—
64KB-RAM-Chip-Aufrüstsatz (9 Stück) einzeln geprüft! X/A	32,—
256 KB-RAM-Chip-Aufrüstung (9 Stück) einzeln geprüft! X/A	69,—
Disk-Controller gepr. (f. 2x360 KB-Drives), X	77,—
Disk-Controller gepr. (f. 4x360 KB-Drives)(Extra Kabel=32,–), X	34,—/94,—
Disk-Controller gepr. (f. 2x1,2 MB-Drives), X	229,—
Disk-Controller gepr. (f. 2x1,2 MB-Drives, auch f. 360 K geeignet), A	169,—
Harddisk-Controller f. 2xHD, X	287,—
Harddisk + Floppy-Controller (1 HD+2DD) gepr., A	597,—
Monochrome Karte geprüft, X/A	49,—/149,—
Monochrome Graph.+par. Print. Karte (Hercules comp.), X/A	55,—/168,—
Color RGB + Video Graphikkarte + Lightpen-Ausg., gepr., X/A	55,—/126,—
Parallel Printer Karte gepr. o. Kabel (Extra Kabel = 49,–), X/A	29,—/ 49,—
RS 232 Interface (2xRS 232, 1 davon bestückt), X/A	29,—/ 73,—
RS 232 + par. Printer Karte gepr., X/A	136,—
512K-RAM-Karte gepr. (ØK)(mit 64 KB-RAM bestückbar), X	55,—/ 85,—
512K-RAM-Karte gepr. (ØK)(mit 256 KB-RAM bestückbar-kurz), X	114,—
3 MB-RAM-Erweiterungsk. gepr. (ØK), m. 256 K-RAM bestückbar), A	317,—
2,5 MB-Multifunktionskarte, ØK, +RS 232, +par. Printer, gepr., A	397,—
384K-Multifunktionskarte, RS 232, par. Printer. Interf., Game/Joystick Port, Real Time Clock gepuff., OK+RAM-Disk/Spooler/Uhr Softw., X	65,—/189,—
Disk Multi I/O Karte (f. 2x360 K-Drive, 2xRS 232, 1xpar. Printer Int., Game/Joystick Port, Real Time Clock/Kalender gepuff.), X	59,—/169,—
Clock Karte gepuffert, geprüft, X	69,—
I/O Plus-II-Karte, 2xRS 232 (1xbest.), par. Printer-Interf., Game /Joystick Port, Real Time Clock gepuffert, geprüft, X	179,—
I/O Plus-Karte, 2xRS 232 (1xbest.), par. Printer-Interf. Game/Joystick Port, geprüft, A	150,—
EGA-comp. Enhanced Graphic Adapter Karte, mit 256 K-RAM-Bildschirmspeicher, Monochrome-Color Graphik (640x350 Punkte – 16 Farben) + par. Printer Int., voll best. + gepr., X/A	68,—/495,—
AD/DA-Wandler, 12 bit, 16/1, X/A	295,—
AD/DA-Wandler, 14 bit, 16/2, X/A	475,—
Option Board compatible Karte incl. Disk + Manual, (zum sichern kopiergeschützter Disketten, Kopierschutz wird mitübertragen), X/A	347,—
Eprom Writer, (f. 4 St. 2716/32/64/128/256/512 gleichzeitig), X/A	429,—
Prototype/Lochraster Leerkarte, durchkontaktiert, lang	84,—
Profi-Funktionstastatur ASCII + 15er Block ab 195,— /dt. ab	225,—
135W/150W-Netzteil mit eingeb. Ventilator, kurzschlußf., X	185,—/215,—
200W-Netzteil mit eingeb. Ventilator, kurzschlußfest, A	375,—
360KB-Diskdrive Slimline (Track Access Time <4 mS), R/W gepr., X/A	277,—
720KB-Diskdrive Slimline, 2x80 Tr. (TAT <2mS), R/W gepr., X/A	337,—
1,2MB-Diskdrive Slimline, 2x80 Tr. (TAT <2mS), R/W gepr., X/A	347,—
21 MB-NEC-Harddisk, X/A 979,— Micro Science, X/A	849,—
21 MB-NEC-HD+HD-Contr., X 1.255,— MS+HD-Contr., X	1.155,—
44 MB-NEC-Harddisk, X/A 1.857,— 44 MB-NEC+DTC, X	2.349,—
21 MB, Harddisk+Controller in einem, einsteckfertig, X	1.298,—
Joystick f. IBM + Compatible, X/A	58,—
Maus incl. Software (MOUSE Systems comp.), X/A	158,—
Met.-Gehäuse aufkl. (Pl. f. 4-Einb.-Drives), solide Ausf., X=198,—/ A=298,—	
+++PC/XT+AT – KOMPLETT-SYSTEME AUFGEBAUT UND GEPRÜFT+++	
XT-640, 8088 CPU (incl. Boot-Eprom) 256K-Ram incl. Lautsprecher, im IBM-look-like Met.-Gehäuse + Profi-Fkt. Tast. + 135 W-Netzt. gepr.	777,—
XT-640, w.o. + Contr. + 360KB Drive + Color RGB + Vid. Graphik-Karte	987,—
XT-640, w.o. jedoch statt Color mit Monochrome Karte	997,—
XT-640, w.o. jedoch mit Monochrome-Graphik-Printer-Karte	1.017,—
XT-640, 256 K-RAM + 360 KB Drive + Disk Multi I/O + Color-Gr.-Karte	1.097,—
XT-640, 256 K-RAM + 360 KB Drive + Color RGB + Video-Graphik-Karte + 21 MB-Festplatte + Controller	2.097,—
AT-1MB, 6/8 MHz, 80286 CPU, 512 K-RAM, im IBM-look-like Met.-Gehäuse + Profi-Fkt. Tast. + 200W-Netzt. + Color Graph.-Karte + 1,2MB-Drive, gepr.	2.295,—
AT-1MB, w.o. + 21 MB-Harddisk gepr.	3.785,—
Aufpreis für Turbo XT-640 -4,7/8 MHz umschaltbares Board, X	39,—
Aufpreis für XT-640 M-board mit 640 K von 256 auf 640 K-RAM, X	119,—
Aufpreis f. EGA anstatt Color Graph.-Karte, X/A	475,—
MS-DOS 3.1 Diskette incl. Manual, X/A	159,—
MS-DOS + GW-Basic 3.2 sowie 4 dt. orig. Microsoft Hb. (Aufpreis in Verbindung mit Systemkauf), X/A	189,—
8087-Arithmetik-Prozessor, 5 MHz, X = 327,— 80287, A	389,—
V 20-Prozessor, 8 MHz, X	37,—

PC 10-II 512 K-RAM, AGA-Karte, dt. Tast. + Laufw. + 20 MB-Festplatte + Contr. + Monitor	3.887,—
AMIGA-Commodore, 512 K-RAM	1.939,—

APPLE-BUS COMPUTERPLATINEN + PERIPHERIE	LEER/BESTÜCKT
M-board II 48K, gesockelt, vollbest. + gepr.	57,—/337,—
M-board II 64K, gesockelt, vollbest. + gepr.	88,—/378,—
M-board II 64K + Z80 CPU, gesockelt, vollbest. + gepr.	95,—/378,—
M-board IIe 64K vollbestückt + gepr.	98,—/389,—
Superstarkes Netzteil – kurzschlußfest 5A=115,— bis zu 7A=	145,—
Entwicklungslochraster-Leerplatine durchkontaktiert	27,—
16K-Speicher-Karte geprüft	18,95/ 55,—
APPLE orig. 16K-Language Karte gepr.	98,—
Z-80 CPU-Karte geprüft (CP/M Vol. I + II Manual dafür = 69,—)	18,95/ 64,—
Controller DOS 3.3-Karte geprüft	18,95/ 63,—
Auto-Controller DOS 3.2/3.3-Karte gepr.	24,—/ 97,—
Erphi AFDC 2 Controller + Autopatch Softw. + Handbuch, gepr.	198,—
Parallel-Drucker-Karte geprüft (Extra Kabel = 45,—)	18,95/ 74,—
Par.-Drucker Karte-Grappler comp. gepr. (Extra Kabel = 45,—)	27,—/169,—
8bit par. Int. + 64K-Buffer + Kabel gepr. (Grappler comp.)	39,—/299,—
Seriell-Interface-Karte V24 geprüft	18,95/ 88,—
Super-Seriell-Interface Karte gepr.	34,—/169,—
APPLE-Original Super-Seriell-Interface gepr.	295,—
80-Zeichen/24-Zeilen-Karte geprüft	18,95/119,—
80Z/24Z-Karte + Softswitch-Schalter, gesockelt scharf, gepr.	24,—/127,—
80Z/24Z-Softwitch Karte gepr. für IIe	15,—/ 39,—
80Z/24Z + 64K-Ram + Softw.-Karte best. + gepr. f. IIe	18,95/ 87,—
IEEE-488 Int. Karte gepr. (Extra Kabel = 45,—)	29,—/297,—
PAL-Modulator/Color-Karte + UHF Modulator, geprüft	18,95/ 99,—
RGB-Color Karte gepr. (nur f. II+)	18,95/129,—
128K-Speicher-Leerkarte + Software + Manual	99,—
128K-Speicher-Karte + Software + Manual, geprüft	29,—/268,—
Mega Ramcard II, 256 KB best. + Software + dt. Hb., (bis 1 MB aufr.)	498,—
Mega Ramcard II, 1 MB best. + Software + dt. Hb., gepr.	878,—
EPROM-Burner (2716/32/64) geprüft	24,—/109,—
6522-VIA Karte geprüft	24,—/135,—
Clock Karte + Software + Manual, geprüft	24,—/ 94,—
Speech Karte + Software + Manual, geprüft	18,95/ 53,—
Wild Karte + Software + Manual (nicht f. IIe), geprüft	18,95/ 59,—
AD/DA-8Bit Karte + Software + Manual, gepr.	29,—/269,—
UHF/TV-Modulator universell	27,50
Lüfter anclipsbar (220 V)	79,—

APOLLO//(48K) + UHF-Mod. + Gr/Kl vollbest. + gepr.	617,—
APOLLO//(48K) + Disk II F + Contr. + 12"-Monitor	1.269,—
APOLLO//A (48K) + Gr/Kl + 15er-Tastatur vollbest. + gepr.	677,—
APOLLO// ASKF (48K) vollbest. + gepr. sep.Keyboard + Gr/Kl + 15er-Block mit Funkt.-Tasten im „IBM-look like" Gehäuse	835,—
Aufpreis f.o. Systeme von 48K zu 64K	55,—
Aufpreis von 48K zu 64K+Z80 CPU	89,—
Apollo IIe (64K) vollbest. und geprüft	675,—
Apollo IIe (64K) + 80Z/24Z-Karte gepr.	698,—
Apollo IIe (128K) + 80Z/24Z-Karte geprüft	755,—
Apple IIe (64 K) + 15er-Tastatur vollbestückt und geprüft	735,—
Apollo IIe ASKF (64 K) vollbestückt und geprüft	ab 845,—
Apollo IIe (64K) + Disk II F + Contr. + 12" Monitor	1.387,—

APPLE IIe 128K + 80Z/24Z Karte + Softw.	1.598,—
Disk-Drives voll APPLE II, 100% compatibel + R/W geprüft + Track 0 Sensor	
Disk// + Contr. + Kabel DOS 3.3 (Siemens) im Geh.	353,—
Disk// (Siemens) im Geh. + Kabel	298,—
Disk//F-HS (High Speed 1/2 Höhe) im Gehäuse + Contr. + Kabel	379,—
Disk//F-HS (High Speed 1/2 Höhe) im Geh. + Kabel (Track Ac. <2mS)	318,—
Disk//F-HS-80 Track im Gehäuse + Kabel f. orig. Contr. geeignet	389,—
Disk//F-HS-80 w.o. + Contr. + DOS 3.3 + CP/M Patch-Softw.	459,—
Disk II-2x80 Track 640KB f. Erphi Controller modifiziert, o. Gehäuse	299,—
Erphi DuoDisk 1,2 MByte im Gehäuse + Erphi-AFDC 3-Contr., gepr.	854,—

APPLESOFT + Tutorial + Reference Manual engl.	75,—
APPLE-DOS 3.3 Manual engl. = 25,—/Handbuch dt.	45,—
APPLE-Pascal Reference + Operating Manual engl.	48,—
APPLE-Fortran Manual engl.	69,—
CP/M-Softcard Vol. I + II, Manual engl.	69,—

Z80 Karte gepr. f. APPLE//c + RAM-Disk-Software + Hb.	447,—
Macintosh Umrüstung von 128K auf 512K	335,—

EPSON FX 85F/T+, 8bit/parallel 160 Z/S. NLQ + IBM comp.	1.097,—
EPSON FX 105+, 8bit/parallel 15" sonst w.o.	1.397,—
EPSON LQ-800, 24 Nadel, 180 Z/Sec. NLQ+IBM comp.	1.488,—
EPSON LQ-1000, wie 800 breite Ausf.	1.937,—
EPSON EX 800,	1.494,—
EPSON Traktoraufsatz für FX 80 + FX 85	129,50
APPLE/EPSON-Drucker Graphic-Interface + Kabel	169,—
MX 80/82, FX/RX 80, FX 85 Spezialfarbband-Kassette	15,95
BROTHER-Typenradd.: HR 15XL R/II/par. neuestes Modell	1.087,—

Wir führen verschiedene Monitore von Zenith, Phillips, NEC, COGA und Sanyo mit Video – TTL (IBM Komp.) oder Coloreingang. Sowie umfangreiche Computer-Literatur (über 500 versch. Titel) – Bitte Preisliste anfordern! –

Disketten in Box + Aufkleber l. Wahl, 10er Pack/100er Pack./ Stückpreis:

5¼"	BASF	3M	FUJI	DATA MAGNETICS	NEUTRAL
1X, SS/SD	2,79/ 2,59				
1D, SS/DD	2,89/ 2,69	2,88/ 2,78	4,27/ 3,97	,980/ ,936	,858/ ,835
2D, DS/DD	3,49/ 3,29	3,88/ 3,68	5,87/ 5,47	1,150/ ,995	1,055/ ,895
1D/96TPI	3,89/ 3,59	4,88/ 4,68			
2D/96TPI	4,39/ 4,19	4,68/ 4,38	7,87/ 7,67		
2HD/1,2MB	7,59/ 7,29	7,98/ 7,58	9,87/ 9,57		
1X, Hard	4,29/ 4,19				
8"					
1X, SS/SD	4,49/ 4,39	6,68/ 6,48	6,97/ 6,67		
1D, SS/DD	4,79/ 4,59	8,58/ 8,28			
2D, DS/DD	6,49/ 5,99	9,88/ 9,58	10,17/ 9,77		
1X, Hard	4,48/ 4,39	6,68/ 6,48			
3½"					
Einseitig	4,69/ 4,49	5,88/ 5,58	7,67/ 7,17		
Zweiseitig	6,19/ 5,89	7,28/ 6,88	9,17/ 8,57		

Das sind Preise!

5¼" Disketten-Archivbox für 10 Disks (1 St./10 St.)	4,75/43,50
5¼" Disk-Karteikasten Kunststoff (ca. 80 Disk)	26,50
5¼" Disk-Karteikasten Kunststoff (ca. 80 Disk) Rauchglas	45,95
5¼" Disk-K.K.Rauchglas, abschließb. (f. ca. 80-100 Disks)	47,50
2000 Bl. Tabpapier (24 cm x 12" einf.) weiß oder grün/weiß perf.	45,—
4000 Et.-Aufkl. doppelr. (107 x 36 mm auf 240 x 12" perf. Trägerp.)	66,50
4000 Etik.-Aufkl. einreih. (107 x 36 mm auf 125 x 12" perf. Träg.)	65,—

OSZILLOSCOPE HAMEG ab Lager
Bei Vorauszahlung frei Empfangsstation unversichert in der BRD, ausgenommen Papier und Eiltsendungen, sonst N.N. + V.S. ab DM 30,—
Öffnungszeiten: Mo, Di, Do, Fr v. 10-18 h, Mi u. Sa v. 10-14 h, la Sa v. 10-18 h
Telef. Best.: Mo, Di, Do, Fr von 10-19 h, Mi u. Sa wie Öffnungszeiten.
GEWÄHRLEISTUNG: 6 Monate auf alle bei uns gekauften Geräte, durch unsere eigene Service-Werkstatt.

REPARATUREN an Apple + compatiblen Geräten + Zubehör führt unser Spezialistenteam garantiert zuverlässig + besonders kostengünstig aus.
Sprechen sie mit uns. Kostenvorschlag auf Wunsch!

aaa electronic gmbh
Telex 0772642 aaa-d
Habsburgerstraße 134
7800 FREIBURG, Tel (0761) 276864
Bauelemente – Bausätze – µP's
Meßgeräte – Zubehör – Fachliteratur
Fachgeschäft für Elektronik + Mikrocomputer

Luxus-Catalog in Kyan-Pascal

von Matthias Meyer

Die Idee für diesen Include-Befehl entstand beim Testen der System Utilities der Firma Kyan Software. Diese Programmsammlung enthielt einen größtenteils in Pascal geschriebenen, meiner Meinung nach speicherverschlingenden und äußerst langsamen Catalog-Befehl. Aus diesem Grund, und auch weil bis jetzt noch kein ausführlicher Catalog-Befehl als Include-File für Kyan-Pascal veröffentlicht worden ist, möchte ich Ihnen nachfolgend einen effizient arbeitenden, größtenteils in Assembler geschriebenen Catalog-Befehl vorstellen.

Der Catalog-Befehl (s. Include-Datei-Listing **CAT.I**) beherrscht exakt die gleichen zwei Ausgabeformate wie das BASIC.SYSTEM, d.h. es werden Zeilen von 40 oder 80 Zeichen Breite mit detaillierten File-Informationen vollgepackt. Der Catalog-Befehl wird dann mit

errcode := cat (pathname, format, files);

aufgerufen.
Die Funktion cat liest ein beliebiges (Sub-) Directory in den Speicherbereich der Hires-Seite 1 ein und gibt danach bis zu 181 File-Einträge im gewünschten Ausgabeformat auf dem Bildschirm aus. Ein möglicher MLI-Fehler wird der Variablen errcode zugewiesen (0 = kein Fehler).

Zu Beginn Ihres eigenen Pascal-Hauptprogramms müssen die Variablen definiert werden, die in dem Demo-Programm **CAT.P** durch gestrichelte Trennlinien abgegrenzt worden sind. Darüber hinaus muß Ihr Programm mit „useshires" eingeleitet werden.
Das Demo-Programm CAT.P kann als praktisches Beispiel dafür dienen, wie man den neuen Catalog-Befehl in eigene Anwendungen integrieren kann. Der eigentliche Befehl ist bis auf die dezimale Bildschirmdarstellung in reinem Assembler geschrieben und vollständig dokumentiert, so daß die genauere Programmbeschreibung dem Listing zu entnehmen ist.

Dieser komfortable Catalog-Befehl eignet sich auch gut als Ergänzung zu bereits vorhandenen Kyan-Utilities, wie z.B. der ProDOS-Runtime-Library FID (Kyan-Club-Disk #A; vgl. auch Peeker, Heft 7/86, S. 40ff.) oder den erwähnten System Utilities der Firma Kyan Software.

Kurzhinweise

1. Zweck:
Catalog-Befehl für Kyan-Pascal
2. Konfiguration:
Apple II+/e/c; ProDOS; Kyan-Pascal ab Version 2.0
3. Test:
CAT über Kix-Menü (%) starten
4. Sammeldisk:
CAT.I (Include-File)
CAT.P (Demo)
5. Sonstiges:
Die Dateien CAT.I und CAT.P müssen zunächst mit CONVERT oder DOSTOPRO von der DOS-3.3-Sammeldisk auf Ihre Kyan-Arbeitsdisk konvertiert werden.

CAT.P

```
#a
_useshires
#
program catalog(input,output);
{.....................................................}
type
 pathstring = array [1..65] of char;
 filestring = array [1..16] of char;
var
 format: char;
 pathname: pathstring;
 filename: filestring;
 files,index,vsize,fsize,errcode: integer;
#i cat.i
{.....................................................}
begin
repeat
#a
 stx _t
 jsr $fc58 ;home
 ldx _t
#
 writeln('CATALOG-PROGRAMM FUER KYAN-PASCAL V2.Ø2');
 writeln;
 writeln('VON MATTHIAS MEYER (JULI 1986)');
 writeln;
 writeln;
 writeln('CATALOG-FORMATE: S = 4Ø Z/Z, L = 8Ø Z/Z');
 writeln('_____________________________________');
 writeln;
 readln(format)
until format in ['S','s','L','l'];
writeln;
write('BITTE PFADNAME EINGEBEN: ');
readln(pathname);
writeln;
write   (' NAME           TYPE  BLOCKS  MODIFIED');
if format in ['L','l'] then
 writeln('        CREATED         ENDFILE SUBTYPE')
else writeln;
writeln;
errcode:=cat(pathname,format,files);
if errcode>Ø then
 prtmlierror(errcode)
else begin
 index:=1;
 repeat
  filename[index]:=pathname[index];
  index:=index+1
 until pathname[index] in [' ','/'];
 filename[index]:=' ';
 errcode:=volumesize(filename,vsize,fsize);
 writeln;
 if errcode>Ø then
  prtmlierror(errcode)
 else begin
  write   ('BLOCKS USED: ',fsize:5,'   BLOCKS FREE: ',
   vsize-fsize:5);
  if format in ['L','l'] then
   writeln('  TOTAL BLOCKS: ',vsize:5,'  TOTAL FILES: ',
    files:3)
  else writeln
 end
end
end.
```

CAT.I

```
function cat(p:pathstring;f:char;var files:integer):integer;
var      msb,dec:integer;
begin
cat:=Ø;  {Funktionswert von 'cat' = Fehlernummer von ProDOS}
#a
;
blocki   equ  _t+1       ;Index auf Block im Puffer
printi   equ  _t+3       ;Index auf Catalog-Einträge
flag     equ  _t+5       ;Markierung für Catalog-Ende
namelen  equ  _t+6       ;Länge des Catalog-Eintrags
files    equ  _t+7       ;Zähler für Anzahl Einträge
buffer   equ  $2ØØØ      ;Start des 1K Open-Puffers
subdir   equ  $24ØØ      ;Start des Directory-Puffers
prbyte   equ  $fdda      ;Ausgabe einer Sedezimalzahl
;
         stx  _t         ;Register von Pascal retten.
         ldy  #13        ;Zeigt auf das gewählte
         lda  (_sp),y    ;Catalog-Format (S oder L).
         and  #$5f       ;Format als Großbuchstabe
         sta  format     ;zwischenspeichern.
;
         iny             ;Y zeigt auf den Pfadnamen.
pathle   lda  (_sp),y    ;Länge des Pfadnamens
         cmp  #32        ;berechnen.
         beq  lfound
         iny
         bne  pathle
;
lfound   tya
         sec
         sbc  #14        ;Länge des Pfadnamens vor
         ldy  #13        ;erstem String-Byte ablegen.
         sta  (_sp),y
;
         tya
         clc
         adc  _sp        ;Adresse des Pfadnamens mit
         sta  pname      ;Längenbyte berechnen und für
         lda  _sp+1      ;den ProDOS-Open-Call ablegen.
         adc  #Ø
         sta  pname+1
;
         jsr  $bfØØ      ;Erst einmal vorsichtshalber
         db $cc          ;alle offenen Dateien schlie-
         dw closeall     ;ßen.
         bcs  direrr
;
         jsr  $bfØØ      ;das durch den Pfadnamen be-
         db $c8          ;schriebene (Sub-)Directory
         dw opendir      ;wird geöffnet.
         bcs  direrr
;
         jsr  $bfØØ      ;Das Directory wird jetzt in
         db $ca          ;den dafür vorgesehenen Puf-
         dw readdir      ;ferbereich eingelesen.
         bcs  direrr
;
         jsr  $bfØØ      ;Anschießend wird die Direc-
         db $cc          ;tory-Datei wieder geschlos-
         dw closeall     ;sen.
         bcc  noderr
;
direrr   ldy  #9         ;Y zeigt auf Funktionswert.l
         sta  (_sp),y    ;Ein aufgetretener ProDOS-
         jmp  exitloop   ;Fehler wird hier vermerkt.
;
noderr   equ  *
         lda  #$ØØ       ;Startwerte für Catalog-Aus-
         sta  flag       ;wertung werden hier initia-
         sta  files      ;lisiert.
         lda  #>subdir
         sta  blocki     ;Index auf aktuellen (Sub-)
         lda  #<subdir   ;Directory-Block.
         sta  blocki+1
;
loopl    lda  blocki+1   ;Index auf aktuellen Block.h
         cmp  #<$4ØØØ    ;Ende des Directory-Puffers?
         bne  goonØ      ;Hoffentlich nicht!
         jmp  exitloop
goonØ    ldy  #2
         lda  (blocki),y ;Ist dies hier der letzte
         bne  goon       ;Directory-Block (keine Vor-
         iny             ;wärts-Referenz)?
         lda  (blocki),y
         bne  goon
         dec  flag       ;Ja, hier vermerken.
goon     lda  blocki+1
         cmp  #<subdir   ;War das der Key-Block?
         bne  nokey
         lda  #$2b       ;Ja, 1. Eintrag überspringen.
         bne  loop2      ;Verzweige immer.
nokey    lda  #4         ;Nein, kein Key-Block.
;
loop2    tay             ;Y zeigt auf den Start eines
         lda  (blocki),y ;File-Eintrags.
         bne  display    ;Wenn A<>Ø, dann File aktiv,
         jmp  nodispl    ;sonst gelöscht.
display  and  #$Øf       ;Länge des Eintrags im Akku.
         sta  namelen    ;Länge um eins erhöhen und
         inc  namelen    ;vorerst zwischenspeichern.
;
         tya
         pha             ;Y-Zeiger auf Stapel retten.
         clc
         adc  blocki     ;Y-Index zum Blockindex hin-
```

```
        sta  printi     ;zuaddieren und als Print-
        lda  blocki+1   ;index abspeichern.
        adc  #$00
        sta  printi+1
;
; Schreibschutz anzeigen:
;
        ldy  #$1e       ;Zeiger auf Access-Eintrag
        lda  (printi),y ;Access-Eintrag überprüfen.
        and  #$c2       ;File ist schreibgeschützt?
        beq  protect    ;Ja, * anzeigen.
        lda  #$20       ;Nein, also nur Leerzeichen.
        jmp  loop2a
protect lda  #$2a       ;='*'
loop2a  jsr  outdo      ;Standard-ASCII-Ausgabe
;
; Filename anzeigen:
;
        ldy  #1         ;Zeiger auf Filename
loop3   lda  (printi),y ;Zeichen aus String holen.
        cpy  namelen    ;Ende des Namens erreicht?
        bcc  outdo0     ;Nein, noch nicht.
space   lda  #$20       ;Ja, nur noch Leerzeichen.
outdo0  jsr  outdo      ;Standard-ASCII-Ausgabe
        iny
        cpy  #$11       ;Länge von Name+Leerzeichen
        bne  loop3
;
; Filetype anzeigen:
;
        dey             ;Zeiger auf Filetype
        lda  (printi),y ;Akku enthält den Filetype.
        ldx  types      ;Anzahl bekannter Typen -1
loop4   cmp  typeno,x   ;Teste, ob Typ bekannt ist.
        beq  tmatch     ;Ja, stimmt überein.
        dex
        bpl  loop4
        pha             ;Nein, stimmt nicht überein.
        lda  #$24       ;='$'
        jsr  outdo      ;Standard-ASCII-Ausgabe
        pla             ;Unbekannte Filetype-Nummer
        jsr  prbyte     ;als Sedezimalzahl ausgeben.
        jmp  loop4a
tmatch  txa             ;Index auf Typ zum Akku und
        sta  typndx     ;zwischenspeichern.
        asl             ;Mit zwei multiplizieren und
        clc             ;den ursprünglichen Wert
        adc  typndx     ;adddieren.
        tax             ;Ergebnis im X-Register.
        lda  typetx,x   ;Nun die durch X indizierte,
        jsr  outdo      ;zu dem bekannten Filetype
        lda  typetx+1,x ;gehörende Kurzbezeichnung
        jsr  outdo      ;ausgeben.
        lda  typetx+2,x
        jsr  outdo
;
loop4a  ldx  #3
loop4b  lda  #$20       ;Vier Leerzeichen ausgeben.
        jsr  outdo
        dex
        bne  loop4b
;
; Block Used anzeigen:
;
        ldy  #$14       ;Zeiger auf Blocks Used
        lda  (printi),y ;höherwertigen Teil vom Wert
        pha             ;auf Stapel zwischenspeichern
        dey
        lda  (printi),y ;niederwertigen Teil direkt
        ldy  #5         ;in die Pascal-Integer-
        sta  (_sp),y    ;Variable DEC kopieren.
        iny
        pla             ;Wert vom Stapel ebenfalls
        sta  (_sp),y    ;in diese Variable kopieren.
        jsr  savezero   ;Zeiger retten und X laden.
#
write(dec:5); {Integer-Variable als Dezimalzahl ausgeben}
#a
        jsr  loadzero   ;X retten und Zeiger laden.
        lda  #$20       ;Zwei Leerzeichen ausgeben.
        jsr  outdo
        lda  #$20       ;=' '
        jsr  outdo
;
; Last Modification anzeigen:
;
        ldx  #3
        ldy  #$24       ;Zeiger auf Last Mod.
modifd  lda  (printi),y ;Last Modification date/time
        sta  date,x     ;nach GetDate kopieren.
        dey
        dex
        bpl  modifd
        jsr  getdate    ;Datum und Uhrzeit ausgeben.
        lda  format     ;Überprüfung Ausgabeformat:
        cmp  #$4c       ;='L'ong format?
        beq  contln     ;Ja, Ausgabe fortsetzen.
        jmp  endline    ;Nein, Ausgabe fertig.
contln  lda  #$20       ;Zwei Leerzeichen ausgeben.
        jsr  outdo
        lda  #$20       ;=' '
        jsr  outdo
;
; Creation date+time anzeigen:
;
        ldx  #3
        ldy  #$1b       ;Zeiger auf Creation
creatd  lda  (printi),y ;Creation date/time
        sta  date,x     ;nach GetDate kopieren.
        dey
        dex
        bpl  creatd
        jsr  getdate    ;Datum und Uhrzeit ausgeben.
        lda  #$20       ;Zwei Leerzeichen ausgeben.
        jsr  outdo
        lda  #$20       ;=' '
        jsr  outdo
;
; End Of File anzeigen:
;
        ldy  #$17       ;Zeiger auf End Of File
        lda  (printi),y ;Die höherwertigen Bytes auf
        pha             ;dem Stapel zwischenspeich.
        dey
        lda  (printi),y
        pha
        dey
        lda  (printi),y ;Das niederwertigste Byte
        ldy  #5         ;von EOF in das Low Byte
        sta  (_sp),y    ;der Variablen DEC kopieren.
        iny
        pla             ;Aus dem mittleren Byte von
        asl             ;EOF wird Bit 7 herausgenom-
        php             ;men und im Carry-Flag
        lsr             ;zwischengespeichert.
        plp             ;Das mittlere Byte wird dann
        sta  (_sp),y    ;zum High Byte von DEC.
        iny             ;Das höchstwertige Byte von
        pla             ;EOF wird mit Bit 7 von vor-
        rol             ;her nach links rotiert und
        sta  (_sp),y    ;danach ins Low Byte der
        iny             ;Variablen MSB kopiert.
        lda  #0         ;Schließlich wird das High
        sta  (_sp),y    ;Byte von MSB initialisiert.
        jsr  savezero   ;Zeiger retten und X laden.
#
write(32768e0*msb+dec:7:0); {Ausgabe als Dezimalzahl}
#a
        jsr  loadzero   ;X retten und Zeiger laden.
        lda  #$08       ;Letztes Zeichen war ein
        jsr  outdo      ;Dezimalpunkt. Mit Leerzeichen
        lda  #$20       ;überschreiben.
        jsr  outdo
;
; Auxiliary Type anzeigen:
;
        ldy  #$10       ;Zeiger auf Filetype
        lda  (printi),y ;Akku enthält den Filetype.
        tax             ;Filetype in X zwischenspeichern
        cmp  #$0f       ;Ist es ein DIR-File?
        bne  tsttxt     ;Nein, weitertesten.
        jmp  endline    ;DIR hat keinen Aux.Type.
tsttxt  cmp  #$04       ;Ist es ein TXT-File?
        bne  notext     ;Nein, ein normaler Aux.Type
        lda  #$52       ;='R': Bei TXT-Files ist der
        bne  prtequ     ;Aux.Type die 'R'ecord-Länge
notext  lda  #$41       ;='A': Sonst einfach nur ein
prtequ  jsr  outdo      ;gewöhnlicher 'A'ux.Type.
        lda  #$3d       ;='='
        jsr  outdo      ;Standard-ASCII-Ausgabe
        cpx  #$04       ;War es ein TXT-File?
        beq  decimal    ;Ja, also dezimale Ausgabe.
        lda  #$24       ;='$'
        jsr  outdo      ;Standard-ASCII-Ausgabe
        ldy  #$20       ;Zeiger auf Aux.Type high
        lda  (printi),y ;High Byte von Aux.Type in A
        jsr  prbyte     ;Akku sedezimal ausgeben.
        ldy  #$1f       ;Zeiger auf Aux.Type low
```

```
        lda  (printi),y ;s.o.
        jsr  prbyte
        jmp  endline    ;Ausgabezeile beenden.
;
decimal ldy  #$2Ø       ;Zeiger auf Aux.Type high
        lda  (printi),y ;höherwertigen Teil vom Wert
        pha             ;auf Stapel zwischenspeich.
        dey
        lda  (printi),y ;niederwertigen Teil direkt
        ldy  #5         ;in die Pascal-Integer-Vari-
        sta  (_sp),y    ;able DEC kopieren.
        iny
        pla             ;Wert vom Stapel ebenfalls
        sta  (_sp),y    ;in diese Variable kopieren.
        jsr  savezero   ;Zeiger retten und X laden.
#
write(dec:5); {Integer-Variable als Dezimalzahl ausgeben}
#a
        jsr  loadzero   ;X retten und Zeiger laden.
;
endline lda  #$Ød       ;Carriage Return
        jsr  outdo      ;ausgeben.
        pla             ;Y zeigt auf den Start eines
        tay             ;File-Eintrags.
        inc  files      ;Erhöhe Anzahl der Einträge.
;
nodispl tya             ;Einsprung für unsichtbare,
        clc             ;d.h. gelöschte Einträge.
        adc  #$27       ;Offset für nächsten Eintrag
        bcs  nodisl     ;addieren.
        jmp  loop2      ;Nächsten Eintrag anzeigen.
nodisl  pha             ;Low Byte retten.
        inc  blocki+1   ;Blockanfang wegen Referenz-
        lda  blocki+1   ;Header gesondert behandeln:
        lsr             ;ist das High Byte ohne Rest
        pla             ;durch zwei teilbar, so be-
        bcc  skip2      ;ginnt ein neuer Block, von
        jmp  loop2      ;dem die ersten vier Bytes
skip2   bit  flag       ;zu übergehen sind. Das Flag
        bmi  exitlp     ;ist negativ, wenn das Ende
        jmp  loopl      ;des Directory erreicht ist.
exitlp  jmp  exitloop
;
types   db $Ø9          ;Anzahl Tabellenelemente - 1
typeno  db $Ø4,$Ø6,$Øf,$19,$1a,$1b,$fc,$fd,$fe,$ff
typetx  asc  'TXT'      ;ASCII text file
        asc  'BIN'      ;General binary file
        asc  'DIR'      ;Directory file
        asc  'ADB'      ;Data base file
        asc  'AWP'      ;Word processor file
        asc  'ASP'      ;Spreadsheet file
        asc  'BAS'      ;Applesoft program file
        asc  'VAR'      ;Applesoft variables file
        asc  'REL'      ;Relocatable code file
        asc  'SYS'      ;ProDOS system file
typndx  ds 1
;
savezero ldx  #7        ;Zero-Page-Bereich $1Ø-$17
savezØ  lda  _t,x       ;nach tmpzero speichern.
        sta  tmpzero,x
        dex
        bpl  savezØ
        ldx  _t         ;Pascal-X-Register laden.
        rts
;
loadzero stx  _t        ;Pascal-X-Register retten.
        ldx  #7         ;Zero-Page-Bereich $11-$17
loadzØ  lda  tmpzero,x  ;aus tmpzero zurückladen.
        sta  _t,x
        dex
        bne  loadzØ
        rts
;
tmpzero ds 8            ;Zero-Page-Zwischenspeicher.
;
format  ds 1            ;short/long Format
;
closeall db 1           ;ProDOS-MLI-Aufruf:
        db $ØØ          ;alle Dateien schließen.
;
opendir db 3            ;ProDOS-MLI-Aufruf:
pname   dw $ØØØØ        ;(Sub-) Directory öffnen.
        dw buffer
        db $Ø1
;
readdir db 4            ;ProDOS-MLI-Aufruf:
        db $Ø1          ;(Sub-) Directory lesen.
        dw subdir
        dw $1cØØ
```

```
        dw $ØØØØ
;
; GetDate-Unterprogramm: Es übernimmt Datum und Uhrzeit
; von ProDOS in der komprimierten Form und gibt es dann
; im üblichen Format auf dem Bildschirm aus.
;
getdate lda  date+1
        lsr             ;Jahr herausfiltern
        sta  year
;
        lda  date       ;Byte mit Monat und Tag
        bne  dateok     ;Wenn >Ø, dann Datum OK.
        jmp  nodate     ;Kein Datum gesetzt.
;
dateok  ror             ;Mit Bit Ø von date+1 rotie-
        lsr             ;ren und Tag wegschieben.
        lsr             ;Hier werden die Daten des
        lsr             ;Monats zur richtigen Posi-
        lsr             ;tion gebracht.
        sta  month
;
        lda  date       ;Byte mit Monat und Tag
        and  #$1f       ;Tag herausfiltern
        cmp  #1Ø        ;Ist der Tag im Dezimal-
        bcs  date4      ;system zweistellig?
        pha             ;Nein, erst einmal retten
        lda  #$2Ø       ;und ein Leerzeichen ausge-
        jsr  outdo      ;ben.
        pla
date4   jsr  dezout     ;Tag dezimal ausgeben.
        lda  #$2d       ;='-'
        jsr  outdo      ;Standard-ASCII-Ausgabe
        lda  month      ;Byte mit Monat als Textzei-
        asl             ;ger verwenden, also erst
        adc  month      ;einmal Monat mit drei mul-
        tax             ;tiplizieren und dann einen
        lda  months-3,x ;Text mit drei Buchstaben
        jsr  outdo      ;ausgeben.
        lda  months-2,x
        jsr  outdo
        lda  months-1,x
        jsr  outdo
        lda  #$2d       ;='-'
        jsr  outdo      ;Standard-ASCII-Ausgabe
        lda  year       ;Ausgabe der Jahresangabe in
        jsr  date6      ;dezimal mit führender Null.
        lda  format     ;Sollte die Datum-Ausgabe im
        cmp  #$4c       ;='L'ong Format erfolgen?
        bne  date8      ;Nein, das war's.
        lda  #$2Ø       ;Leerzeichen ausgeben.
        jsr  outdo
        lda  time+1     ;Byte mit Stunde
        cmp  #1Ø        ;Ist die Stunde im Dezimal-
        bcs  date5      ;system zweistellig?
        pha             ;Nein, erst einmal retten
        lda  #$2Ø       ;und ein Leerzeichen aus-
        jsr  outdo      ;geben.
        pla
date5   jsr  dezout     ;Stunde dezimal ausgeben.
        lda  #$3a       ;=':'
        jsr  outdo      ;Standard-ASCII-Ausgabe
        lda  time       ;Byte mit Minute
date6   cmp  #1Ø        ;Ist die Minute im Dezimal-
        bcs  date7      ;system zweistellig?
        pha             ;Nein, erst einmal retten
        lda  #$3Ø       ;='Ø'
        jsr  outdo      ;Standard-ASCII-Ausgabe
        pla
date7   jmp  dezout     ;Minute dezimal ausgeben.
date8   rts;

dezout  ldy  #5         ;Dezimal auszugebender Wert
        sta  (_sp),y    ;liegt im Akku vor und wird
        iny             ;in das Low Byte von DEC ko-
        lda  #Ø         ;piert. Nachfolgend wird das
        sta  (_sp),y    ;High Byte auf Null gesetzt.
        jsr  savezero   ;Zeiger retten und X laden.
#
write(dec); {Integer-Variable als Dezimalzahl ausgeben}
#a
        jmp  loadzero   ;X retten und Zeiger laden.
;
nodate  ldx  #Ø
txtout  lda  tnodate,x  ;Text: "Kein Datum" ausgeben
        beq  txtend
        jsr  outdo
        inx
        bne  txtout
txtend  lda  format     ;Sollte die Datum-Ausgabe im
```

```
         cmp  #$4c        ;='L'ong Format erfolgen?
         bne  date9       ;Nein, das war's.
         ldx  #6          ;Ja, deshalb gleich anschie-
date8a   lda  #$2Ø        ;ßend sechs Leerstellen aus-
         jsr  outdo       ;geben.
         dex
         bne  date8a
date9    rts
;
date     dw $ØØØØ        ;Hier wird Datum und Uhrzeit
time     dw $ØØØØ        ;im ProDOS-Format übergeben.
month    db $ØØ          ;Zwischenspeicher für Monat,
year     db $ØØ          ;       "          " Jahr.
;
months   asc  "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
;
tnodate  asc  "<NO DATE>"
         db $ØØ
;
outdo    ora  #$8Ø        ;Standard-ASCII-Ausgabe
         jmp  cout
;
exitloop clc
         ldy  #11         ;Die Anzahl der ausgegebenen
         lda  (_sp),y     ;File-Einträge wird nachfol-
         sta  _t+1        ;gend in die Pascal-Integer-
         iny              ;Variable FILES übertragen.
         lda  (_sp),y
         sta  _t+2
         ldy  #Ø
         lda  files
         sta  (_t+1),y
         iny
         lda  #Ø
         sta  (_t+1),y
         ldx  _t
#
end;

function volumesize
 (filename:filestring;var vsize,fsize:integer):integer;
begin
volumesize:=Ø; {Errcode:=Ø}
#a
         stx  _t          ;X-Register-Inhalt retten.
         ldy  #11         ;Y zeigt auf den Anfang des
flen     lda  (_sp),y     ;Volume-Namens. Der Volume-
         cmp  #32         ;name wird in einen Puffer
         beq  ldone       ;kopiert. Dabei wird gleich-
         sta  fname-1Ø,y  ;zeitig dessen Länge mitge-
         iny              ;zählt.
         bne  flen
;
ldone    tya
         sec
         sbc  #11         ;Die Länge des Volume-Namens
         sta  fname       ;wird hier abgespeichert.
;
         jsr  $bfØØ       ;ProDOS-MLI-Aufruf:
         db $c4           ;Get File Info.
         dw gfinfo
         ldy  #5          ;Eine eventuell aufgetretene
         sta  (_sp),y     ;Fehlernummer abspeichern.
         ldy  #7
         lda  (_sp),y     ;Die Adresse der Variablen
         sta  _t+1        ;fsize bestimmen und in _t+1
         iny              ;und _t+2 ablegen.
         lda  (_sp),y
         sta  _t+2
         iny
         lda  (_sp),y     ;Die Adresse der Variablen
         sta  _t+3        ;vsize bestimmen und in _t+3
         iny              ;und _t+4 ablegen.
         lda  (_sp),y
         sta  _t+4
         ldy  #Ø
         lda  blused      ;Den Wert von Blocks Used in
         sta  (_t+1),y    ;die Variable fsize kopieren
         iny
         lda  blused+1
         sta  (_t+1),y
         ldy  #Ø
         lda  auxtyp      ;Den Wert von Aux.Type, der
         sta  (_t+3),y    ;in diesem Fall die Gesamt-
         iny              ;speicherkapazität der Disk
         lda  auxtyp+1    ;angibt, in die Variable
         sta  (_t+3),y    ;vsize kopieren.
         jmp  endinfo
;
```

```
gfinfo   db $Øa           ;Get File Info: 1Ø Parmeter
         dw fname         ;Adresse des Volume-Namens
         db $ØØ           ;Zugriffsbefugnis
         db $ØØ           ;Filetype
auxtyp   dw $ØØØØ         ;Auxiliary Type
         db $ØØ           ;Speichertyp
blused   dw $ØØØØ         ;Blocks used
         dw $ØØØØ         ;Modification date
         dw $ØØØØ         ;      "      time
         dw $ØØØØ         ;Creation date
         dw $ØØØØ         ;    "    time
;
fname    ds 17            ;Puffer für Volume-Name
;
endinfo  ldx  _t
#
end;

procedure prtmlierror (errcode:integer);
begin
write ('PRODOS-FEHLER $');
#a
         stx  _t          ;X-Register-Inhalt retten
         ldy  #5          ;Y zeigt auf Fehler-Nummer
         lda  (_sp),y     ;A enthält Fehler-Nummer
         jsr  prbyte      ;Akku sedezimal ausgeben.
         lda  #$8d        ;Carriage Return ausgeben.
         jsr  cout
         ldx  _t          ;X-Register-Inhalt zurück
#
end;
```

Ist „Sprache“ eine Metapher?

Kritische Betrachtungen zum Begriff der Programmiersprache

von Ulrich Stiehl

Man-machine identity is achieved not by attributing human attributes to the machine, but by attributing mechanical limitations to man (Mortimer Taube).

1. Animismen und Metaphern

„Mit der ersten *Sprache* erlernt man nicht nur ein Vokabular und eine Grammatik, sondern man erschließt sich eine Gedankenwelt.“ Würde man nicht wissen, daß dieser Satz in einem Informatik-Lehrbuch steht (N. Wirth, Systematisches Programmieren, 5.Aufl., Stuttgart 1985, S.8), so müßte man meinen, daß von Deutsch oder Englisch und nicht von Algol oder Pascal die Rede ist.
Etwas später lesen wir in bezug auf den Prozessor: „Im Zentrum steht der Begriff der *Handlung*. Eine Handlung sei ein Vorgang, der in einer endlichen Zeitspanne abläuft und eine beabsichtigte und wohldefinierte Auswirkung – einen Effekt – hat“ (S. 13). Ferner heißt es, daß „die Maschine ein genau festgesetztes, endliches Befehlsrepertoire besitzt und die angenommenen *Befehlsfolgen* mit absolutem Gehorsam und totaler Kritiklosigkeit ausführt.“

Woher rührt die Selbstverständlichkeit, mit der Informatiker Assembler, BASIC und Pascal als „Sprachen“ bezeichnen? Und wie ist es möglich, daß Wissenschaftler den Computer für „handlungsfähig“ und „gehorsam“ erklären? Liegt dieser Terminologie nicht vielmehr eine längst überwunden geglaubte animistische, also alles beseelende Denkweise zugrunde, wonach Steine sprechen und Bäume handeln können? Wir pflegen herablassend zu lächeln, wenn ein von der Zivilisation bislang unberührter Ureinwohner vom Rundfunkgerät als „dem Stein, der spricht“ oder vom Roboter als „dem Gott aus Eisen“ redet. Aber müssen wir uns nicht selbst belächeln, wenn wir ähnlich naiv in bezug auf Computer von „den die Gedankenwelt erschließenden Sprachen“ oder von „gehorsamen Sklaven“ reden?

Im Alltagsleben begegnet man oft diesem anthropomorphisierenden (vermenschlichenden) Verhalten: Wer hätte nicht schon einmal mit seinem Auto „gesprochen“, wenn es den „Gaspedal-Befehl“ nicht „gehorsam“ ausführte? Und so nimmt es uns deshalb nicht wunder, daß sich in die Computerterminologie einige Wörter eingeschlichen haben, die eigentlich ganz andere Dinge bezeichnen. Ein Wort kann nämlich nicht nur eine wörtliche oder eigentliche, sondern auch eine uneigentliche, übertragene oder metaphorische Bedeutung haben. Dabei sind Metaphern keineswegs nur eine Domäne der Poesie, die „in blumiger Sprache“ Gedanken und Empfindungen zum Ausdruck bringt. Vielmehr ist man selbst in den exakten Wissenschaften geneigt, *zunächst* Metaphern zu gebrauchen, wenn neuartige Phänomene beschrieben werden sollen, die durch das bisherige Vokabular noch nicht abgedeckt werden. Die Bequemlichkeit darf jedoch nicht so weit gehen, daß man alles, was vier Beine hat und hinten wackelt, als „Wau-wau“ bezeichnet.

Nachfolgend soll deshalb untersucht werden, ob und inwieweit der Begriff der Sprache in der Informatik seine Berechtigung hat.

2. Lexikalische Definition

Wenn man das sechsbändige „Große Wörterbuch der deutschen Sprache“ (Mannheim 1981) unter dem Begriff „Sprache“ aufschlägt, so findet man als Erklärungen
(a) „das Sprechen“,
das unter dem Stichwort „sprechen“ tautologisch auf die Sprache zurückweist, sowie
(b) „Sprachsystem“,
das unter dem Stichwort „Sprachsystem“ als „System aus in gleicher Weise immer wieder vorkommenden und sich wiederholenden *sprachlichen* Elementen...“ definiert wird, wobei man unter dem Stichwort
(c) „sprachlich“
die Erklärung „die Sprache betreffend“ vorfindet, die prompt zu einem Circulus vitiosus führt:
(d) „Sprache ist ein System aus die Sprache betreffenden Elementen“.

Ergänzend sei vermerkt, daß das deutsche Wort „Sprache“ etymologisch von „sprechen“ (= Laute äußern) kommt, während das englische „language“ und das französische „langue“ vom lateinischen „lingua“ (= Zunge) herrühren.

3. Linguistische Definition

Linguisten sind sich keineswegs einig, wenn das Wesen der Sprache erklärt werden soll:
E. Sapir: „Sprache ist eine ausschließlich

dem Menschen eigene, nicht im Instinkt wurzelnde *Methode* zur Übermittlung von Gedanken, Gefühlen und Wünschen mittels eines Systems von frei geschaffenen Symbolen". Kommentar: Diese Definition beschränkt die Sprache auf den Menschen. Danach haben Tiere und Maschinen, z.B. Computer, keine Sprache.
B. Bloch: „Sprache ist ein *System* willkürlicher Lautsymbole, mit deren Hilfe eine soziale Gruppe gemeinsam handelt." Kommentar: Der Begriff „soziale Gruppe" könnte sich auch auf eine Gruppe von nicht-menschlichen Lebewesen, z.B. eine Horde von Affen, jedoch in keinem Fall auf eine „Horde von Robotern" beziehen.
R.A. Hall: „Sprache ist eine *Institution*, mit deren Hilfe Menschen miteinander kommunizieren und unter Verwendung gewohnheitsmäßig benutzter, oral-auditiver, willkürlicher Symbole in Interaktion treten." Kommentar: Auch hier wird der anthropologische Aspekt unterstrichen.
N. Chomsky: „Sprache ist eine endliche oder unendliche *Menge* von Sätzen, jeder endlich in seiner Länge und konstruiert aus einer endlichen Menge von Elementen." Kommentar: Dies ist eine jener Leerdefinitionen, wie man sie heute so sehr schätzt. Der Wert der Erklärung hängt vom Begriff des Satzes ab, der als eine „endliche Menge von Elementen" definiert wird, was indessen für alle realen „Mannigfaltigkeiten" auf dieser unserer Erde zutrifft (vgl. G. Cantor, Über die elementare Frage der Mannigfaltigkeitslehre, 1890).
Interessant sind die Oberbegriffe für Sprache, nämlich Methode, System, Institution und Menge. Daneben finden wir in anderen Quellen als Oberbegriffe Informationsträger, Kommunikationsmittel, Medium u.a.

4. Philosophische Definition

Der Mensch als „vernunftbegabtes Tier" (animal rationale) ist ein denkendes, fühlendes und wollendes Wesen. Folglich lassen sich rationale, emotionale und volitionale Bewußtseinsinhalte unterscheiden. Der Mensch ist darüber hinaus aber auch ein „geselliges Tier" (animal sociale), das ein Bedürfnis verspürt, seinen Mitmenschen *rationale* Gedanken und Erkenntnisse, *emotionale* Gefühle und Empfindungen sowie *volitionale* Empfehlungen und Wünsche mitzuteilen, zu vermitteln oder verständlich zu machen. Die *Verständigung* von Mensch zu Mensch ist jedoch *nicht direkt* von Bewußtsein zu Bewußtsein („von Hirn zu Hirn" → Telepathie), sondern *nur indirekt* über die Sensomotorik möglich (sensorisch = sinnliche Wahrnehmung betreffend; motorisch = muskelbetätigende Handlung betreffend).

Ein Mensch kann sich mit Hilfe von Gesichtsgebärden (Mimik = Gesichtssprache) oder Körpergebärden (Gestik = Körpersprache) verständigen, doch lassen sich damit nur rudimentäre Botschaften vermitteln. Als erheblich flexibler hat sich statt dessen die Verständigung mit Hilfe von „Zunge" (→ lingua) und „Mund" (→ mündlich) erwiesen, da sich mit diesen „Sprechwerkzeugen" etwa hundert verschiedene Laute erzeugen lassen, die zu praktisch beliebig vielen bedeutungtragenden Gebilden kombiniert werden können. Diese bedeutungtragenden Lautgebilde (Wörter usw.) sind indessen nicht willkürlich aus der Luft gegriffen worden, sondern quasi natürlich entstanden. Wir können deshalb definieren:

Sprache im engeren Sinne ist ein menschliches, natürliches, indirektes, mündliches Verständigungsmittel zur Mitteilung von Bewußtseinsinhalten.

5. Streichlösung

Es ist evident, daß bei dieser restriktiven Definition weder Gebärden*sprachen* noch Programmier*sprachen* unter den Begriff der Sprache fallen würden. Wir werden deshalb im folgenden untersuchen müssen, welche Begriffsmerkmale eliminiert oder gestrichen werden können (Streichlösung), damit *gerade noch* von Sprache geredet werden kann.

5.1. Mündliche Verständigung

Neben der mündlichen Verständigung (Sprecher/Hörer) gibt es auch die schriftliche Verständigung (Schreiber/Leser) sowie andere Formen der Kommunikation (Gestik, Mimik usw.), die sich aus der menschlichen Sensomotorik ergeben. Wir können deshalb das Begriffsmerkmal „mündlich" fallenlassen und unter dem Begriff der Sprache auch die anderen Verständigungsmittel subsumieren, die üblicherweise als Sprachen bezeichnet werden, z.B. Schriftsprache, Gebärdensprache, Blindensprache usw. Da sich Programmiersprachen der Schriftzeichen bedienen, können sie *insoweit* als Sprachen bezeichnet werden. Unsere Definition lautet jetzt: „Sprache ist ein menschliches, natürliches, indirektes Verständigungsmittel zur Mitteilung von Bewußtseinsinhalten." Programmiersprachen werden durch diese Definition jedoch noch nicht abgedeckt.

5.2. Indirekte Verständigung

Da es eine direkte Verständigung von Bewußtsein zu Bewußtsein mittels Telepathie („Gedankenlesen") offenbar nicht gibt, ist jede Kommunikation indirekt oder mittelbar, so daß wir den tautologischen Zusatz „indirekt" streichen können. Trotzdem ist die Unterscheidung zwischen direkter und indirekter Verständigung insofern relevant, als die Mittelbarkeit durch die modernen Kommunikationsmittel sehr komplexe Formen angenommen hat. Während die normale mündliche Sprache noch eine räumliche Anwesenheit von Sprecher und Hörer voraussetzt, ist dies bei der Schriftsprache, also z.B. beim Lesen von Büchern, bereits nicht mehr der Fall. Aber auch die mündliche Sprache kann mit Hilfe von technischen Einrichtungen „auf Distanz" gehen. So sind etwa beim Telefongespräch zwischen „Nachrichtenquelle" (Sprecher) und „Nachrichtensenke" (Hörer) der Sender (Codierer) und der Empfänger (Decodierer) zwischengeschaltet, die über den „Nachrichtenkanal" den chiffrierten Bewußtseinsinhalt übertragen.
Wir können festhalten, daß die Verwendung technischer Hilfsmittel, zu denen auch der Computer gehören kann, dem Begriff der Sprache keinen Abbruch tut. Wir sind damit bei folgender Definition angelangt: „Sprache ist ein menschliches, natürliches Verständigungsmittel zur Mitteilung von Bewußtseinsinhalten."

5.3. Natürliche Verständigung

Unsere Kultursprachen sind, wie der Name sagt, keine Naturprodukte, sondern von Menschen geschaffene Verständigungsmittel. Trotzdem pflegen wir beispielsweise Deutsch als eine natürliche Sprache zu bezeichnen, weil sie historisch gewachsen und nicht durch willkürliche Konvention „am grünen Tisch" entstanden ist. Ein typischer Indikator für eine „ausgewachsene" Sprache ist die Tatsache, daß keine primären Wörter mehr entstehen. Dies sind Wörter, die sich nicht mehr weiter zerlegen lassen, z.B. „Fisch" (von „piscis") oder „Tisch" (von „discus"). Obwohl wir neben „Tisch" und „Fisch" zahlreiche andere „euphonische", d.h. gut auszusprechende Wörter erzeugen könnten, z.B. „Risch" und „Lisch", ist dies aus psychologischen Gründen nicht üblich. Selbst bei Programmiersprachen machen wir die Erfahrung, daß völlig aus der Luft gegriffene Lautgebilde, etwa RISCH statt PRINT oder LISCH statt READ ungebräuchlich sind. Zwischen einer sog. natürlichen Sprache (z.B. Deutsch) und einer sog. künstlichen Sprache (z.B. Pascal) besteht also nur ein gradueller Unterschied, so daß wir bei unserer Sprachdefinition das Begriffsmerkmal „natürlich" vorbehaltlos streichen können. Damit lassen sich dann auch Programmiersprachen, die man auch als formalisierte Sprachen bezeichnet, *in dieser Hinsicht* Sprachen nennen. Wir sind damit bei

folgender Definition angelangt: „Sprache ist ein menschliches Verständigungsmittel zur Mitteilung von Bewußtseinsinhalten."

Wir möchten in diesem Zusammenhang darauf hinweisen, daß zwischen natürlichen Sprachen und natürlichen Zeichen ein Unterschied besteht. Von natürlichen Zeichen sprechen wir, wenn zwischen Zeichen und bezeichnetem Gegenstand ein sachlicher, z.B. kausaler (Rauch – Feuer; Symptom – Krankheit) oder abbildender (Paßbild – Person; Film – Wirklichkeit), Zusammenhang besteht. Eine auf Lauten basierende Sprache kann deshalb nur in bezug auf die Laute selbst natürliche Zeichen aufweisen (Onomatopöie = Lautmalerei, z.B. „Wau-wau" usw.). Da dies jedoch eher die Ausnahme denn die Regel ist, sind Sprachen in der Regel künstliche Zeichensysteme.

5.4. Menschliche Verständigung

Wir sind jetzt bei unserer Streichlösung an einem kritischen Punkt angelangt. Können wir auch auf das Begriffsmerkmal „menschlich" verzichten? Wenn wir berücksichtigen, daß auch Tiere eine Sprache haben, die mit der menschlichen Gestik und Mimik vergleichbar ist, so können wir das Wort „menschlich" streichen, weil unsere Sprachdefinition dann immer noch durch das Wort „Bewußtseinsinhalte" zusammengehalten wird: „Sprache ist ein Verständigungsmittel zur Mitteilung von Bewußtseinsinhalten." Wenn und soweit Tiere oder andere Wesen ein Bewußtsein haben, können deren Sprachen mit Recht als Sprachen bezeichnet werden. Doch haben auch Computer ein Bewußtsein?

5.5. Verständigungsmittel

Auf diesen Oberbegriff werden wir bei unserer Streichlösung nicht verzichten können, doch ist es an der Zeit, ihn näher zu erläutern. Unter Verständigungsmittel verstehen wir ein *System* (F. de Saussure: Langue) von sprachlichen Elementen (Wortschatz, Lexik) und sprachlichen Regeln (Grammatik, Syntax), mit deren Hilfe sich eine praktisch unbegrenzte Anzahl von sprachlichen Äußerungen bilden lassen. Programmiersprachen sind insoweit zweifellos als *Sprachsysteme* zu bezeichnen. Den Unterschied zwischen Lexik und Syntax wollen wir an einer künstlich konstruierten Sprache (= *Objektsprache*) verdeutlichen, die wir in einer hier vereinfachten Form mit Hilfe der natürlichen Sprache (= *Metasprache*) wie folgt beschreiben:
Die Lexik umfasse die „Personenbegriffe"
A = Anton; B = Berta; C = Caesar
sowie die „Relationenbegriffe"
S = sieht; T = trifft; U = untersucht.
In der Syntax legen wir fest, daß ein korrekter Satz aus einem der Elemente S, T oder U besteht, dem eines der Elemente A, B, C vorangeht und eines der Elemente A, B, C folgt, wobei das vorangehende Element nicht mit dem nachfolgenden Element identisch sein darf. Es lassen sich damit Sätze wie ASB ASC BSA BSC CSA CSB usw., nicht jedoch Sätze wie ASA ABC STU ABS usw. bilden. Wie ersichtlich, sind Programmiersprachen wie Assembler, BASIC, Pascal usw., aber auch mathematische und chemische Formelsprachen in dieser Hinsicht als Sprachen anzusehen. Und genau dieser Umstand, daß künstliche, symbolische oder formalisierte Sprachen eine Lexik und eine Syntax haben können, hat denn auch dazu geführt, daß man Programmiersprachen als Sprachen bezeichnet.

5.6. Mitteilung

Der klassische Fall der sprachlichen Kommunikation oder des Sprechens (F. de Sassure: Parole) ist das Gespräch zwischen zwei Menschen, die als Gesprächspartner abwechselnd als Sprecher und Hörer fungieren. Eine Kommunikation kann jedoch auch in nur *einer* Richtung verlaufen, wie dies für Vorträge, Reden und Rundfunksendungen typisch ist. All diesen Kommunikationsformen ist gemeinsam, daß sich an beiden Enden des „Kommunikationskanals" mindestens *ein* bewußtes Wesen befindet. Zwar sind auch „Predigten vor leeren Bänken" denkbar, und aus der Literaturgeschichte wissen wir, daß manche Werke (die dann postum erschienen sind) von Autoren „nur für sie selbst" geschrieben wurden. Typisch sind diese Kommunikationen jedoch nicht, denn schließlich steckt im Begriff der „Mit-Teilung", des „Gemeinsam-Machens" (communicatio → communis), daß die Nachricht nicht nur ausgesendet, sondern auch empfangen wird.
Wer sind jedoch die Gesprächspartner bei einer Programmiersprache? Wenn wir die Möglichkeit von uns weisen, daß sich zwei Programmierer *in* Pascal oder *in* BASIC (→ Objektsprache) unterhalten (wohl aber *über* Pascal oder *über* BASIC → Metasprache), so verbleibt eigentlich nur noch die Möglichkeit, daß ein Programmierer mit seinem Computer in einer Programmiersprache kommuniziert. Damit haben wir jedoch nicht mehr eine Mensch-Mensch-, sondern eine Mensch-Maschine-Kommunikation. Dies verträgt sich jedoch nicht mit der Mitteilung von Bewußtseinsinhalten. Bewußtsein ist das Mit-Wissen (con-scientia) des denkenden, fühlenden und wollenden Ichs. Ohne Bewußtsein gibt es auch keine Kommunikation, denn solange Kommunikator und Kommunikant kein Bewußtsein haben, sind „Wörter" und „Sätze" nur physikalische Erscheinungen, also Schallwellen im Raum oder Farbpigmente auf dem Papier. Um den Begriff der Sprache für die Informatik dennoch retten zu können, müßten wir entweder den Computer als denkendes, fühlendes und wollendes Wesen begreifen, womit wir wieder beim anthropomorphisierenden Weltbild des Urmenschen angelangt wären, oder wir müßten auch unbewußte Kommunikationen zulassen, womit beispielsweise die Erythrozyten mit den Leukozyten in der Blutbahn oder der 6502- mit dem Z80-Prozessor im Computer kommunizieren würden. Dies führt aber zu einer Aufweichung des Kommunikationsbegriffs selbst, weil jetzt alles kommuniziert, was korreliert oder in Wechselbeziehung steht. In diesem Sinne schreibt beispielsweise L.G.Tesler: „Einige Sprachen verleihen dem Computer sogar gleichsam eine gespaltene Persönlichkeit: Er wir zu einem Verbund unabhängiger Stellen, die ihre eigenen Berechnungen vornehmen und untereinander kommunizieren."

6. Semiotik

Die von Ch. S. Pierce begründete Semiotik oder allgemeine Zeichenlehre unterscheidet syntaktische (Zeichen – Zeichen), semantische (Zeichen – Bedeutung) und pragmatische (Zeichen – Benutzer) Aspekte des Zeichens. Dies bedeutet in bezug auf Sprachen folgendes:

– Unter Syntax verstehen wir generell die Beziehungen zwischen Zeichen oder speziell die „Zusammenstellung" (syn-tas-sein) der Wörter im Satz. So ergeben beispielsweise die Wörter „Hans", „Fritz" und „sieht" je nach Wortstellung („Hans sieht Fritz" und „Fritz sieht Hans") einen unterschiedlichen Sinn. Ähnliches gilt für Programmiersprachen, doch ist deren Syntax erheblich starrer als die der natürlichen Sprachen. Der syntaktische Aspekt von Programmiersprachen ist damit unbestritten.
– Schwieriger wird es dagegen, wenn wir den semantischen (bedeutungsmäßigen) Aspekt unter die Lupe nehmen. Was bedeutet Bedeutung? Wenn Sie einen Baum sehen, so haben Sie eine Wahrnehmung dieses Baumes in Ihrem Bewußtsein. Wenn Sie eine schwarze Farbe sehen, so haben Sie ebenfalls eine bestimmte Wahrnehmung in ihrem Bewußtsein, nämlich die der schwarzen Farbe. Wenn Sie jedoch das folgende, aus schwarzer Farbe bestehende Gebilde
BAUM

PEEKER Börse

Gelegenheitsanzeigen / Kleinanzeigen

Sie können unter dieser Rubrik zu einem besonders günstigen Preis

- Ihre Hardware und Software verkaufen
- Ihre Hard- und Software suchen
- Kontakte knüpfen und vieles mehr

Musteranzeige privat (nicht gewerblich)

1 Druckzeile à 32 Buchstaben nur DM 5,50 zuzügl. ges. MwSt. Mindestens 2 Druckzeilen

Beispiel:

> Verkaufe neuwertigen Typenraddrucker mit Apple-Interface. Preis auf Anfrage. Tel. 007

nur DM 18,81 inkl. MwSt.

Musteranzeige gewerblich

1 Druckzeile à 32 Buchstaben nur DM 11,– zuzügl. ges. MwSt. Mindestens 2 Druckzeilen

Beispiel:

> Neu im Angebot: Professionelle, separate Tastatur für Apple II plus 16 Funktionstasten und separatem Ziffernblock. Fa. Keyboard & Co.

nur DM 62,70 inkl. MwSt.

PEEKER

Peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

Bitte veröffentlichen Sie in der nächsterreichbaren Ausgabe nachstehenden Text unter folgender Rubrik:

☐ suche Hardware ☐ suche Software ☐ Verschiedenes ☐ gewerblich
☐ biete Hardware ☐ biete Software ☐ Chiffre ☐ nicht gewerblich

Bitte den Text mit Schreibmaschine oder in Druckbuchstaben ausfüllen

Jeweils 32 Buchstaben pro Zeile – einschl. Satzzeichen und Wortzwischenräume. Bitte Absender nicht vergessen. Mindestens 2 Zeilen (1 Druckzeile à 32 Buchstaben DM 5,50 nicht gewerblich, DM 11,– gewerblich + MwSt.) – Chiffregebühr DM 6,– + MwSt.

PEEKER

Produkt-Karte

Zu der in **Peeker**, Heft _____, Seite _____ erschienenen

Anzeige über _____

_____

bitte ich um detaillierte Information.

Ich wünsche ☐ Prospekt, Datenblatt ☐ Preisliste ☐ schriftliches Angebot ☐ tel. Rückruf

Menge	Produkt und Bestellnummer	à DM	gesamt DM

gebe ich nebenstehende Bestellung unter Anerkennung Ihrer in der Anzeige genannten Liefer- und Zahlungsbedingungen auf.

_____

Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

PEEKER

Info-Karte

_____

_____

_____

_____

_____

_____

_____

Ich interessiere mich für Beiträge über

☐ Apple ☐ IBM ☐ Atari ☐ _____

Peeker-Börse

Vorname, Name

Firma

Straße

Wohnort

PLZ/Ort

Bitte veröffentlichen Sie den umstehenden Text von _____ Zeilen à _____ DM in der nächsterreichbaren Ausgabe vom **Peeker**

Bei Angeboten: Ich bestätige, daß ich alle Rechte an den angebotenen Sachen besitze

Datum Unterschrift

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Anschrift der Firma angeben, bei der Sie bestellen bzw. von der Sie Informationen wünschen

Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Bitte freimachen

POSTKARTE

Peeker-Börse
Anzeigen-Service

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

Bitte freimachen

POSTKARTE

Inserent

Straße

PLZ/Ort

Bitte freimachen

POSTKARTE

Peeker
Redaktion

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

Produkt-Karte

Wünschen Sie weitere Informationen zu einer der im Heft erschienenen Anzeigen?

Nichts einfacher als das. Produkt-Karte ausfüllen, frankieren und an den Inserenten (nicht an die Peeker-Redaktion) senden.

Bitte aber nicht vergessen:
Kreuzen Sie an, welchen Informationswunsch Sie haben.

Damit erleichtern Sie dem Hersteller eine gezielte Beantwortung Ihrer Anfrage

Zum Schluß tragen Sie auf der Rückseite die genaue Anschrift des Inserenten und Ihren Absender ein.

PEEKER

sehen, taucht in Ihrem Bewußtsein neben der Wahrnehmung dieses schwarzen Gebildes zusätzlich die Vorstellung eines Baumes auf. Diese Vorstellung, die mehr oder weniger abstrakt sein kann, nennen wir den Begriff des Baumes oder allgemein die Bedeutung eines Zeichens. Ähnliches gilt auch für die Bedeutungsgebilde in Programmen, z.B. PRINT. Doch gibt es bei Programmiersprachen im Gegensatz zu natürlichen Sprachen einen kleinen, aber sehr wichtigen Unterschied: Die Bedeutung des Zeichens PRINT taucht als Vorstellung nur in *Ihrem* Bewußtsein auf und nicht etwa in dem Bewußtsein Ihres Computers, auch wenn Sie ihn noch so oft mit PRINT „ansprechen".

– Der pragmatische (sprechhandelnde) Aspekt betrifft die Relation zwischen Zeichen und Benutzer. Bei einer natürlichen Kommunikation gibt es immer zwei gleichwertige Benutzer, nämlich Sprecher und Hörer, Schreiber und Leser, Kommunikator und Kommunikant, die bezüglich des mitgeteilten Zeichens zumindest partiell deckungsgleiche Vorstellungsinhalte haben müssen, weil sie sonst aneinander vorbeireden würden, und dies ist bekanntlich keine Kommunikation. Letzteres gilt auch für die pragmatische Mensch-Computer-Beziehung, denn der Computer versteht nicht die Bedeutung von PRINT. Daß er trotzdem auf PRINT reagiert, wie auch ein Zigarettenautomat auf ein Markstück reagiert, hat mit Kommunikation nichts zu tun.

7. Handlung

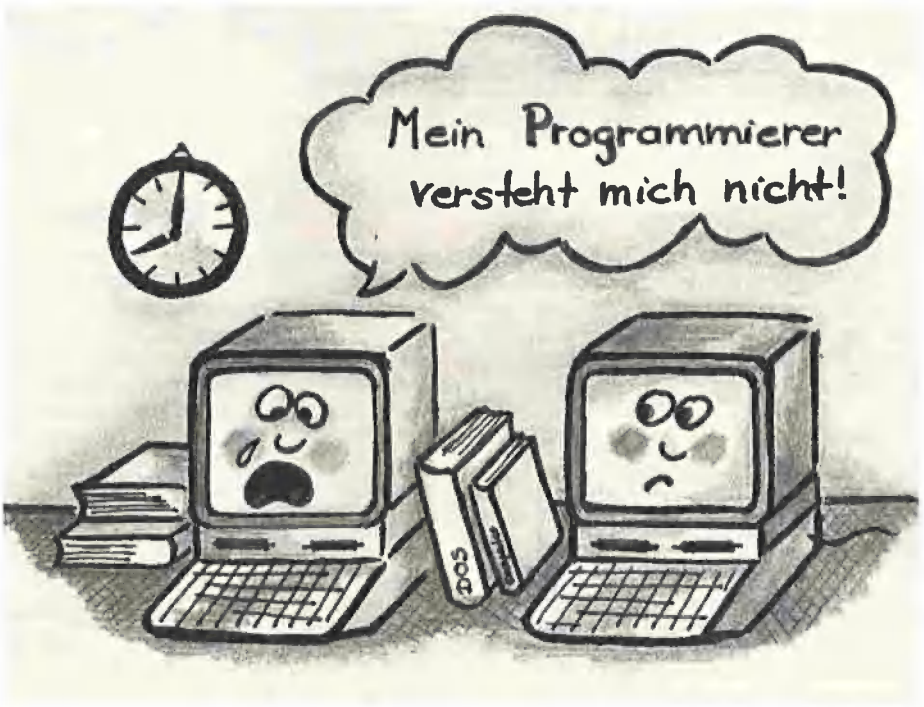
Zum Abschluß unserer Erörterungen wollen wir noch kurz auf den Begriff der Handlung eingehen. Programmiersprachen werden gerne als *Befehlssprachen* erklärt, denn „programmieren" (griechisch: prographein) heißt vor-schreiben, anordnen, befehlen. Ein Programm wäre damit eine Folge von Befehlen an den Computer als Befehlsempfänger. Befehle wie CLEARSCREEN (Lösche den Bildschirm!) oder PRINT 7 (Zeige die Zahl 7 an!) lassen sich durch Befehlssätze (Imperative) umschreiben. In Programmen tauchen zwar auch Aussagesätze auf, doch niemals isoliert, sondern stets als Bestandteile von Befehlssätzen (IF Aussagesatz, UNTIL Aussagesatz, WHILE Aussagesatz usw.).

Eine Handlung kann als bewußte, willentliche Tätigkeit definiert werden. „Schon Aristoteles betont als Kriterien des selbsttätigen, ungezwungen freien Handelns die Bewußtheit und die Gewolltheit. Beide sind grundsätzlich ichhaft, ichzugehörig und tätigkeitsverbunden,... Selbstbestimmtheiten des selbst- und gegenstandsbewußten Willenswesens Ich" (B. von Brandenstein).
Dabei lassen sich die Handlungen *anderer* Menschen – graduell abgestuft – durch Empfehlungen, Wünsche und Befehle beeinflussen. Es gibt Teilbereiche des menschlichen Zusammenlebens, die überwiegend durch Befehlssprachen bestimmt sind (Kasernenhofsprache). Wenn wir eine Trennung zwischen dem RAM-Speicher, in dem das Programm als eine Folge von Befehlen abgelegt ist, und dem eigentlichen Prozessor als Befehlsempfänger machen, so können wir sagen, daß Prozessoren die in dem Programm fixierten Befehle „mit absolutem Gehorsam" (N. Wirth) ausführen. All dies muß jedoch in einem metaphorischen Sinne verstanden werden, denn den Computer im sprichwörtlichen Sinne als handelndes und gehorsames Wesen aufzufassen, scheint uns verfehlt zu sein. Konsequenterweise müßten wir dann nämlich auch von einem „gehorsamen Stein" sprechen, weil er immer „gehorsam" zur Erde fällt, sobald wir ihm durch das Öffnen der Hand „befehlen", nach unten zu fallen.

8. Ausblick

Im Laufe der nächsten Jahrzehnte werden wir Bekanntschaft mit Robotern machen, die in unserer Muttersprache „antworten" werden, wenn man sie „anspricht", und die „handeln" werden, wenn man ihnen „Befehle" erteilt. In „geisterhaften" Fabriken werden sie sich miteinander „unterhalten", um für eine reibungslose „Erfüllung" der ihnen übertragenen „Aufgaben" zu sorgen. Der alte Menschheitstraum des künstlichen Menschen (= Homunkulus) wird dann in greifbare Nähe rücken. Es läßt sich unschwer prophezeien, daß man diesen Homunkuli auch „menschliche" Namen gegeben wird, auf die sie dann „hören" werden. Sollten sie auch im häuslichen Bereich als dienstbare „Geister" Einzug halten, so werden findige Programmierer dafür sorgen, daß auch das Menschlich-Allzumenschliche gebührend berücksichtigt wird mit der Folge, daß manche den Unterschied zwischen Mensch und Maschine nicht mehr so genau nehmen werden.

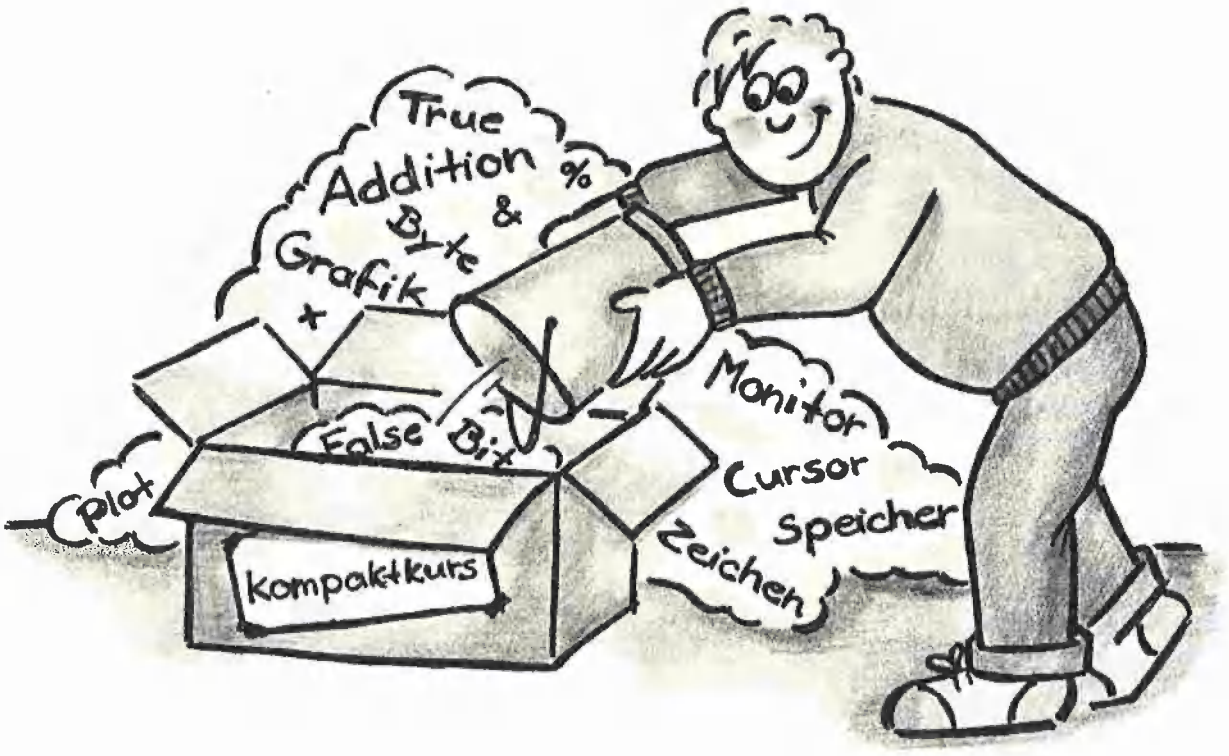
Der Sinn dieses Beitrages soll nicht darin bestehen, daß wir in Zukunft Begriffe wie „Sprache" oder „Handlung" aus unserem informationstechnischen Vokabular streichen oder fortan nur noch in Anführungszeichen setzen. Es würde schon genügen, wenn wir uns gelegentlich auf die ursprünglichen Bedeutungen zurückbesinnen und mithin die Tatsache nicht vergessen, daß wir im Grunde genommen in Metaphern reden. So schreibt denn auch N. Wirth, indem er von seinen eingangs zitierten Aussagen abrückt, an anderer Stelle: „Um die abstrakten Datenstrukturen und Algorithmen eines Programms formulieren zu können, ist eine formale Notation unumgänglich... Solche formalen Notationen heißen Programmiersprachen, obwohl diese Bezeichnung irreführend ist. Denn Programmieren hat wenig gemein mit dem Schreiben in einer Sprache" (Spektrum der Wissenschaft, Sonderheft Computersoftware, 1986, S. 16).



GFABASIC-Kompaktkurs

Teil 1: Grundlagen und Mathematik

von Ulrich Stiehl



Vorbemerkung: Dieser Kompaktkurs wendet sich überwiegend an Atari-Anfänger, die bislang noch nie programmiert haben. Mit seinen rund 300 Befehlen dürfte GFABASIC jedoch auch fortgeschrittenen BASIC-Programmierern viel Neues bieten.

1. Allgemeines

GFABASIC ist zur Zeit wahrscheinlich das leistungsfähigste 68000-BASIC, das andere Dialekte, z.B. das beim Atari mitgelieferte ST-BASIC von Digital Research oder das für den Macintosh gesondert erhältliche Microsoft-BASIC nachgerade als Amateurprodukte erscheinen läßt. GFABASIC kann mit einem beachtlichen Programmierkomfort, einer ungewöhnlich hohen Ausführungsgeschwindigkeit, einem sehr großen Befehlsumfang sowie leistungsfähigen pascal-ähnlichen Programmstrukturen aufwarten. Neben dem Interpreter ist auch ein Compiler erhältlich, doch ist der Interpreter derart effizient, daß eine Compilierung oft entbehrlich ist. Für kleinere bis mittelgroße Programmprojekte ist GFABASIC deshalb zur Zeit anderen Hochsprachen (Pascal, C usw.) spürbar überlegen. Darüber hinaus kann man bei Bedarf Assemblerroutinen einbinden, wodurch sich weitere Einsatzmöglichkeiten ergeben.

1.1. Interpreter und Compiler

Ein **Befehl** (eine Instruktion, ein Kommando) veranlaßt den Computer zur Durchführung einer bestimmten Tätigkeit (Tastaturabfrage, Rechenoperation, Bildschirmausgabe usw.), wobei eine Folge von Befehlen bzw. Befehlszeilen als **Programm** bezeichnet wird. Zur Erstellung von Programmen dienen symbolische Befehlsnotationen, die als **Programmiersprachen** bezeichnet werden. Ein Mikroprozessor, z.B. der 68000, beherrscht nur sehr elementare binäre Befehle, z.B. das Übertragen eines Worts (16stelligen Bitmusters) von einer Speicherstelle in die andere oder das Setzen und Löschen einzelner Bits. Folglich ist die **Maschinensprache** als die Summe dieser binären Befehle die einzige Sprache, die der Mikroprozessor „versteht". Die Eingabe von Bitmustern, z.B. „01001110 01110101" usw., kann jedoch keinem Programmierer zugemutet werden. Folglich wurden **Assemblersprachen** entwickelt, die maschinensprachliche Instruktionen durch

Befehlskürzel (mnemonische Kurzwörter) ausdrücken. Beispielsweise steht dann „RTS“ für das obige Bitmuster. Nun haben jedoch die Assemblerbefehle mit den Aufgaben, die man mit dem Mikrocomputer lösen will, kaum etwas gemein. So muß man beispielsweise für das Plotten eines Kreises ein Programm mit Tausenden von Assemblerbefehlen schreiben. Deshalb wurden **Hochsprachen** geschaffen, die sich gewissermaßen aus Makrobefehlen zusammensetzen, z.B. „Circle x,y,r“ = „Zeichne einen Kreis mit dem Radius r an der xy-Koordinate“. Da der Computer jedoch mit dem Wort „Circle“ nichts anfangen kann, muß der Kreisbefehl zunächst in den **Objektcode**, d.h. in die entsprechende maschinensprachliche Befehlsfolge, umgewandelt werden, bevor der Kreis dann tatsächlich am Bildschirm erscheinen kann. Ein Programm besteht jedoch normalerweise nicht aus einem einzigen Befehl („Einzeiler“), sondern aus einer Folge von Befehlszeilen, dem **Quelltext**. Hier bieten sich nun zwei Möglichkeiten an: Wenn der Quelltext *insgesamt* in den Objektcode umgewandelt wird, sprechen wir von einer **Compilersprache**. Wenn umgekehrt immer nur diejenige Programmzeile, die gerade abgearbeitet werden soll, in eine kurze maschinensprachliche Befehlsfolge umgewandelt wird, sprechen wir von einer **Interpretersprache**.

BASIC (Beginner's All-purpose Symbolic Instruction Code) gehört also zu den Interpretersprachen. Dies hat Vor- und Nachteile:
Vorteil: Befehle, die man im sog. Direkt-Modus eingibt, werden unmittelbar ausgeführt (ähnlich wie bei einem Taschenrechner). Wenn man z.B. über die Tastatur „Print 7 * 8“ eingibt, so wird sofort das Ergebnis, also hier „56“, angezeigt. Unbekannte Befehle kann man also mühelos ausprobieren. Ferner können Programme, die gerade in den Editor eingegeben worden sind, ohne Zeitverlust mit „Run“ gestartet werden. Bei Compilersprachen ist hierzu erst eine oft mehrere Minuten dauernde Compilierung erforderlich.
Nachteil: Da bei Interpretersprachen während des laufenden Programms jeder Befehl einzeln interpretiert, d.h. in Maschinensprache umgewandelt, werden muß, ist BASIC insbesondere bei Programmschleifen relativ langsam. Deshalb sind zu den meisten BASIC-Dialekten zusätzlich Compiler verfügbar, die den BASIC-Quelltext *insgesamt* in einen BASIC-Objektcode konvertieren. Die Entwicklung eines Programms erfolgt dann mit dem BASIC-Interpreter, und nach Abschluß aller Testläufe wird das fertige Programm nur ein einziges Mal compiliert.

1.2. Direkt- und Run-Modus

Die klassischen BASIC-Dialekte verfügen über einen *gemeinsamen* Bildschirm für Direkt-, Edit- und Run-Modus:
● Die Eingabe von z.B.
Print “Hallo“
bewirkt die direkte Ausführung des Befehls, also hier die Ausgabe von „Hallo“ auf dem Bildschirm (Direkt-Modus).
● Die Eingabe von z.B.
10 Print “Hallo“
bewirkt die Übernahme dieses Befehls in den Programmspeicher (Edit-Modus). An der führenden *Zeilennummer*, hier 10, erkennt der traditionelle BASIC-Interpreter, daß nicht der Direkt-, sondern der Edit-Modus gemeint ist.
● Die Eingabe von z.B.
Run
bewirkt den Start des zuvor in den Speicher eingegebenen Programms (Run-Modus).

GFABASIC macht im Gegensatz zu den traditonellen BASIC-Dialekten, aber im Einklang mit den meisten anderen Hochsprachen, von dem Zeilennummernkonzept keinen Gebrauch. Folglich muß zwischen Edit- und Direkt/Run-Bildschirm unterschieden werden:
● Der Edit-Bildschirm ist an einer Kopfleiste ersichtlich (s. **Abb. 1**), die in 2 Zeilen mit je 10 Befehlsfeldern eingeteilt ist, die den Funktionstasten Shift-F1 bis Shift-F10 (obere Zeile) sowie F1 bis F10 (untere Zeile) zugeordnet sind. Alle im Edit-Modus eingegebenen Befehlszeilen werden in den Programmspeicher (= Quelltext) übernommen. Die Funktionstasten sind nur im Edit-Modus wirksam.
● Wenn man auf die Esc-Taste tippt (ohne Return), so gelangt man jederzeit vom Editor in den Direkt/Run-Bildschirm (s. **Abb. 2**), der über keine Kopfleiste verfügt. Hier kann man entweder Direktbefehle über die Tastatur eingeben, die dann sofort ausgeführt werden, oder ein im Speicher befindliches Programm mit „Run“ starten. Mit Esc (gefolgt von Return) gelangt man in den Edit-Modus zurück.
Merke: Mit Esc (oder Shift F9) gelangt man vom Edit- in den Direkt-Modus und umgekehrt mit Esc-Return (oder „Edit“) vom Direkt- in den Edit-Modus.
Anmerkung: Andere BASIC-Interpreter, z.B. ST-BASIC für den Atari oder Microsoft-BASIC für den Macintosh machen von der Fenstertechnik Gebrauch und verfügen dann über Command-, List- und Output-Fenster. Vorteil: Man sieht alles auf einen Blick. Nachteil: Bildschirmsteuerung frustrierend langsam.

2. Editor

2.1. Laden des Interpreters

Nachdem man die Atari-Systemdiskette gebootet hat, kann man GFABASIC wahlweise über das GEM-Desktop oder über eine DOS-Shell starten:
GEM-Desktop: GFABASIC-Diskette einlegen, entsprechendes Diskettenbild, z.B. Diskstation A, öffnen und dann zweimal auf GFABASIC.PRG-Ikone klicken. Wenn Sie später GFABASIC über den Quit-Befehl verlassen, gelangen Sie automatisch in das GEM-Desktop zurück.
DOS-Shell: Es gibt verschiedene DOS-Shell-Programme, die meistens COMMAND.PRG heißen. Aus einem solchen Command-Programm heraus starten Sie GFABASIC durch Eingabe des Namens über die Tastatur, z.B. A:GFABASIC.PRG oder kürzer GFABASIC. Wenn Sie später GFABASIC über den Quit-Befehl verlassen, gelangen Sie automatisch in das Command-Programm zurück.

2.2. Befehlszeile

Nach dem Start von GFABASIC befinden Sie sich zunächst im Edit-Bildschirm. Geben Sie nun über die Tastatur eine Zeile ein, z.B.
Print “Hallo“

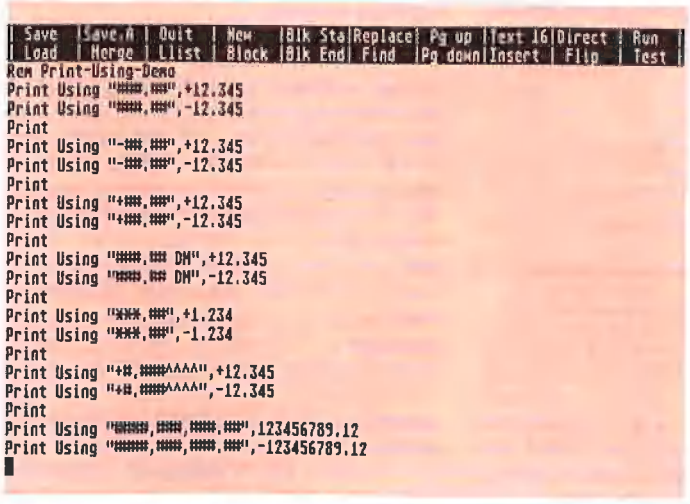


Abb. 1: Edit-Bildschirm

Abb. 2: Run-Bildschirm

Befehlswörter wie Print können Sie wahlweise in Groß- oder Kleinbuchstaben eingeben, z.B. Print, PRINT, print usw. Für fast alle Befehle gibt es Abkürzungen, die Sie dem GFABASIC-Handbuch entnehmen können. Für Print würde z.B. P oder p genügen. Da man sich jedoch die Abkürzungen nicht alle merken kann, werden sie in diesem Kurs nicht verwendet. Drei Sonderzeichen sollen jedoch gesondert aufgeführt werden:

? steht für Print,
@ steht für Gosub,
' steht für Rem.

Zwischen Befehlswörtern (z.B. Print) und den folgenden Parametern (z.B. „Hallo") brauchen Sie *meistens* keine Leerzeichen zu tippen, denn die eingegebene Befehlszeile wird ohnehin vom Editor nachträglich formatiert. Statt
Print "Hallo"
könnte man also auch
Print"Hallo" oder
?"Hallo"
eingeben.
Bei Funktionen muß sogar die Leertaste zwischen Befehlswort und eingeklammerten Parametern entfallen, also
Print Sin(30) richtig
Print Sin (30) falsch.

2.3. Programmaufbau

Ein GFABASIC-Quelltext besteht aus einer Folge von Befehlszeilen, die jeweils durch Return abgeschlossen werden müssen. Dabei kann eine Zeile mehr als 80 Zeichen einnehmen (horizontales Scrollen bis maximal 255 Zeichen), doch darf sie nur einen einzigen Befehl enthalten. Eine Ausnahme stellt der hinzugefügte Kommentar dar, der mit „!" anstelle von Rem oder „'" eingeleitet wird, z.B.
Print "Hallo" ! Kommentar...
Anmerkung: Bei anderen BASIC-Dialekten kann man mehrere Befehle pro Zeile eingeben, die dann üblicherweise durch den Separator „:" getrennt werden. Der Grund, daß GFABASIC nur 1-Befehl-Zeilen zuläßt, liegt darin, daß erstens bei Verwendung langer Variablennamen eine Bildschirmzeile ohnehin schnell gefüllt wird und zweitens bei Programmschleifen automatisch eingerückt wird. Dies wäre bei einer kompakten Notation in der Art anderer BASIC-Dialekte kaum möglich. Man vergleiche hierzu die nachfolgende Gegenüberstellung:

```
Rem Applesoft-BASIC
10 For I=1 to 10: For J=1 to 10
   Print I,J: Next J,I
Rem GFABASIC
For I=1 to 10
  For J=1 to 10
    Print I,J
  Next J
Next I
```

2.4. Editierbefehle

Aus der Befehlsleiste des Edit-Bildschirms sind 20 Kommandos ersichtlich, die durch Anklicken mit der Maus ausgelöst werden können:

● Über *Save* und *Load* kann man Programme mit Suffix „Bas" (im Binärformat) speichern bzw. laden. Mit *Save,A* und *Merge* werden Programme mit Suffix „Lst" (im ASCII-Format) gespeichert bzw. eingelesen. Beim Merge-Befehl wird der Textfile ab der momentanen Cursorposition in ein bereits im Speicher befindliches Programm eingefügt. Mit *Llist* wird das Programm ausgedruckt.

● Mit *Direct* (oder durch Drücken auf die Esc-Taste) kann man in den Direkt-Bildschirm überwechseln. Mit *Flip* kann man zwischen Edit- und Run-Bildschirm hin- und herschalten (Toggle-Schalter).

● *New* löscht das Programm im Speicher, *Test* prüft, ob alle Programmschleifen richtig aufgebaut sind, *Run* startet das Programm im Speicher. Mit *Quit* verläßt man den GFABASIC-Interpreter.

● Mit *Block Start* wird die Zeile, auf der der Cursor ruht, als Anfang eines Textblocks markiert. Mit *Block End* wird das Ende des markierten Textblocks festgelegt, der sich bis zu derjenigen Zeile erstreckt, *unter der* der Cursor steht, d.h. ausschließlich der momentanen Cursorzeile. Ein durch Block Start und Block End markierter Textblock wird punktschraffiert am Bildschirm angezeigt. Mit *Block* erscheint eine weitere Menüzeile mit diversen Blockoperationen, aus denen ausgewählt werden kann, und zwar
C: Copy = Duplizieren und somit Verdoppeln des markierten Textblocks vor diejenige Zeile, auf welcher der Cursor momentan ruht.
M: Move = Verschieben des Textblocks vor die Cursorzeile.
W: Write = Speichern eines Textblocks als Textfile mit Suffix „Lst" (Gegenstück zu Merge).
L: Llist = Ausdruck eines Textblocks.
S: Start = Cursor zum Block-Anfang.
E: End = Cursor zum Block-Ende.
Ctrl-D: Del = Löschen eines Textblocks.
H: Hide = Blockmarkierung aufheben.

● Mit *Find* wird die zu suchende Zeichenfolge festgelegt, die dann mit Ctrl-F gefunden werden kann. Mit *Replace* werden Such- und Ersetz-Zeichenfolge festgelegt. Mit Ctrl-R findet dann das Ersetzen statt.

● Mit *Insert* und *Overwrite* kann man zwischen dem Einfüge- und Überschreibmodus wechseln (Toggle-Schalter). Normalerweise befindet man sich im Einfügemodus.

● Mit *Text 16* und *Text 8* kann man zwischen großer und kleiner Textschrift umschalten (Toggle-Schalter). Normalerweise wird die Text-16-Schrift verwendet (25 Zeilen/Seite abzüglich der Menüleiste).

Man kann statt Maus und Menüleiste auch mit den entsprechenden Funktionstasten arbeiten (s.u.)

Cursorbewegung

Rechtspfeil	C. nach rechts
Linkspfeil	C. nach links
Hochpfeil	C. nach oben
Tiefpfeil	C. nach unten
Return	C. nach unten
Ctrl-Rechtspfeil	C. zum Zeilenende
Ctrl-Linkspfeil	C. zum Zeilenanfang
Ctrl-Hochpfeil	C. zum Bildschirmanfang
Ctrl-Tiefpfeil	C. zum Bildschirmende
Ctrl-Home	C. zum Programmanfang
Ctrl-Z	C. zum Programmende
Tab	C. 8 Stellen nach rechts
Ctrl-Tab	C. 8 Stellen nach links

Man beachte, daß man eine Programmzeile nicht nur mit Return, sondern auch mit Hochpfeil oder Tiefpfeil abschließen kann und daß in allen drei Fällen der Cursor nicht am Zeilenende stehen muß.

Delete/Insert/Find/Replace

Backspace-Taste	Zeichen rückwärts löschen
Del-Taste	Zeichen vorwärts löschen
Undo-Taste	Änderungen zurücknehmen
Ctrl-Del	Eine ganze Zeile löschen
Insert-Taste	Eine ganze Zeile einfügen
Ctrl-F	Find (vorher F6)
Ctrl-R	Replace (vorher Shift-F6)

Funktionstasten

F1	Load Program.Bas
Shift-F1	Save Program.Bas
F2	Merge Program.Lst
Shift-F2	Save Program.Lst
F3	Llist des Programms
Shift-F3	Quit oder System
F4	Blockoperation C = Copy (Duplizieren) M = Move (Verschieben) W = Write (Block.Lst) L = Llist (Block-Ausdruck) S = Cursor zum Block-Start E = Cursor zum Block-Ende H = Hide (Entmarkierung) Ctrl-D = Delete Block
Shift-F4	New
F5	Markierung Block End
Shift-F5	Markierung Block Start
F6	Find-String-Eingabe
Shift-F6	Replace-String-Eingabe
F7	wie Ctrl-Tiefpfeil
Shift-F7	wie Ctrl-Hochpfeil
F8	Insert/Overwrite-Toggle
Shift-F8	Text-8/16-Toggle
F9	Flip
Shift-F9	Direct
F10	Test
Shift-F10	Run

2.5. Direkte Systembefehle

Wer bislang mit anderen BASIC-Interpretern gearbeitet hat, kann diverse Befehle statt über die Edit-Menüleiste auch über den Run-Bildschirm eingeben. Nehmen wir an, wir hätten nach dem Start von GFABASIC, das uns zunächst in den Edit-Bildschirm führte, die Zeile
Print "Hallo"
eingegeben. Wenn wir nun auf die Esc-Taste tippen, wechseln wir in den Direkt-Bildschirm über. Wenn wir nun „Run" eingeben, wird der Einzeiler gestartet, der „Hallo" anzeigt und uns dann – leider – sofort wieder in den Edit-Bildschirm zurückführt. Also tippen wir erneut auf die Esc-Taste, um zum Run-Bildschirm zurückzukehren. Nunmehr können wir diverse System- und Diskettenbefehle über die Tastatur eingeben. Beispiele:
Save Demo
speichert den Einzeiler unter dem Namen „Demo.Bas" in Binärform auf Diskette.
List Demo
speichert den Einzeiler unter dem Namen „Demo.Lst" in ASCII-Form auf Diskette.
Files
zeigt das Ersatz-Inhaltsverzeichnis (Default-Directory) an.
Llist
druckt den Einzeiler aus (vorher Drucker einschalten).
Load Demo
lädt den Einzeiler wieder von Diskette.
Kill Demo.Lst
entfernt die ASCII-Datei Demo.Lst von der Diskette.
(Ein detaillierte Beschreibung der Diskettenbefehle folgt in einem späteren Teil dieses Kompaktkurses.)

Sie müssen selbst experimentieren und prüfen, ob Ihnen die Edit-Menüleiste- oder die Direkt-Tastatur-Befehle mehr zusagen. Beachten Sie jedoch, daß die Funktionstasten im Direkt-Modus nicht funktionieren, und merken Sie sich, daß Sie mit Esc (+ Return) jederzeit zwischen Edit- und Direkt-Bildschirm hin- und herwechseln können.

3. Mathematik

In diesem Abschnitt werden wir nicht nur die mathematischen Befehle erklären, sondern in diesem Zusammenhang auch auf das semantisch-syntaktische Grundgerüst der Programmiersprache BASIC eingehen.

3.1. Taschenrechner

BASIC-Anfänger beginnen am besten mit den mathematischen Befehlen, weil diese von Taschenrechnern bereits teilweise bekannt sind. Übungshalber gehen wir deshalb mit Esc in den Direkt-Modus. Wenn wir nun beispielsweise
Print 8 * 8
tippen, erhalten wir
64
als Ergebnis am Bildschirm angezeigt. Weitere Übungen:

```
Print 1Ø+2              ergibt   12
Print 1Ø-2              ergibt    8
Print 1Ø*2              ergibt   2Ø
Print 1Ø/2              ergibt    5
Print (2*3)+(3*4)       ergibt   18
Print  2*3 + 3*4        ergibt   18
Print (2+3)*(3+4)       ergibt   35
Print 2 + 3 * 3 + 4     ergibt   15
Print -8 + -8           ergibt  -16
Print -8 -  8           ergibt  -16
Print 1Ø*1Ø             ergibt  1ØØ
Print 1Ø↑2              ergibt  1ØØ
Print Sqr(1ØØ)          ergibt   1Ø
Print 1ØØ↑(1/2)         ergibt   1Ø
Print 1Ø*1Ø*1Ø          ergibt 1ØØØ
Print 1Ø↑3              ergibt 1ØØØ
Print 1ØØØ↑(1/3)        ergibt   1Ø
Print Pi                ergibt  3.141...
Print Tan(45)           ergibt  1.619...
Print Tan(45*Pi/18Ø)    ergibt  1.Ø
Print Atn(1)            ergibt  Ø.785...
Print Atn(1)*18Ø/Pi     ergibt 45.Ø
```

Nach diesen mehr intuitiven Beispielen, mit denen Sie als BASIC-Anfänger zunächst eine Zeitlang experimentieren sollten, können wir nunmehr zu einer systematischen Darstellung übergehen.

3.2. Befehlswörter

GFABASIC kennt etwa 300 Befehlswörter, z.B. Print, Input usw. Diese Wörter gelten als reserviert, denn sie dürfen nicht als Variablennamen verwendet werden.
Befehle werden auch als Anweisungen bezeichnet. Im einzelnen unterscheidet man
– einfache Anweisungen, z.B. Zuweisungen (Y = 10), Prozeduren (z.B. Print Y) und Funktionen (z.B. Y = Sin(X)), sowie
– strukturierte Anweisungen, z.B. bedingte Anweisungen (If A = B Then...) und Wiederholungsanweisungen (Repeat ... Until A = B).

Die Anweisung **Print** gibt beispielsweise diejenigen Werte am Bildschirm aus, die auf das Befehlswort Print folgen:
Print 7
gibt „7", gefolgt von Return, aus.
Print 7;
gibt „7" *ohne* Return aus.
Print 7;8
gibt „78" mit Return nach 8 aus.
Print 7 * 8
gibt das *Ergebnis* „56" aus.
Print "7 * 8"
gibt die Zeichenfolge „7 * 8" aus, also das, was in Anführungszeichen steht.
Print "7 * 8 ="; 7 * 8
gibt „7 * 8 = 56" aus.

3.3. Zahlenarten

GFABASIC unterscheidet 3 Zahlentypen:
Ganzzahlen (Integer-Zahlen), bei denen als Zahlzeichen nur die Ziffern 0 bis 9 sowie die Vorzeichen + und - zulässig sind, liegen im Bereich
-2 147 483 648 bis +2 147 483 647
und werden intern als 4 Bytes gespeichert ($00000000 bis $7FFFFFFF für positive Ganzzahlen).
Fließkommazahlen (Real-Zahlen), bei denen zusätzlich der Dezimalpunkt sowie ggf. „E" bzw. „e" für die Exponentialform zulässig sind, liegen im Bereich
-1.0 E+154 bis +1.0 E+154
und werden intern als 6 Bytes gespeichert (wobei für den Exponenten 9 Bits zur Verfügung stehen: 2 ↑ (2 ↑ 9) = 10 ↑ 154). Die Mantisse kann bis zu 11 Ziffern umfassen, z.B. 1.2345678901.
Wahrheitswerte (boolesche Werte), die für True = -1 und False = 0 stehen und intern als 2 Bytes gespeichert werden (bei Arrays als 1 Bit).

Wenn Sie am internen Zahlenaufbau interessiert sind, so können Sie das nachfolgende Programm eingeben, das wahlweise Fließkomma- oder Ganzzahlen in binärer und hexadezimaler Form am Bildschirm anzeigt. Damit Sie die Bitmuster besser erkennen können, geben Sie übungshalber Zweierpotenzen minus 1 ein, z.B. 255 (2 ↑ 8-1), 2147483647 (= 2 ↑ 31-1) usw.

```
Rem Interner Zahlenaufbau
Fl$=" ØØØØØØØ"
F2$="        Ø"
Do
  Print
  Input " Zahl: ",Z$
  Exit If Len(Z$)=Ø
  Z$=Upper$(Z$)
  Z=Val(Z$)
  If Instr(Z$,".")>Ø Or Instr(Z$,"E")>Ø
    M$=Mkf$(Z)
  Else
    M$=Mkl$(Z)
  Endif
  For I=l To Len(M$)
    B$=Bin$(Asc(Mid$(M$,I,l)))
    Print Left$(Fl$,9-Len(B$));B$;
  Next I
  Print "  Zahl ";Z
  For I=l To Len(M$)
    H$=Hex$(Asc(Mid$(M$,I,l)))
    Print Left$(F2$,9-Len(H$));H$;
  Next I
  Print
Loop
Clear
```

3.4. Variablen

Wie in der Mathematik können Variablen oder genauer Variablennamen an die Stelle konkreter Zahlen treten, z.B. „Print X * Y". Beim GFA-BASIC können Variablennamen theoretisch bis zu 255 Zeichen lang sein und werden dann auch bis zur vollen Länge unterschieden, z.B. „Langname.x" und „Langname.y". Variablennamen müssen mit einem Buchstaben beginnen; danach dürfen neben Buchstaben auch Ziffern folgen. Zwischen Groß- und Kleinbuchstaben wird nicht unterschieden. Ferner sind der eingebettete Punkt (.) und der Unterstreichungsstrich (_) erlaubt, also z.B. A, A1, A_B, A.B usw. Man beachte jedoch, daß keine reservierten Befehlswörter (Print, Input usw.) als Variablennamen deklariert werden dürfen.
Damit der BASIC-Interpreter zwischen den verschiedenen Datentypen unterscheiden kann, werden Variablennamen mit Datentypkennungen versehen:

```
Name% (%)     Integer-Variable
Name! (!)     boolesche Variable
Name$ ($)     String-Variable
Name  ( )     Real-Variable.
```

Wenn also ein Variablenname keinen Datentypzusatz aufweist (X, Y, Z usw.), dann wird automatisch eine Real-Variable unterstellt.

3.5. Literalien und Konstanten

Zu den **Literalien** (englisch: Literals) als den „buchstabengetreuen Ausdrücken" zählen (Beispiele):

```
1          Integer-Literal
-1         Integer-Literal
1.Ø        Real-Literal
1.ØE1Ø     Real-Literal
1E1Ø       Real-Literal
&HFF       Hex-Literal (&H...)
&Ø77       Oktal-Literal (&Ø...)
&X1Ø       Binär-Literal (&X...)
abc        String-Literal
"abc"      String-Literal
```

Konstanten sind Werte, die sich im Laufe des Programms nicht ändern, gleichgültig ob sie als Literalien (z.B. 3.1415) oder als Konstantennamen (z.B. Pi) deklariert werden. GFABASIC kennt u.a. die vordefinierten Konstanten

```
Pi      3.1415...
True    -1
False   Ø
```

Daneben kann man eigene Konstanten definieren (z.B. E = 2.7182), die jedoch in BASIC die gleiche Funktion wie Variablen haben und deshalb auch im Laufe eines Programms umdefiniert werden könnten. Dies wäre in Pascal nicht zulässig.

3.6. Zuweisung

3.6.1. Terme

Bei einer mathematischen Gleichung, z.B.
2x + 3x - 5 + 7 = 5x + 2
bezeichnen wir die Ausdrücke, die links und rechts vom Gleichheitszeichen stehen, als Terme. Bei den höheren Programmiersprachen sind Terme
– Literalien,
– Variablen,
– Konstanten,
– Funktionen oder
– Formelausdrücke.
Eine Wertzuweisung hat die Form „Variable = Term", z.B.

```
Y = 10        Literal
Y = X         Variable
Y = Pi        Konstante
Y = Sin(X)    Funktion
Y = X * 10    Formelausdruck
```

Das Zeichen „=" ist hier kein Gleichheitszeichen, wie es bei den später zu erläuternden Vergleichsaussagen vorkommt (z.B. If Y = X Then...), sondern ein Wertzuweisungszeichen, wofür zur Vermeidung von Mehrdeutigkeiten in Pascal „:=" oder in der Mathematik der Pfeil (←) verwendet wird. „Y = 10" bedeutet folglich, daß Y den Wert 10 annehmen soll. Allgemein bedeutet eine Zuweisung, daß der links vom „=" stehenden Variablen der Wert des rechts vom „=" stehenden Terms zugewiesen werden soll.

3.6.2. Funktionen

Eine Funktion besteht aus dem Funktionswort (z.B. Sin) und dem eingeklammerten Term, der in der Mathematik als **Funktionsargument** X bezeichnet wird, und liefert einen Wert, der als *Funktionsergebnis* oder Funktionswert Y bezeichnet wird.
1. Wenn der Wert des Arguments X einer Variablen Y zugewiesen wird, so sprechen wir von einer *expliziten Funktion*, also z.B.
Y = Sin(X) oder
S = Sin(30) usw.
2. Dagegen liegt eine *implizite Funktion* vor, wenn dem Funktionswert eine Anweisung (meist Print) vorangestellt wird, z.B.
Print Sin(X) oder
Print Sin(30) usw.
Funktionen können wie Klammerausdrücke geschachtelt werden:
Print Sqr(Sqr(16))
ergibt 2, denn die Quadratwurzel (Sqr = square root) von 16 ist 4 und die Quadratwurzel von 4 ist 2.

Darüber hinaus erwähnen wir für fortgeschrittene BASIC-Programmierer drei Sonderfälle:

1. Einige Funktionen dürfen nur in Print-Konstruktionen und niemals in Wertzuweisungen auftreten, z.B. Spc und Tab:
Print Spc(10); "Hallo" !richtig
S$ = Spc(10) !falsch
2. Einige Funktionen haben aus syntaktischen Gründen ein Dummy-Argument, das zwar ignoriert wird, aber trotzdem nicht fehlen darf, z.B. Lpos(dummy) und Fre(dummy).
3. Einige Funktionen haben scheinbar überhaupt kein eingeklammertes Argument, z.B. Inkey$, Crscol, Crslin usw., z.B.
T$ = Inkey$
In Wirklichkeit werden bei solchen Funktionen Hardware-Zustände (Tastatur, Cursorposition usw.) abgefragt, die dann intern als Argumente übergeben werden.

3.6.3. Operatoren

Operatoren sind die von der Mathematik her vertrauten Operationszeichen. Neben den mathematischen Operatoren (+, - usw.) gibt es die Vergleichsoperatoren (<, > usw.), die logischen Operatoren (And, Or usw.) sowie den String-Verkettungsoperator (+). Zu den mathematischen Operatoren gehören im einzelnen:

Klammern:
(runde Klammer auf
) runde Klammer zu
Klammerausdrücke genießen die höchste Priorität, d.h. eingeklammerte Ausdrücke werden vor allen anderen Ausdrücken ausgerechnet. Man vergleiche:
2 * 2 + 2 * 2 ergibt 8
2 * (2 + 2) * 2 ergibt 16

Vorzeichen:
+ positiv (entfällt meist)
- negativ
Beispiele:
-2 + 3 ergibt +1
-2 + -3 oder besser
-2 + (-3) ergibt -5
-X + (-Y)

Real-Operatoren:
↑ Potenz, z.B. 5.0 ↑ 2.0 → 25.0
+ Addition, z.B. 5.0 + 2.0 → 7.0
- Subtraktion, z.B. 5.0 - 2.0 → 3.0
* Multiplikation, z.B. 5.0 * 2.0 → 10.0
/ Division, z.B. 5.0 / 2.0 → 2.5
Es gelten die Prioritäten: zuerst Potenzierung, dann Punktrechnung vor Strichrechnung, z.B.
100.0 ↑ (1/2) → 10.0
100.0 ↑ 1/2 → 50.0
Achtung: Bei der Potenzierung muß eine negative Basis eingeklammert werden:
Print −2 ↑4 → −16
Print (−2) ↑4 → +16
Man muß in GFABASIC nicht notwendigerweise 2.0 statt 2 schreiben, wenn aus dem Zusammenhang klar ist, daß Fließkommazahlen und nicht Ganzzahlen gemeint sind, weil immer auf denjenigen Zahlentyp umgerechnet bzw. abgerundet wird, der als Ergebnis verlangt wird. Wenn keine Rundung möglich ist, wird eine Fehlermeldung ausgegeben. Beispiele:
I% = Pi ↑ 18
Print I% ergibt 888582403
R = Pi ↑ 18
Print R ergibt 888582403.08
I% = Pi ↑ 19 ergibt Fehlermeldung
Wenn ein Term mindestens eine Fließkommazahl enthält, wird der *gesamte* Term als Fließkommazahl berechnet und dann ggf. auf die Integer-Variable abgerundet. Da reine Integer-Berechnungen erheblich schneller als Real-Berechnungen sind, sollten Sie gemischte Ausdrücke vermeiden, soweit dies sinnvoll und möglich ist.

Integer-Operatoren
+ Addition, z.B. 5 + 2 → 7
- Subtraktion, z.B. 5 - 2 → 3
Div Division, z.B. 5 Div 2 → 2
Mod Rest, z.B. 5 Mod 2 → 1
„Div" steht für die Ganzzahldivision und „Mod" für den Rest aus einer Ganzzahldivision: 5 geteilt durch 2 = 2 Rest 1.

Schnelle Grundrechenarten
Wenn man auf den ursprünglichen Wert einer Fließkomma- oder Ganzzahlvariablen verzichten kann, lassen sich bestimmte Grundrechenarten etwa doppelt so schnell ausführen:

Add X,F entspricht X = X + F
Sub X,F entspricht X = X - F
Mul X,F entspricht X = X * F
Div X,F entspricht X = X / F
Beispiele:
F = 2 ! Faktor F bleibt unverändert
X = 3 ! Variable X wird verändert
Add X,F → X = 5
Sub X,F → X = 3
Mul X,F → X = 6
Div X,F → X = 3
Anstelle der Quadrierung X * X können wir auch Mul X,X verwenden.

Dec X entspricht X = X - 1
Inc X entspricht X = X + 1
Beispiele
X = 5
Dec X → X = 4
Inc X → X = 5

3.7. Funktionen

In GFABASIC sind folgende mathematische Funktionen vordefiniert, d.h. bereits in die Runtime-Bibliothek eingebaut:

3.7.1. Transzendentale Funktionen

Bei diesen Funktionen, die nur in Sonderfällen „glatte Zahlen" liefern, z.B. bei Print Sqr(4) u.a., muß mit Rundungsfehlern ab der 10. Stelle gerechnet werden. Als Funktionsargumente dienen Real- und Integer-Terme, während die Funktionsergebnisse meistens Real-Zahlen sind, die jedoch oft auf Integer-Zahlen abgerundet werden können.

Y = Log(X)
liefert den natürlichen Logarithmus (zur Basis e), während
Y = Log10(X)
den dekadischen Logarithmus (zur Basis 10) ausrechnet. Die Argumente müssen positive Zahlen sein.
Y = Exp(X)
entspricht e ↑ X, wobei e = 2.7182... Beispiele:
Print Log10(100) → 2
Print Log(100)/Log(10) → 2
Print Exp(Log(10)+Log(10)) → 10 * 10
Y = Exp(1) → e
Y = Log(Exp(1)) → 1
Wenn gilt
243 = 3 ↑ 5 oder
P = G ↑ H bzw.
Potenz = Grundzahl ↑ Hochzahl,
dann gilt
P = Exp(Log(G)*H), z.B.
P = Exp(Log(3)*5) → 243
P = 3 ↑ 5 ist hier genauer!
G = Exp(Log(P)/H), z.B.
G = Exp(Log(243)/5) → 3
G = 243 ↑ (1/5) ist hier ebenso ungenau!
H = Log(P)/Log(G), z.B.
H = Log(243)/Log(3) → 5

Y = Sin(X)
Y = Cos(X)
Y = Tan(X)
berechnen Sinus, Kosinus und Tangens des Winkels X im Bogenmaß (rad). Bei Verwendung des Gradmaßes (360 Grad) muß man das *Argument* X mit Pi/180 = 0.01745.. multiplizieren.
Y = Atn(X)
ist der Arkustangens oder die Umkehrfunktion zu Tan(X). Für die Umrechnung in das Gradmaß muß man das *Ergebnis* Y mit 180/Pi =

57.2957... multiplizieren. Arkussinus und -kosinus fehlen und müssen simuliert werden. Beispiele:
Print Sin(30*Pi/180) → 0.5
Print Tan(45*Pi/180) → 1.0
Print Atn(1)*180/Pi → 45.0
Y = 4 * Atn(1) → Pi
Y = Sin(x)/Cos(x) → Tangens
Y = Cos(x)/Sin(x) → Kotangens
Y = 1/Tan(x) → Kotangens
Y = Atn(X/Sqr(1-X*X)) → Arkussinus
Y = Pi/2-Atn(X/Sqr(1-X*X)) → Arkuskosinus
Achtung: Gewisse Tangens-Argumente, z.B. Print Tan(Pi/2), die mathematisch unzulässig wären, führen bei GFABASIC *nicht* zu einer Fehlermeldung („Division durch Null"), sondern liefern statt dessen falsche Ergebnisse!

Y = Sqr(X)
berechnet die Quadratwurzel aus einer positiven Real- oder Integer-Zahl. Für Kubikwurzeln usw. muß man auf die Potenzierung mit hoch 1/X zurückgreifen, z.B.
Print 8 ↑ (1/3) ergibt 2.
Übrigens ist im Gegensatz zu Taschenrechnern die Kehrwertfunktion 1/X nicht implementiert. Desgleichen gibt es keinen speziellen Quadrierungsbefehl, der in Pascal als Sqr im Gegensatz zu Sqrt vorhanden ist, doch kann man sich leicht mit Y = X * X (oder auch mit Mul X,X, s.o.) behelfen. Im übrigen gilt bis zur Kubikzahl:
X * X schneller als X ↑ 2
X * X * X schneller als X ↑ 3
Danach ist die Potenzierung schneller. Ferner ist beispielsweise Sqr(Sqr(X)) = 4. Wurzel schneller als X ↑ (1/4). Bei rechenintensiven Programmen lohnt es sich also, mit den möglichen Befehlen zu experimentieren.

Y% = Random(Obergrenze%)
erzeugt eine Integer-Zufallszahl im Bereich von 0 (einschließlich) bis zur Obergrenze (ausschließlich), während
Y = Rnd[(Dummy)]
eine Real-Zufallszahl im Bereich von 0.0 (einschließlich) bis 1.0 (ausschließlich), d.h. von 0.0 bis 0.999..., generiert. Das Dummy-Argument kann einschließlich der runden Klammern entfallen. Beispiele:
Print Random(100)+1
erzeugt Integer-Zahlen im Bereich 1 bis 100.
Print Rnd(1)*100+10 oder
Print Rnd*100+10
erzeugt Real-Zahlen im Bereich 10.0 bis 110.0. Da in beiden Fällen (Random und Rnd) derselbe Fließkomma-Algorithmus verwendet wird, ist Rnd sogar noch schneller, weil bei Random zusätzlich auf die Integer-Zahl abgerundet werden muß.
Anmerkung: Der Randomize-Befehl bzw. die Rnd-Funktion mit Nicht-Dummy-Argument fehlt beim GFABASIC, so daß sich *dieselbe* Zufallszahlenfolge nicht ohne Tricks regenerieren läßt. Allerdings ist die Zufallszahlenverteilung recht günstig:

```
Rem "Schnee"
Do
  X%=Random(640)
  Y%=Random(400)
  Plot X%,Y%
Loop
```

Dieses Programm plottet wahllos Grafik-Punkte auf dem Bildschirm. Nach etwa einer Stunde ist der gesamte Bildschirm „zugeschneit".

3.7.2. Vorzeichen- und Rundungsbefehle

Diese Befehle dienen dem Runden und Abhacken der Vor- und Nachkommastellen von Real-Zahlen sowie der Ermittlung des Vorzeichens und der „Geradheit" von Zahlen.

Y = Abs(X)
entfernt ein möglicherweise vorhandenes negatives Vorzeichen. Beispiele:
Print Abs(-2) → 2
Print Abs(10) → 10 bleibt unverändert.
Für die Negierung einer positiven Zahl verwende man
X = -X
Y = Sgn(X)
liefert das Vorzeichen (= Signum) wie folgt (Beispiele):
Print Sgn(+10) → 1 → positiv
Print Sgn(-10) → -1 → negativ
Print Sgn(0) → 0 → Null
Ein Programmfragment für Fortgeschrittene:

```
Input X
On Sgn(X)+2 Gosub Minus, Null, Plus
Rem              1      2     3
End
Procedure Minus
 Print X " ist minus"
Return
usw.
```

Y! = Even(X)
Y! = Odd(X)
liefern den booleschen Wert True (-1), wenn X gerade („even"), bzw. False (0), wenn X ungerade („odd") ist. Beispiele:
Print Even(10) → -1 → True
Print Odd(10) → 0 → False
Print Even(11) → 0 → False
Print Odd(11) → -1 → True
Diese Befehle kann man vor Dpeek/Dpoke und Lpeek/Lpoke verwenden, um ungerade Speicheradressen zu vermeiden.

Y = Fix(X) oder Y = Trunc(X)
weisen der Variablen Y den ganzzahligen bzw. Vorkomma-Teil der Fließkommavariablen X zu (Fix und Trunc sind funktionsgleich), während
Y = Frac(X)
den Bruch- bzw. Nachkomma-Teil von X zuweist. Es gilt demnach theoretisch
X = Fix(X) + Frac(X),
doch produziert Frac Rundungsfehler, die offenbar über die üblichen Binärfehler hinausgehen (vgl. Peeker, 12/86, S. 68-70). Beispiele:
X = 123.456
Print Fix(X) → 123
Print Frac(X) → 0.45599999977
Sgn(X) * Int(Abs(X)) → Fix(X)

Y = Int(X)
Während Fix(X) die Nachkommastellen abhackt, *rundet* Int(X) auf die *nächstkleinere* Ganzzahl, so daß Y kleiner/gleich X ist. Beispiele:
Print Int(1.5) → 1 wie Fix
Print Int(-1.0) → 1 wie Fix
Print Int(-1.5) → -2 bei Fix jedoch -1
Print Int(-2.0) → -2 wie Fix
Auf dem Zahlenstrahl
... -2.0 -1.5 -1.0 +0.0 +1.0 +1.5 ...
ist -2 die nächstkleinere Ganzzahl nach -1.5. Fix und Int unterscheiden sich also nur bei negativen Zahlen, die Nachkommastellen haben.

Rundung auf N Nachkommastellen
Oft möchte man auf N Nachkommastellen aufrunden. Dabei unterscheidet man zwischen V = Vorkommastellen, N = Nachkommastellen und G = Gesamtstellen. Den Platz für Vorzeichen (+/-) und Dezimalpunkt muß man gesondert berücksichtigen. Es bieten sich dann folgende Möglichkeiten an:

Rundungsformel
Y = Int(X*10 ↑ N+0.5)/10 ↑ N
oder für negative Zahlen
Y = Sgn(X) * Int(X*10 ↑ N+0.5)/10 ↑ N
wobei N für die Anzahl der Nachkommastellen steht. Beispiel:
X = 123.456
N = 2, damit 10 ↑ N = 100:
Print Int(X*100+0.5)/100 → 123.46
X = 123.4
Print Int(X*100+0.5)/100 → 123.4!
Nachteil: Wenn weniger als N geltende Nachkommastellen vorhanden sind, wird nicht mit Nullen aufgefüllt. Man muß deshalb mit entsprechenden String-Befehlen nachhelfen.

Defnum G
Mit diesem Befehl wird die Gesamtstellenanzahl G für *alle* Zahlenausgaben festgelegt (1 >= G <= 11). Beispiele:
Defnum 5
Print 123456.7 → 123460!
Print 12345.67 → 12346!
Print 1234.567 → 1234.6!
Print 123.4567 → 123.46
Print 123.456 → 123.46
Print 123.45 → 123.45
Print 123.4 → 123.4!
Nachteil: Der globale Defnum-Befehl ist nur sinnvoll, wenn bekannt ist, daß G - V >= N.

Print Using "###.##", X
Dieser Befehl arbeitet mit rechtsbündig gefüllten Ausgabemasken (# = Dezimalstelle, „." = Dezimalpunkt). Beispiele:
Print Using "###.##", 123456 → "%123456"
Print Using "###.##", 12345 → "%12345"
Print Using "###.##", 1234.5 → "%1234.5"
Print Using "###.##", 123.45 → "123.45"
Print Using "###.##", 123.456 → "123.46"
Print Using "###.##", 12.345 → "□12.35"
Print Using "###.##", -12.345 → "-12.35"
Print Using "###.##", -12.3 → "-12.30"
Nachteil: Wenn die Vorkommastellen für die Zahl nicht (einschließlich des automatisch berücksichtigten Vorzeichens) ausreichen, wird die Maskenlänge überschritten und „%" als Fehlermeldung vorangestellt. Dafür werden gegenüber den obigen Befehlen fehlende Nachkommastellen mit Nullen aufgefüllt.

Weitere Feinheiten (Beispiele):
Auch „+" – ganz links – ausgeben:
Print Using "+###.##", 1 → "+□□1.00"
Führende Nullen mit „*" auffüllen:
Print Using "***.##", 1 → "**1.00"
Komma für Tausenderstelle einfügen:
Print Using "##,###.##", -1000 → "-1,000.00"

Binärmathematische Rundungsfehler

Programm-Listings

Im Peeker, Heft 12/85, S. 68–70, haben wir die Ursachen binärmathematischer Rundungsfehler erläutert. Im Nachtrag zu diesem Beitrag bringen wir im folgenden die entsprechenden GFABASIC-Programme:

Binärbruch-Umwandlung

Dieses Programm verwandelt einen über Input eingegebenen Dezimalbruch, z. B. 1234 5678 90.1234 5678 90 (maximal je 10 Vor- und Nachkommastellen) in den exakten Binärbruch um, wobei der Anfang der Periode durch "–" markiert wird. Die Anzahl der Nachkommastellen (hier Maxstelle % = 2000) kann erhöht werden, denn bei dem Dezimalbruch 0.12 345 würde sich bereits eine 500stellige Binärbruch-Periode ergeben.

Grundrechenarten

Bei diesem Programm gibt man zunächst den 1. Operanden (z. B. 1677 7216), dann den Operator (z. B. +) und schließlich den 2 Operanden (z. B. 0.0009765625) über Input ein. Danach erfolgt eine Ausgabe der internen Zahlenformate wie bei den nachstehenden Beispielen, die wir erneut abdrucken, weil im letzten Heft die jeweils vorletzten Zeilen von der Reproabteilung irrtümlich auseinandergeschnitten wurden.

Newton-Wurzel

Dieses Programm, das aus Platzgründen nur teilweise gelistet ist, zeigt Rundungsfehler bei Näherungsverfahren an, und zwar hier am Beispiel der Quadratwurzelberechnung.

```
16777216 + 0.0009765625 = 16777216.001    16777216.0010000000

 00000000 00000000 00000000 00000000 00000010 00011000    16777216
 00000000 00000000 00000000 00000000 00000001 11110110    0.0009765625
 00000000 00000000 00000000 00000000 00100010 00011000    16777216.001

 00000000000000000000000000000000000000000000  1000011000      16777216
 00000000000000000000000000000000000000000000  0111110110      0.0009765625
 00000000000000000000000000000000000001000  1000011000         16777216.001

+1.00000000000000000000000000000000000000000 +000011000 +24   16777216
+1.00000000000000000000000000000000000000000 -000001010 -10   0.0009765625
+1.00000000000000000000000000000000001000 +000011000 +24      16777216.001

+1000000000000000000000000.000000000000
                    +0.000000000100000000000000000000000000000000000000
+1000000000000000000000000.000000000001000


134217728 + 0.0009765625 = 134217728   134217728.000000000

 00000000 00000000 00000000 00000000 00000010 00011011    134217728
 00000000 00000000 00000000 00000000 00000001 11110110    0.0009765625
 00000000 00000000 00000000 00000000 00000110 00011011    134217728

 00000000000000000000000000000000000000000000  1000011011      134217728
 00000000000000000000000000000000000000000000  0111110110      0.0009765625
 00000000000000000000000000000000000000000001  1000011011      134217728

+1.00000000000000000000000000000000000000000 +000011011 +27   134217728
+1.00000000000000000000000000000000000000000 -000001010 -10   0.0009765625
+1.00000000000000000000000000000000000000001 +000011011 +27   134217728

+1000000000000000000000000.000000000000
                    +0.000000000100000000000000000000000000000000000000
+1000000000000000000000000.000000000001


536870912 + 0.0009765625 = 536870912   536870912.000000000

 00000000 00000000 00000000 00000000 00000010 00011101    536870912
 00000000 00000000 00000000 00000000 00000001 11110110    0.0009765625
 00000000 00000000 00000000 00000000 00000010 00011101    536870912

 00000000000000000000000000000000000000000000  1000011101      536870912
 00000000000000000000000000000000000000000000  0111110110      0.0009765625
 00000000000000000000000000000000000000000000  1000011101      536870912

+1.00000000000000000000000000000000000000000 +000011101 +29   536870912
+1.00000000000000000000000000000000000000000 -000001010 -10   0.0009765625
+1.00000000000000000000000000000000000000000 +000011101 +29   536870912

+1000000000000000000000000000.000000000
                    +0.000000000100000000000000000000000000000000000000
+1000000000000000000000000000.000000000
```

Letztes Bit wird intern und extern erfaßt.

Letztes Bit wird nur noch intern erfaßt.

Letztes Bit wird nicht mehr erfaßt.

Binärbruch-Umwandlung

```
Print "Dezimal- in Binärbruch"
Maxstelle%=2000
Dim Rest%(Maxstelle%)
Dim Ziffer%(Maxstelle%)
Do
  Print
  Print
  Input "Dezzahl eingeben: ",Dezzahl$
  Rem --- Vor- und Nachkommastellen ---
  Exit If Dezzahl$=""
  I%=Instr(Dezzahl$,".")
  Exit If I%<2 !Punkt fehlt
  Vorkomma=Abs(Val(Left$(Dezzahl$,I%-1)))
  Exit If Vorkomma>2^30
  Exit If Len(Dezzahl$)=I%
  Nachkomma=Val(Mid$(Dezzahl$,I%+1))
  Cls
  Print Int(Vorkomma);".";Int(Nachkomma)
  Print
  If Left$(Dezzahl$,1)="-"
    Print "-";
  Endif
  Rem --- Vorkommastellen ausgeben ---
  Ganzzahl%=Vorkomma
  If Ganzzahl%=0
    Print "0";
  Else
    Potenz%=1
    While Ganzzahl%>=Potenz%
      Potenz%=Potenz%*2
    Wend
    Repeat
      Potenz%=Potenz% Div 2
      If Ganzzahl%>=Potenz%
        Print "1";
        Ganzzahl%=Ganzzahl%-Potenz%
      Else
        Print "0";
      Endif
    Until Potenz%=1
  Endif
  Rem --- Binärpunkt ---
  Print ".";
  Rem --- Nachkommastellen ---
  Zaehler%=Nachkomma
  Nenner%=10^(Len(Str$(Zaehler%)))
  If Zaehler%=0
    Print "0";
  Else
    Stelle%=0
    Flag%=0
    Periode%=0
    Do
      Zaehler%=2*Zaehler%
      Inc Stelle%
      If Zaehler%>=Nenner%
        Ziffer%(Stelle%)=1
        Zaehler%=Zaehler%-Nenner%
      Else
        Ziffer%(Stelle%)=0
      Endif
      Rest%(Stelle%)=Zaehler%
      Exit If Stelle%=Maxstelle%
      Exit If Zaehler%=0
      For X%=1 To Stelle%
        If Rest%(X%)=Zaehler%
          If Stelle%<>X%
            Periode%=X%
            Flag%=1
          Endif
        Endif
      Next X%
      Exit If Flag%=1
    Loop
    For X%=1 To Stelle%
      Print Ziffer%(X%);
      If X%=Periode%
        Print "_";
      Endif
    Next X%
  Endif
Loop
Clear
```

Grundrechenarten

```
Dim Z(3)
Dim Z$(3,4)
Dim P(3)
Do
  Print
  Input "1. Zahl :",Z$
  Exit If Len(Z$)=0
  Z(1)=Val(Z$)
  Print "+ - * / :";
  Repeat
    0$=Inkey$
  Until 0$="+" Or 0$="-"
   Or 0$="*" Or 0$="/"
  Print 0$
  Input "2. Zahl :",Z$
  Z(2)=Val(Z$)
  If 0$="+"
    Z(3)=Z(1)+Z(2)
  Else
    If 0$="-"
      Z(3)=Z(1)-Z(2)
    Else
      If 0$="*"
        Z(3)=Z(1)*Z(2)
      Else
        Z(3)=Z(1)/Z(2)
      Endif
    Endif
  Endif
  Cls
  Rem *** Bitstrings erzeugen
  For Z=1 To 3
    Gosub Realanzeige(Z(Z))
    Z$(Z,1)=Anzeige1$
    Z$(Z,2)=Anzeige2$
    Z$(Z,3)=Anzeige3$
    Z$(Z,4)=Anzeige4$
  Next Z
  Rem *** Bitstrings anzeigen
  Print
  Print Z(1);" ";0$;" ";Z(2);" = ";Z(3);" ";
  Print Using "##########.#########",Z(3)
  Print
  For X=1 To 3
    For Y=1 To 3
      Print Z$(Y,X);Tab(58);Z(Y)
    Next Y
    Print
  Next X
  Rem *** Punkt unter Punkt
  For X=1 To 3
    P(X)=Instr(Z$(X,4),".")
  Next X
  P=Max(P(1),P(2),P(3))
  For X=1 To 3
    Print Spc(P-P(X));Z$(X,4)
  Next X
Loop
Clear
End
```

Zweierpotenzen (Tabellenauszug)

+4	2↑+04	16.0000
+3	2↑+03	8.0000
+2	2↑+02	4.0000
+1	2↑+01	2.0000
+0	2↑+00	1.0000
-1	2↑-01	0.5000
-2	2↑-02	0.2500
-3	2↑-03	0.1250
-4	2↑-04	0.0625

Unterprogramm Real-Binärumwandlung

```
Procedure Realanzeige(Zahl)
  Rem *** Binärmasken
  Maske1$=" 00000000"
  Maske2$="00000000"
  Maske3$="0000000000"
  Rem *** Anzeigen 1 und 2 ------------------------------------
  Zahl$=Mkf$(Zahl)
  Bits$=""      !Endlos-Bitstring
  Anzeige1$="" !aufgeteilt als 6 Bytes zu je 8 Bits
  Anzeige2$="" !aufgeteilt in Mantisse + Exponent
  Anzeige3$="" !normiert in Exponentialdarstellung
  Anzeige4$="" !umgewandelt in Nicht-Exponentialdarstellung
  For I%=1 To Len(Zahl$)
    B$=Bin$(Asc(Mid$(Zahl$,I%,1)))
    Anzeige1$=Anzeige1$+Left$(Maske1$,9-Len(B$))+B$
    Bits$=Bits$+Left$(Maske2$,8-Len(B$))+B$
  Next I%
  Anzeige2$=" "+Left$(Bits$,38)+"  "+Right$(Bits$,10)
  Rem *** Auswertung der Bits ---------------------------------
  Mantisse$=Left$(Bits$,38)
  Exponent$=Right$(Bits$,9)
  Man_sign$=Mid$(Bits$,1,1)
  Exp_sign$=Mid$(Bits$,39,1)
  Exp_wert%=Val("&x"+Exponent$)
  Rem *** Zahl 0 ist ein Sonderfall!
  Zero_flag!=(Man_sign$="0" And Exp_sign$="0" And Exp_wert%=0)
  If Man_sign$="1"
    Man_vorz$="-"
  Else
    Man_vorz$="+"
  Endif
  If Zero_flag!=False
    Mid$(Mantisse$,1,1)="1"
  Endif
  If Zero_flag!=True
    Exp_vorz$="+"
  Else
    If Exp_sign$="1"
      Exp_vorz$="+"
    Else
      Exp_vorz$="-"
      Exp_wert%=512-Exp_wert%
    Endif
  Endif
  Exponent$=Bin$(Exp_wert%)
  Exponent$=Left$(Maske3$,9-Len(Exponent$))+Exponent$
  Rem *** Anzeige 3 -------------------------------------------
  Anzeige3$=Man_vorz$+Left$(Mantisse$,1)+"."+Mid$(Mantisse$,2,37)
  Anzeige3$=Anzeige3$+" "+Exp_vorz$+Exponent$
  Anzeige3$=Anzeige3$+" "+Exp_vorz$+Str$(Exp_wert%)
  Rem *** Anzeige 4 -------------------------------------------
  Anzeige4$=Man_vorz$
  If Zero_flag!=True
    Anzeige4$=Left$(Mantisse$,1)+"."+Right$(Mantisse$,37)
  Else
    If Abs(Exp_wert%)<38
      If Exp_vorz$="+"
        For I%=1 To Len(Mantisse$)
          Anzeige4$=Anzeige4$+Mid$(Mantisse$,I%,1)
          If I%=Exp_wert%+1
            Anzeige4$=Anzeige4$+"."
          Endif
        Next I%
      Else
        For I%=1 To Exp_wert%
          Anzeige4$=Anzeige4$+"0"
          If I%=1
            Anzeige4$=Anzeige4$+"."
          Endif
        Next I%
        Anzeige4$=Anzeige4$+Mantisse$
      Endif
    Else
      Anzeige4$=Anzeige4$+" zu groß"
    Endif
  Endif
Return
```

Newton-Wurzel

```
(Listing-Auszug)

Input "Radikand: ",R
S=Sqr(R)
Print "S = SQR(";R;") = ";S
W=R   !W=Wurzel
V=0   !V=Vorher-Wurzel
Z=0   !Z=Zähler
Do
  W=(W+R/W)/2
  Inc Z
  Exit If V=W Or Z=1000
  V=W
Loop
Gosub Anzeige
K=1.0E-11   !K=Konstante
W=R         !W=Wurzel
V=0         !V=Vorher-Wurzel
Z=0         !Z=Zähler
Do
  W=(W+R/W)/2
  Inc Z
  Exit If Abs(V-W)<K Or Z=1000
  V=W
Loop
Gosub Anzeige
```

Zahlenformate im Vergleich

01000000	1000000001	GFA	+3.0
11000000	01000010	DRI	
01000000	10000010	AS	
00000000	1000000000	GFA	+1.0
10000000	01000001	DRI	
00000000	10000001	AS	
00000000	0000000000	GFA	+0.0
00000000	00000000	DRI	
00000000	00000000	AS	
00000000	0111111111	GFA	+0.5
10000000	01000000	DRI	
00000000	10000000	AS	
00000000	0111111110	GFA	+0.25
10000000	00111111	DRI	
00000000	01111111	AS	
00000000	0111111101	GFA	+0.125
10000000	00111110	DRI	
00000000	01111110	AS	
01111100	1000000010	GFA	+7.875
11111100	01000011	DRI	
01111100	10000011	AS	
11111100	1000000010	GFA	-7.875
11111100	11000011	DRI	
11111100	10000011	AS	
Mantisse	Exponent		

Bücher

Westermann-Reihe Informatik
Lern- und Arbeitsbuch
mit Programmbeispielen in BASIC
von H.-G. Schumann
1986, 107 S., kart., DM 9,80
Westermann Schulbuchverlag, Braunschweig
Gliederung
Zahlen und Zählen – Algorithmen – Ein-, Ausgabe und Verarbeitung – Aufbau eines Heimcomputers – Ein BASIC-Programmbeispiel – Programmieren – Wiederholungen – Verfahren und Unterprogramme – Sammeln von Daten – Auswerten von Daten – Ein Kartei-Programm – Ein- und Ausgabe von Daten – Sichern und Laden – Ordnen und Suchen – Ändern und Löschen – Verbesserungen des Kartei-Programms – Zusammenfassung, Begriffs- und Befehlslexikon – Register
Bemerkungen
Das Buch ist als Grundlage für den Informatik-Unterricht oder für das Selbststudium geeignet. Ausgehend von den Grundbegriffen der Informatik wird der Leser stufenweise mit Befehlen, Verfahren und Prozeduren vertraut gemacht. Die Programmbeispiele werden in BASIC angeführt, Aufbau und Strukturierung der Programme oder Programmteile werden zuvor jeweils in Pascal-ähnlichen Ablaufplänen entwickelt. Am Ende des Buches liegt ein vollständiges Kartei-Programm in BASIC vor, das zuvor schrittweise erarbeitet wurde. Aufgaben am Ende jedes Kapitels dienen der Vertiefung des Erlernten und leiten zum eigenen Programmieren an. Eine interessantere Gestaltung, etwa durch farbige Abbildungen, Hervorheben wichtiger Abschnitte usw. dürfte dazu beitragen, daß es mehr Spaß macht, mit Hilfe dieses Buches in die Gedankenwelt der Informatik einzusteigen. Allerdings müßte man dazu sicher tiefer in die Tasche greifen.

Westermann-Reihe Informatik
Logo Lern- und Arbeitsbuch
von J. Ziegenbalg
1986, 152 S., kart., DM 9,80
Westermann Schulbuchverlag, Braunschweig
Gliederung
Grundelemente des Programmierens – Grundlegende Programmiertechniken – Aus der Teilbarkeitslehre – Brüche und Anwendungen – Funktionen, Wertetafeln, Schaubilder – Simulation zufälliger Ereignisse
Bemerkungen
Dieses Buch setzt Grundlagen der Informatik und Kenntnisse über den Umgang mit dem Computer voraus. Es vermittelt dem Leser den Einstieg in die Programmiersprache Logo anhand von Programmbeispielen für den Commodore C64. Da Logo recht geräteunabhängig ist, laufen die meisten Programme auch auf dem Apple II und Apple-Kompatiblen.
Allgemeine Programmiertechniken und Logo-spezifische Befehle werden an Beispielen verdeutlicht und durch Aufgaben aus den Bereichen Mathematik und Wirtschaft erlernt.

BTX auf Ihrem PC
von H.-P. Wagner
1986, 220 S., 100 Abb., kart., DM 45,–
Dr. Alfred Hüthig Verlag, Heidelberg
Gliederung
Einleitung – Der PC im BTX-Gesamtsystem – Einsatzmöglichkeiten des PC im BTX-Gesamtsystem – Die PC-Ausstattung für BTX – Der Einführungsprozess von BTX auf dem PC – Ausblick – Anhänge: Gebühren, Institutionen, Hardwarehersteller, Software, Fachwort-, Stichwort-, Literaturverzeichnis
Bemerkungen
Bildschirmtext (BTX) und Personal Computer (PC) sind die beiden neuen Technologien, die einer breiten Masse von Anwendern elektronische Datenverarbeitung und Kommunikation nahebringen werden. Dieses Buch bietet in allgemeinverständlicher Sprache Informationen über beide Bereiche. Die Vorteile des Zusammenwirkens beider Systeme in einem PC-BTX-Gesamtsystem werden eingehend erläutert und praktische Konzepte zur ihrer Realisierung dargestellt. Der Autor bespricht die Standards, die von der Post zum Einsatz von BTX vorgeschrieben sind und die Auswahl der Hardware beeinflussen, Standardsoftware und Peripheriegeräte. Im Kapitel 5 wird der Einführungsprozess von BTX auf dem Personal Computer behandelt. Ansprechpartner des Systems, Organisation, Programmentwicklung, Kostenfragen und eine optimale Gestaltung des BTX-Informationsangebots werden gründlich analysiert, um den Einsteigern in diese Technologie echte Hilfestellung zu leisten. Ein umfangreicher Anhang informiert über Gebühren, BTX-Institutionen, fertige BTX-Software, Hard- und Softwarehersteller. Ein Literaturverzeichnis und ein Verzeichnis der Abkürzungen und Fachausdrücke runden das Handbuch ab.

Die Prozessoren 68000 und 68008
Rechnerarchitektur und Sprache im NDR-KLEIN-Computer
von R.-D. Klein
1986, 543 S., 530 Abb., geb., DM 78,–
Franzis Verlag, München
Gliederung
Der NDR-KLEIN-Computer – Das Grundprogramm 68K – Versuche mit dem Grundprogramm – Pascal/S – Gosi, ein Compiler für den 68000/68008 – Betriebssysteme und Disketten-Laufwerke – Die Baugruppen – Anhang
Bemerkungen
Dieses Buch ist für Leser gedacht, die bereits Kenntnisse der Mikrocomputertechnik mitbringen; ihnen bietet das Buch eine Fülle von Informationen über Hardware, Software und dazugehörige Peripherietechnik der Prozessoren 68000 und 68008. Als Hilfsmittel zum Zugang zur 16-Bit-Technologie dient dabei der NDR-KLEIN-Computer, ein modularer Computer, der durch den Einbau neuer Baugruppen stets auf dem aktuellen Stand der Technik steht. In Kapitel 7 des Buches werden alle Schaltpläne, Baugruppen, ihr Aufbau und Test besprochen, so daß ein Selbstbau dieses Computers möglich ist.
Zu Beginn lernt der Leser den Umgang mit dem Grundmenü und dem Texteditor, dann erfolgt eine schrittweise Einführung in die Assemblerprogrammierung des 68000/8 mit Programmlistings und Übungsaufgaben. Erst nachdem so Verständnis für die Funktionsweise der Prozessoren geschaffen wurde, folgt ein Kapitel über die Programmierung in den höheren Programmiersprachen Pascal/S und Gosi. In Pascal/S – der Befehlsschatz stellt eine Teilmenge der Pascal-Befehle dar – werden die anspruchsvollen Aufgaben „Türme von Hanoi" und das Simulationsprogramm „Mondlandung" aufgearbeitet. Gosi (von „Grafisch orientierte Sprache I") ist eine Logo-ähnliche Compilersprache, deren Compiler die Programme in 68000/68008-Assemblerprogramme übersetzt.
Das Kapitel über Diskettenlaufwerke und Betriebssysteme gibt zunächst allgemeine Informationen, dann geht es tiefer in die Materie: Mikrodos, ein einfaches Disk-Operating-System, wird als Assemblerlisting dargestellt und erläutert. Im Anhang findet der Leser das Assemblerlisting des Universal-Formatieres, Befehlsübersichten über die Befehle des Grundprogramms und die Pascal/S-, Gosi- und 68000/68008-Befehle. Es folgen Schemazeichnungen der verwendeten ICs, ein Literatur-, Fachwort- und Stichwortverzeichnis.

Anwenderbuch CP/M-68K
Einstieg in das Betriebssystem für die 68000-Familie
von J. Plate
1986, 144 S., kart., DM 48,–
Franzis Verlag, München
Gliederung
Eigenschaften und Dienste des CP/M-86K – Texteditor ED – CP/M-68K- Assembler und Hilfskommandos – Dynamic Debugging Tool DDT-68K – CP/M-68K-Schnittstelle – Das BIOS – Der C-Compiler – BIOS-Listing für NDR-KLEIN-Computer – Sachwortverzeichnis
Bemerkungen
Dieses Handbuch bietet den Benutzern des Systems die wichtigsten Systeminformationen und erleichtert den Einstieg in das Diskettenbetriebssystem CP/M-68K. Es kann das Systemhandbuch nicht ersetzen und gibt auch keine Einführung in die Programmierung des 68000 oder in die Programmiersprache C. Dafür gibt es andere Bücher, die sich ausschließlich mit diesen Themen befassen und genannt werden.
Das Anwenderhandbuch beschreibt zunächst alle Kommandos, ihre Aufrufstruktur und Bedeutung und erklärt dann den Umgang mit dem Texteditor. Auf dieser Basis ist die Entwicklung eigener Programme mit dem Assembler AS68 und dem DDT-68K-Debugger möglich. Hintergrundinfor-

mationen liefern die Kapitel über die CP/M-68K-Systemschnittstelle und das BIOS; als Beispiel für ein BIOS wird im Anhang das vollständige BIOS für den NDR-KLEIN-Computer aufgelistet.

Logo auf dem Spectrum
von E. Hölscher
1986, 92 S., kart., DM 24,–
Dr. Alfred Hüthig Verlag, Heidelberg
Gliederung
Grundlagen – Einfache Prozeduren – Ein Mathematik-Programm – Verbesserungen – Statistik – Der Speicher – Der Drucker – Grafik und Mathematik – Selbstdefinierte Grafikzeichen – Mathematische Funktionen – Anhang, Sachwortverzeichnis
Bemerkungen
Dieses Buch vermittelt dem Leser, der nur die Grundlagen aus dem Logo-Handbuch mitbringen muß, die Struktur und die vielfältigen Möglichkeiten der Programmierung mit Logo. Das verwendete Sinclair-Logo unterscheidet sich nur wenig von der ursprünglichen Logo-Idee und eignet sich deshalb besonders gut zur Einführung in diese Sprache, die dank ihrer Anschaulichkeit schon von Kindern erlernt werden kann. Mit diesem Buch kann sich der Leser solide Kenntnisse in Logo aneignen. Von dieser Basis aus gelingt es ihm schnell, eigene Programmideen zu entwickeln und in die Tat umzusetzen.

Was Sie über Software wissen sollten
Die Handhabung der Software in Wirtschaft und Industrie
von F. Haugg
1985, 215 S., geb., DM 38,–
Franzis Verlag, München
Gliederung
Einführung – Problemstellung – Was man wissen muß, um den Computer zu verstehen – Der Einsatz der Software: Personalverwaltung, Lohn- und Gehaltsabrechnung, Bürosysteme, CAD/CAM, Ingenieurmäßige Berechnungen, Prozeßrechner – Wie es mit der Software weitergeht – Stichwortverzeichnis – Literatur
Bemerkungen
In seiner Einführung beschreibt das Buch auf amüsante Weise eine Situation, die typisch ist für viele Entscheidungsträger in Wirtschaft und Industrie: Ein Manager entschließt sich für den Computereinsatz zur Lösung seiner Aufgaben und erlebt dabei zahlreiche Probleme, Lösungen und neue Probleme. Was der Software-Benutzer wissen sollte, um in Zusammenarbeit mit Softwarehäusern die jeweils optimale Software zu erhalten, erfährt er aus diesem Buch. Von der Problemstellung über Programmiersprachen, Betriebs- und Datenbanksysteme bis zum Software-Engineering und der Qualitätssicherung werden hier Zusammenhänge verständlich gemacht und Begriffe erklärt. Dann werden die Möglichkeiten des Software-Einsatzes in verschiedenen Bereichen der Wirtschaft erläutert. Den Abschluß bildet ein Kapitel über Roboter und künstliche Intelligenz. Dieses Buch stellt für den angesprochenen Personenkreis eine einfach verständliche Einführung in den Softwarebereich dar. Wer sich detaillierter über Softwarekonzeptionen für bestimmte Anwendungsprobleme informieren möchte, muß allerdings nach spezialisierter Literatur greifen.

Sybex Ratgeber dBASE II
von Dr. G. Renner
1986, 355 S., geb., DM 55,–
Sybex Verlag, Düsseldorf
Gliederung
Allgemeine Einführung – Dateien, Felder, Variablen – Syntax, Ausdrücke und Operatoren – Mathematische Funktionen – Datums-, Zeitfunktionen – Konvertierungs- und Testfunktionen – Befehle zur Erstellung und Bearbeitung von Datenbank-Dateien – Befehle zur Bearbeitung anderer Dateien – Auswahl eines Datensatzes – Hinzufügen von Daten – Editieren von Daten – Ausgabe von Daten – Hilfsinformationen – Speichervariable – Programmierung – Geräte-Steuerung – System-Parameter – Anhang
Bemerkungen
Das Buch wurde für Anwender und Programmierer des Datenverwaltungsprogramms dBASE II geschrieben, die schnelle Detail-Informationen zu bestimmten Problemen benötigen. Der Ratgeber, dem die deutsche Version 2.41D von dBASE II zugrunde liegt, versteht sich als Nachschlagewerk und Ergänzung zum dBASE-II-Handbuch. Dementsprechend ist der Aufbau des Buches nach Anwenderbereichen gegliedert: Nach einer allgemeinen Einführung in Leistung und Sprachelemente werden die Funktionen und Befehle von dBASE II einzeln besprochen. Die einheitliche Gliederung in Aufgabenbeschreibung, Syntax, Anmerkungen, Beispiel und Querverweise erleichtern die Handhabung des Nachschlagewerks. Jedem Sprachelement wird in der Kopfzeile eine Funktionsgruppe und ein entsprechendes Symbol zugeordnet. Innerhalb dieser Funktionsgruppen sind die Sprachelemente alphabetisch geordnet.

Das Referenz-Manual für den Applesoft-BASIC-Programmierer
1986, 352 S., Spiralheftung, DM 69,50
Addison-Wesley Verlag (Deutschland), Bonn
Gliederung
Vorwort – Allgemeine Informationen – Variablen und Arithmetik – Die Steuerung des Programmablaufs – Strings und Arrays – Ein- und Ausgabe – Grafik – Spezielle Applesoft-Befehle – Anhangskapitel: A: Zusammenfassung der Befehle – B: Syntaxdefinitionen – C: ASCII-Zeichencodes – D: Reservierte Wörter – E: Fehlermeldungen – F: PEEKs, POKEs und CALLs – G: Programme effektiver machen – H: Internes über Applesoft – I: Ausgabeformate über Zahlenwerte – J: Der Bildschirmeditor – K: 40/80-Zeichen-Darstellung – L: Unterschiede zu Integer BASIC – M: Betrieb mit einem Cassettenrecorder – N: BASIC und die Maus – Glossar – Index – Referenztabelle
Bemerkungen
Die deutsche Ausgabe des amerikanischen Originals wurde von Mitarbeitern der Firma Apple Deutschland erweitert und auf den neuesten Stand gebracht. Das Manual baut auf den Apple-Benutzerhandbüchern und dem Applesoft-Tutorial auf und ist als Handbuch und Nachschlagewerk für die Programmiersprache Applesoft-BASIC gedacht. Erfahrene Programmierer werden v.a. auf die vollständige Beschreibung aller Applesoft-Befehle in Anhang A und die Referenzkarte zurückgreifen. Die Referenztabelle besteht aus einem abtrennbaren Karton mit einer Übersicht über den Applesoft-Sprachumfang: darin sind die wichtigsten Applesoft-Befehle und Funktionen, Variablendefinitionen, Programmoperationen, die arithmetischen, logischen und relationalen Operatoren aufgeführt. Im Referenz-Manual werden alle Befehle von Applesoft zusammen mit ihren Gültigkeitsbereichen, syntaktischen Definitionen und Funktionsbeschreibungen erläutert. Jeder Befehl wird durch ein Kurzbeispiel verdeutlicht. Hinweise auf Ausnahmebedingungen, die im Applesoft Tutorial nicht erwähnt wurden, werden hier aufgeführt. Querverweise und Kurzinformationen am Textrand ergänzen die Befehlsbeschreibungen, besondere Hinweise werden grau unterlegt. Das Referenz-Manual ist zwar nicht als Lehrbuch konzipiert, doch sind die Kapitel nach Funktionsbereichen gegliedert und behandeln inhaltlich zusammengehörende Applesoft-Befehle. Die Anhangskapitel F, G und H sind für fortgeschrittene Programmierer interessant: Anhang F beschreibt Syntax und Funktion der wichtigsten speziellen Funktionen des Apple, die man von einem Applesoft-Programm aus durch PEEK, POKE oder CALL aufrufen kann. Anhang G befaßt sich mit der Optimierung von BASIC-Programmen (Speicherplatzersparnis und Ausführgeschwindigkeit). Details der inneren Operationsweise von Applesoft erfährt der Leser aus Anhang H: Die Speicherbelegung des Apple, der Aufbau der Variablentabelle, die Belegung der Zero-Page und die Liste der Applesoft-Tokens werden hier behandelt. Die im Text verwendeten Fachausdrücke werden im Glossar erklärt.

Applesoft Tutorial
1986, 303 S., Spiralheftung, DM 89,50
Addison-Wesley Verlag (Deutschland), Bonn
Gliederung
Einführung: Programmieren lernen – Einführung in Applesoft – Elementare Programmierung – Korrektur von Programmen – Grafik – Strings und Arrays – Anhang A: Zusammenfassung der Befehle – Anhang B: Reservierte Wörter in Applesoft – Anhang C: Fehlermeldungen – Anhang D: Hilfestellungen – Anhang E: Weitere Programme – Anhang F: BASIC und die Maus – Glossar – Index
Bemerkungen
Dieses Buch ist die deutsche Übersetzung der englischen Originalausgabe des Tutorials aus dem Hause Apple. Das Applesoft Tuto-

rial ist ein Einführungskurs in BASIC für den Apple IIe und IIc zusammen mit dem neuen Apple-II-Betriebssystem ProDOS. Es wendet sich an Programmieranfänger, die bisher nur das Benutzerhandbuch ihres Apple gelesen haben und jetzt die Programmiersprache BASIC kennenlernen möchten. Das Handbuch liefert *keine* vollständige Dokumentation von Applesoft oder den ProDOS-Befehlen. Aus Gründen der Übersichtlichkeit fehlen einige Dinge, die für einen Anfänger noch nicht unbedingt wichtig sind. Aufbauend auf dem Applesoft Tutorial findet man im Applesoft BASIC Programmer's Reference Manual eine genaue und vollständige Beschreibung aller Applesoft-Befehle, viele nützliche Programmierbeispiele, Tips und eine Reference Card. Alle ProDOS-Kommandos werden im Buch „BASIC Programmieren mit ProDOS" ausführlich beschrieben. Kapitel 1 des Tutorials erläutert einfache Befehle zur Ausgabe von Text, Berechnungen und Grafik. In Kapitel 2 werden Direkt- und Programmodus und die Abspeicherung von Programmen auf Diskette besprochen. Programmschleifen, Kommentare und die Formatierung der Ausgabe werden erlernt. Im nächsten Kapitel werden die verschiedenen Möglichkeiten zur Veränderung von Programmen behandelt. Kapitel 4 erklärt, wie man Grafik, Geräusche und Zufallszahlen erzeugt und Unterprogramme benutzt. Im letzten Kapitel werden Zahlenfolgen (Strings) und Felder (Arrays) angeschnitten.

Die Gestaltung des Applesoft-Handbuchs ist sehr übersichtlich. In der Fußzeile werden das aktuelle Kapitel und Thema angegeben. Am Textrand ist Platz für Ergänzungen, Zusammenfassungen, wichtige Hinweise und Bemerkungen des Lesers. Wichtige Stellen im Text werden am Rand besonders gekennzeichet. Am linken Textrand fassen Anmerkungen in kleinerem Schriftgrad wichtige Informationen zusammen und geben Querverweise. Beim späteren Nachschlagen oder bei der Suche nach wichtigen Befehlen und ihrer Syntax erweisen sich diese Stellen als besonders nützlich. Grau unterlegte Kästchen enthalten Wiedergaben von Bildschirminhalten oder wichtige Hinweise zu den Kapiteln.

Nach dem Abschluß inhaltlicher Einheiten schlagen die Autoren sogar Pausen vor, die dem Programmier-Neuling Zeit zum „Verdauen" des Stoffes geben sollen.
Am Ende jedes Kapitels folgt eine kurze Zusammenfassung, in der alle neuen Befehle, Fehlermeldungen, Fachtermini etc. aufgelistet werden.
Die mitgelieferte Begleitdiskette enthält alle im Buch ausführlich besprochenen Beispielprogramme. Darüber hinaus findet man 4 weitere Programme, die zusätzliche Programmiertips geben und z.T. Unterprogramme liefern, die in jedes Applesoft-Programm eingebaut werden können.
Das Tutorial wendet sich an BASIC-Einsteiger. Dieser Gruppe vermittelt es in leichtverständlicher Sprache die Grundlagen der BASIC-Programmierung und das Verständnis für die „Denkweise" des Computers.

Der Schuhlöffel zum Computer

von P. v. Eynern
1986, 160 S., zahlreiche Abb., geb., DM 22,–
Universitas Verlag, München
Gliederung
Teil 1: Die Anwendung: Erste Vorstellung und Probe – Lehrer ohne Schlaf und Launen – Die Druckerei im Computer – Helfer im Geschäftsbetrieb – Innenleben mit Nullen und Einsen – Speicher für Impulse – Ein wenig über das Programmieren – Unix- ein modernes Betriebssystem – Bildschirmzeichnen - C.A.D. Teil 2: Die Technik: Die wundersame Kraft – Elektrizität schafft Elektronik – Rechenmaschinen elektrisch bis elektronisch – Die logische Entscheidung – Speicherbausteine – Rechenwerke
Bemerkungen
Dieses Buch hat sich zum Ziel gesetzt, die „Schwellenangst" vor dem Umgang mit dem Computer abzubauen. Hauptursache für die distanzierte Haltung gegenüber dem Computer ist oft die Furcht vor dem Erarbeiten eigener Programme. Deshalb eröffnet der Autor dem Leser den Zugang zum Computer über die Anwendung fertiger Programme am Beispiel kommerzieller Lernprogramme. In einfacher Sprache vermittelt er die Grundlagen von Aufbau und Bedienung eines Computersystems, führt den Leser in die Welt des binären Rechnens und der Informationsspeicherung ein. Am Beispiel von BASIC-Befehlen wird das Erstellen eigener Programme erklärt. Der Autor erläutert, was Textverarbeitungssysteme, Menüs und Masken leisten können und stellt Anwendungen des Unix-Betriebssystems und CAD vor. Im zweiten Abschnitt werden die mathematischen und physikalischen Grundlagen der Computertechnik anhand zahlreicher Abbildungen und Schaltbilder erläutert.

Kaufen
Verkaufen
Kontakte knüpfen

Hüthig PUBLIKATION

PEEKER Börse

bietet Ihnen einen kostengünstigen und erfolgversprechenden Weg, mit einer Kleinanzeige an Ihr Ziel zu gelangen. Problemlos und schnell mit dem anhängenden Coupon.

PEEKER Börse **Anzeigenauftrag**

GELEGENHEITSANZEIGEN
KLEINANZEIGEN

Bitte veröffentlichen Sie in der nächstmöglichen Ausgabe für mich folgenden Text als

- ☐ private/nicht gewerbliche Kleinanzeige (mindestens 2 Druckzeilen, DM 5,50 je Zeile zuzügl. Mwst.)
- ☐ gewerbliche Kleinanzeige (mindestens 2 Druckzeilen, DM 11,– je Zeile zuzügl. Mwst.)
  Gewerbliche Anzeigen müssen wir aus wettbewerbsrechtlichen Gründen mit G kennzeichnen.
- ☐ unter Chiffre (Chiffre-Gebühr DM 6,– zuzügl. Mwst.)

Peeker-Börse
Anzeigen-Service
Dr. Alfred Hüthig Verlag
Postfach 10 28 69
D-6900 Heidelberg

Preise privat nicht gewerblich +Mwst.	Preise gewerblich +Mwst.
Zeilen	Zeilen
1 DM 5,50	1 DM 11,–
2 DM 11,–	2 DM 22,–
3 DM 16.50	3 DM 33,–
4 DM 22,–	4 DM 44,–
5 DM 27,50	5 DM 55,–
6 DM 33,–	6 DM 66,–
zuzüglich evtl. Chiffregebühr	

Eine Druckzeile umfaßt 32 Buchstaben. Die ersten Wörter Ihrer Anzeige drucken wir in **fetter** Schrift. Nach jedem Wort ein Kästchen freilassen. Jede angefangene Zeile wird als ganze berechnet. Falls Name/Anschrift/Telefon-Nr. in der Anzeige erscheinen sollen, tragen Sie bitte auch diese Angaben in die Kästchen ein. Bitte den Text mit Schreibmaschine oder mit Druckbuchstaben eintragen.

Anzeigenschluß für Heft 2/87 am 22.12.86

Rubriken:
- ☐ suche Hardware
- ☐ biete Hardware
- ☐ suche Software
- ☐ biete Software
- ☐ verschiedenes

Bei Angeboten:
Ich bestätige, daß ich alle Rechte an den angebotenen Sachen besitze.

1 Zeile
2 Zeilen
3 Zeilen
4 Zeilen
5 Zeilen
6 Zeilen

Meine Anschrift:

Name/Vorname Telefon

Straße PLZ/Ort Unterschrift

Schach dem Apple

Schachprogramm CHESS 7.0

getestet von Dr. Peter Abel

1. Gegenstand

Mit CHESS 7.0 von Larry Atkin (u.a. Coautor von CHESS 4.6, das 1977 in Toronto Computer-Schachweltmeister wurde) bietet die Firma Odesta das neben SARGON (Fa. Hayden) wohl bekannteste Schachprogramm für den Apple II+/e/c in der Preisklasse von DM 100,– bis DM 150,– an. Im Lieferumfang enthalten sind ein ausführliches Handbuch, das eigentliche Schachprogramm sowie 32 Schachpartien und 8 Schach-Problemstellungen, die auf der Rückseite der Programmdiskette abgespeichert sind und mit CHESS 7.0 nachgespielt bzw. gelöst werden können. Die Handhabung von CHESS 7.0 ist denkbar einfach, denn es werden lediglich zwei Cursor-Tasten und die Return-Taste (bzw. Paddle) für die menügesteuerte Eingabe aller Funktionen und Züge benutzt; dadurch ist es allerdings nicht möglich, Züge in der entsprechenden Schachnotation einzugeben.

2. CHESS 7.0 als Spielgegner

Zur Steuerung der Spielstärke von CHESS 7.0 stehen insgesamt 15 Spielstufen zur Verfügung (0-8 und A-F), durch die u.a. festgelegt wird, wieviele Halbzüge das Programm in der Regel vorausberechnen soll. Die für die einzelnen Spielstufen angegebenen Durchschnittszeiten für die Berechnung eines Zuges sind allerdings mit Vorsicht zu genießen: abhängig von der jeweiligen Stellung der Figuren können aus den erwarteten 5 Minuten durchaus auch 50 oder mehr Minuten werden. Deshalb kommen für ein Spiel gegen CHESS 7.0 – wenn man nicht gerade mehrere Tage Zeit für eine Partie hat – auch nur die Spielstufen 0-7 bzw. A-C in Frage.

Gerade für den Hobby-Schachspieler wird ein Spiel gegen CHESS 7.0 am Anfang in der Regel wenig erfreulich sein, wenn er sich nicht zur Sicherheit auf eine niedrige Spielstufe beschränkt. Er wird nämlich meistens sehr schnell durch Figurenverluste in einem quantitativen und qualitativen Nachteil sein, da CHESS 7.0 – wie viele andere Schachprogramme auch – strategisch primär auf Figurengewinn ausgelegt ist. Es nutzt jeden Zugfehler des Spielers, der für das Programm in wenigen Zügen zu einem Figurengewinn führt, erbarmungslos aus. Glücklicherweise gibt es dann die Möglichkeit, Züge zurückzunehmen und dafür andere, hoffentlich bessere Züge zu probieren, das Programm Zugvorschläge machen zu lassen oder einfach die Seiten zu wechseln. Und wenn nichts mehr hilft, dann gibt es immer noch den Geräte-Netzschalter!
In der (kurzfristig) quantitativ orientierten Spielstrategie liegt gleichzeitig die Chance, gegen CHESS 7.0 zu gewinnen, indem man sich für eine längerfristig ausgerichtete Strategie entscheidet, die vielleicht sogar mit einem Figurenopfer zum Erringen eines qualitativen Vorteils verbunden ist.

Für die Eröffnung des Spiels sollte man zumindest Grundkenntnisse in der Theorie besitzen, insbesondere dann, wenn man sich für eine der bekannten Standarderöffnungen entscheidet, da CHESS 7.0 auf eine Eröffnungsbibliothek zurückgreift. Hat man einmal das Endspiel einigermaßen ungeschoren erreicht, so kann man sich fast schon gratulieren, denn in diesem Teil hat CHESS 7.0 die geringste Spielstärke. Grund dafür ist die oft große Anzahl von Zügen, die nun vorausberechnet werden müssen, um eine optimale Spielstrategie zu finden. Im Endspiel kann man sich selbst aus einer vergleichsweise ungünstigen Position oft noch in ein Remis retten oder sogar den Sieg erringen.

Daß CHESS 7.0 durchaus besiegt werden kann, zeigt folgende Partie, die ich gegen CHESS 7.0 (Weiß) auf Spielstärke C gespielt habe: (Spielstärke C bedeutet durchschnittliche Rechenzeit pro Zug laut Handbuch 1-10 Min., vorausberechnet werden durchschnittlich 3-5 Halbzüge)

1. b3 Sf6 2. Lb2 d5 3. f4 Lf5 4. Sf3 e6 5. Sh4 Lg6 6. Sg6: g6: 7. e3 Sbd7 8. Le2 De7 9. Sc3 0-0 10. 0-0 De8 11. Sb5 Lc5 12. d4 Lb6 13. Dd3 a6 14. Sc3 Th4 15. a4 Dh8 16. a5 La7 17. h3 Dh6 18. Sd1 Th3: 19. h3: Dh3: 20. Te1 Th8 21. Sf2 Dg3+ 22. Kf1 Th2 23. Sd1 Dg2++

Möchte man eine Partie nicht zu Ende spielen, so kann man den Zwischenstand auf einer Diskette abspeichern und zu einem anderen Zeitpunkt weiterspielen, oder man läßt das Programm gegen sich selbst zu Ende spielen. Hat man genug vom Schachspiel gegen den Computer, so sollte man sich einen menschlichen Spielpartner (oder eine Spielpartnerin) suchen und CHESS 7.0 als reines Schachbrett und Protokollant für die Zugfolge benutzen. Wer Briefschach vorzieht, hat im Modus P die Möglichkeit, eine Stellung mit einer Rechentiefe von bis zu 23 Halbzügen analysieren zu lassen.

3. Analyse von Spielstellungen

Um nähere Informationen über die Spielstärke von CHESS 7.0 zu erhalten, wurden dem Programm mehrere Spielstellungen vorgegeben, für die CHESS 7.0 den nächsten Zug berechnen sollte. Es handelte sich dabei um Problemstellungen, in denen es nur *einen* optimalen Folgezug gibt, der auch für einen sehr guten Turnierspieler nicht unbedingt leicht zu finden ist.

Abb. 1

1. Analyse (s. **Abb. 1**): Partieschluß Rodriguez–Miyasaka (gespielt 1972 in Skopje). Obwohl Miyasaka (Schwarz) drei Bauern mehr hatte, war seine Stellung so unterentwickelt, daß er von Rodriguez (Weiß, am Zug) in vier Zügen matt gesetzt werden konnte.

Für die Lösung solcher Matt-Aufgaben besitzt CHESS 7.0 den Modus M, in dem die einzige richtige Lösung Te6:+ nach etwa 40 Minuten gefunden wurde. Aber auch in den Modi 6 (nach ca. 3 Min.), 7 (ca. 5 Min.), 8 (ca. 10 Min.), C (ca. 10 Min.), D, E und F (jeweils ca. 48 Min.) wurde die richtige Lösung gefunden.

Abb. 2

2. Analyse (s. **Abb. 2**): Matt-Aufgabe (in drei Zügen) von F. Metzenauer (Al Hamishmar, 1965).
Die einzige richtige Lösung La3 wurde im Modus M bereits nach weniger als 9 Minuten gefunden, was selbst für einen professionellen Löser von solchen Schachproblemen angesichts der Schwierigkeit dieser Aufgabe eine gute Leistung wäre. Die Spielstufen bis 8 und A bis C waren mit dieser Stellung allerdings überfordert und entschieden sich für Le1, Sd2+ bzw. Lc3. Lediglich mit den Spielstufen D (ca. 68 Min.), E und F (jeweils ca. 147 Min.) wurde die richtige Lösung gefunden.

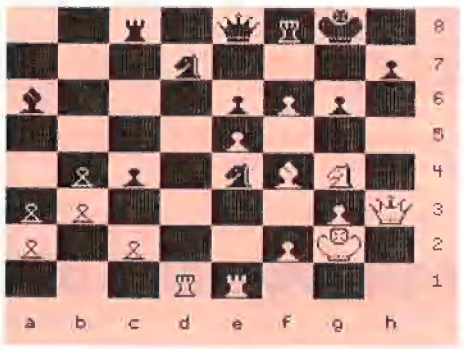


Abb. 3

3. Analyse (s. **Abb. 3**): Partieschluß Platz–Just (gespielt 1972 bei den DDR-Meisterschaften).
Ein Härtetest für CHESS 7.0, denn in diesem Fall besteht die optimale Lösung darin, daß Weiß (am Zug) den Gegner in acht Zügen matt setzt. Mit keiner einzigen Spielstufe wurde der optimale Zug Dh7:+ gefunden, sondern es wurde der weniger günstige Zug Sh6+ bzw. Te4: gewählt. Im Modus M wurde nach 2 1/2 Tagen die bis dahin erfolglose Suche abgebrochen.

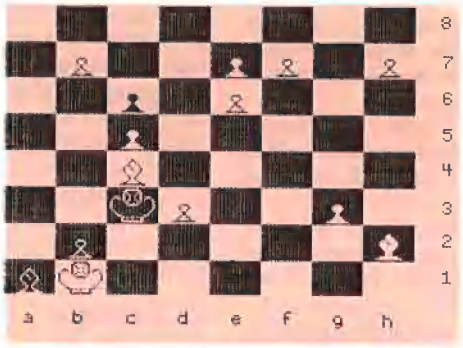


Abb. 4

4. Analyse (s. **Abb. 4**): Matt-Aufgabe (in vier Zügen) von F. Köhnlein (Münch. Neu. Nachr., 1903).
Die Lösung dieser Aufgabe besteht in einer Folge von Figurenumwandlungen (1. f8D! z.B. 1. ... Kb4 2. h8L Kc5: 3. b8T Kd6 4. e8S++). An dieser Stellung ist CHESS 7.0 gescheitert, da Unterumwandlungen von Figuren (z.B. in einen Springer) nicht berücksichtigt werden. Deshalb wurden von allen Spielstufen (einschließlich M) nur Lösungen gefunden, die zu einem Matt in fünf (oder mehr) Zügen führten.

Eine Reihe weiterer Analysen mit Spielstellungen, bei denen es genügte, vier bis sechs Halbzüge vorauszuberechnen, um einen optimalen Zug zu finden, zeigte, daß CHESS 7.0 durchaus in der Lage ist, ein „Matt in zwei oder drei Zügen" zu finden. Dies wurde nicht nur mit der Spielstufe M, sondern oft schon auf den Spielstufen 6-8 und C-F erreicht, wenn bei dieser Rechentiefe bereits eine optimale Zugfolge ersichtlich war.

Fazit

CHESS 7.0 ist vor allem ein Schachprogramm für den (fortgeschrittenen) Hobbyspieler. Er kann durch Spiele gegen CHESS 7.0 seine Spielstärke durchaus noch verbessern, da er v.a. lernt, Fehler zu vermeiden, die zu (unnötigen) Figurenverlusten führen. Unterstützung findet er dabei durch eine Reihe von Programm-Features wie z.B. der Möglichkeit, (schwache) Züge zurückzunehmen oder Zugvorschläge und Bewertungszahlen für die jeweilige Stellung abzurufen.

Der elektronische Buchhalter

Test des Buchhaltungsprogramms BUCH

von Dagmar Berberich

1. Notwendige Hardware

Die Firma Röntgen-Software hat mit ihrem Buchhaltungsprogramm BUCH ein professionelles Buchhaltungssystem erarbeitet, das im folgenden Testbericht vorgestellt werden soll. BUCH läuft auf Apple II+/e/c und Kompatiblen und auf IBM-Geräten und Kompatiblen. Für Benutzer eines IBM XT erläutert ein Anhangskapitel die Buchhaltung mit Festplatte. Folgende Konfiguration ist notwendig: 2 Diskettenlaufwerke oder 1 Harddisk und 1 Diskettenlaufwerk sowie ein Drucker zur Datenausgabe.

2. Das Programm

Vor Beginn der praktischen Arbeit wird zunächst eine *Sicherungskopie der Programmdiskette* erstellt. Aus Copyright-Gründen kann diese Kopie nicht zum Buchen verwendet werden; sie wird nicht als Originaldisk anerkannt. Im Falle eines Defektes kann die zerstörte Programmdiskette damit jedoch wiederhergestellt werden. Danach wird aus dem Eingangsmenü „Backup" der Punkt N angewählt und eine *Datendiskette* angelegt, auf die später die Buchungssätze geschrieben werden. Mit „#" kehrt man aus dem Backup-Menü zur Buchhaltung zurück.

2.1. Initialisierung

Die Initialisierung der Buchhaltung erfolgt über die Festlegung der Jahreszahl und des aktuellen Tagesdatums (Erstellungszeitpunkt). Danach erscheint das Buchhaltungsmenü (Hauptmenü), aus dem heraus alle weiteren Arbeitsschritte angewählt werden. Beim Erstgebrauch werden zunächst die verschiedenen, frei wählbaren *Konten* entsprechend dem bestehenden Kontenrahmen des Unternehmens angelegt. Wahlweise können Kontennamen oder Kontennummern angegeben werden.

Die verschiedenen Buchhaltungskonten werden in übergeordneten *Kontenklassen* zusammengefaßt. Die jeweilige Kontenklasse jedes Kontos wird durch die ersten Zeichen des Kontonamens festgelegt; eine Unterscheidung verschiedener Konten einer Kontenklasse erfolgt durch die Namensanhänge, die dem Kontokürzel folgen.
(Bsp.: Kürzel aller Anlagekonten ist das „%"-Zeichen, die verschiedenen Anlagekonten heißen demnach z.B. %Kfz, %Lagerhalle, %Maschinen usw.)

Folgende verschiedenen Kontenklassen und programminternen Abkürzungen werden in BUCH verwendet:
Geldkonten (Kürzel: POST, BANK, KASSE) umfassen alle liquiden Mittel, z.B. Kasse, Bank, Postscheck usw.
Anlagekonten (Kürzel: %): dazu zählen alle abgeschriebenen Maschinen, Gebäude usw., also alle Sachmittel.
Debitorkonten (Kürzel: $ oder DEBIT): Konten, die Forderungen enthalten.
Creditorkonten (Kürzel: # oder CREDIT) enthalten die eigenen Verbindlichkeiten.
Erloese (Kürzel: ERLOESE) enthält alle Erträge, die Mehrwertsteuer enthalten.
Mehrwertsteuerkonto (Kürzel: MWST): alle Umsatzsteuerbeträge der ERLOESE-Konten werden auf dieses Konto gebucht.
Vorsteuerkonto (Kürzel: VORSTEUER): Alle Umsatzsteuerbeträge der Ausgaben laufen automatisch auf dieses Konto.
Privatkonten (Kürzel: PRIVAT) enthalten Privatentnahmen und Privateinlagen.
Erfolgskonten (Kontennamen *ohne* vorangehendes Kürzel): hier werden Erträge und Aufwendungen verbucht.
Gewinn- und Verlust-Konto (Kürzel: G&V): Konto, auf dem alle Ausgaben und Einnahmen bei der Gewinn- und Verlust-Rechnung verbucht werden.
Sonderkonten (Kürzel: &): alle Konten, die nicht in die übrigen Kontenklassen einzuordnen sind.

Die Kennzeichnung der verschiedenen Kontenklassen kann auch nach eigenen Vorstellungen umdefiniert werden.

Bei der *Konteneröffnung* werden vom Programm der Kontoname bzw. die Kontonummer und die Überträge von Soll und Haben (bei Fortsetzung der Buchhaltung auf einer neuen Diskette) erfragt und nach eventuellen Korrekturen abgespeichert. Werden keine weiteren Konten mehr angelegt, kehrt man ins Hauptmenü zurück. Werden Kontennummern (max. 4stellige Angaben) gewählt, ist ein Aufruf des Kontos wahlweise über die Angabe seines Namens oder seiner Nummer möglich.

2.2. Buchungen

Die eigentlichen *Buchungsvorgänge* werden über Punkt B des Hauptmenüs angewählt. Das Programm lädt die eingegebenen Konten in den Speicher und erstellt ein Buchungssatzformular, das für jeden Buchungssatz auszufüllen ist. Darin werden für jede Buchung Tag, Monat, Belegnummer, Buchungstext, die angesprochenen Konten, der Betrag und der Umsatzsteuersatz eingetragen. Bei fortlaufenden Buchungen werden die Belegnummern automatisch erzeugt; sie können jedoch auch vom Benutzer eingegeben werden. Vor der Absicherung der Buchungssätze können Korrekturen durchgeführt werden.

Sehr nützlich ist die Möglichkeit, sich zusätzlich zur Maske des Buchungssatzes eine Liste aller eröffneten Konten auf dem Bildschirm anzeigen zu lassen.

Beträge, die inklusive Umsatzsteuer gebucht werden, müssen mit einem vorausgehenden Minuszeichen eingegeben werden; das Programm berechnet dann automatisch die im Betrag enthaltene Steuer und belastet das entsprechende Steuernkonto damit. Nettobeträge werden entsprechend ohne Vorzeichen eingegeben.
Einnahmen müssen über das Konto „Erloese" laufen, damit die Mehrwertsteuer richtig verbucht wird. Bei anderen Konten wird die Umsatzsteuer automatisch als Vorsteuer verbucht.

In einem Buchungssatz können max. 49 Buchungen aneinandergereiht werden. Nach dem Ende der Buchungseingaben werden alle Buchungen nach dem Datum geordnet und auf dem angeschlossenen Drucker ausgegeben. Vorgemerkte Buchungen werden immer ausgedruckt. Diese Ausdrukke gelten als Belege für vorgenommene oder korrigierte Buchungen. Die Geldbewegungen auf den Geldkonten Kasse und Bank werden registriert und die Gesamtveränderung zur Nachkontrolle mit ausgegeben. Jetzt können nochmals Fehler in den Buchungssätzen korrigiert werden, bevor sie endgültig auf Diskette geschrieben werden.

2.3. Betriebsübersicht

Werden keine weiteren Buchungen gewünscht, kann aus dem Hauptmenü der Punkt „Betriebsübersicht" angewählt werden. Nach Eingabe des Anfangsmonats liefert das Programm eine Übersicht bis zum aktuellen Datum, in der Verbindlichkeiten, Forderungen, Erträge, Aufwendungen, Anlagevermögen und Geldbestand aus allen Konten und Buchungen errechnet und dargestellt werden. Ausgabemedium ist wahlweise der Bildschirm oder der angeschlossene Drucker. Nach einem beliebigen Tastendruck wird eine *Umsatzsteuerberechnung* für den gewählten Zeitraum erstellt, in welcher der steuerpflichtige Umsatz mit vollem und halbem Steuersatz, die direkt gebuchte Mehrwertsteuer, Vorsteuer und Kürzungsbeiträge aufgeführt werden. Daraus wird die Umsatzsteuerschuld errechnet. Nicht berücksichtigte Umsatzsteuerbuchungen (Vorsteuer, Haben und Mwst, Soll) werden getrennt aufgelistet.

Um eine umfangreiche Darstellung aller *Kontenbewegungen* zu erhalten, wird die Option S aus dem Hauptmenü gewählt. Wählt der Be-

nutzer auf die Frage „Nur Summen (J/N)" ein N, so erhält er ab dem eingegebenen Eingangsmonat eine Übersicht über die Kontenblätter aller geführten Konten. Mit dem Befehl „Konto ausgegeben" ist die gleiche Kontenübersicht für jedes einzelne Konto abrufbar. Das Kontoblatt enthält alle Buchungen des jeweiligen Kontos.
Wird nur die Ausgabe der Summen gewünscht, erhält man eine Übersicht mit den Salden aller Konten.

Das *Journal* wird auf der Datendiskette im Laufwerk 2 als Datei abgespeichert. Nach Eingabe von „J" im Hauptmenü kann es mit sämtlichen darin befindlichen Buchungen auf Drucker oder Bildschirm aufgelistet werden. Für jede Buchung werden Nummer, Datum, Erläuterungstext , Betrag, Prozentsatz der Umsatzsteuer und die angesprochenen Konten dargestellt.

Die *Gewinn- und Verlust-Rechnung* bucht die Salden aller Erfolgskonten auf das Gewinn- und Verlustkonto; sie wird üblicherweise nur am Ende des Geschäftsjahres aufgerufen. Da die Gewinn- und Verlust-Rechnung durch explizite Buchungen das Journal erweitert, fragt das Programm vor der Durchführung nochmals nach, ob alle Konten im Journal abgeschlossen werden sollen.
Nach Eingabe des Buchungstages wird die Gewinn- und Verlust-Rechnung durchgeführt und auf Drucker oder Bildschirm protokolliert. Die Buchungen werden in das Journal abgespeichert.

3. Handhabung

Der gesamte Programmablauf ist über Menüpunkte des Hauptmenüs ansteuerbar. Deshalb ist die Handhabung des Programms sehr einfach und auch für Anwender, die bisher wenig oder gar nicht mit Buchhaltungsprogrammen gearbeitet haben, schnell zu erlernen. Alle Eingaben von Benutzerseite aus sind in Groß- oder Kleinschrift, mit oder ohne Leerzeichen möglich.
Allerdings wünscht man sich manchmal ein etwas besser aufgearbeitetes Ausgabeformat, bei dem die einzelnen Tabellenspalten übersichtlicher voneinander getrennt und besser hervorgehoben wären.

Das etwas unprofessionell anmutende, spiralgeheftete Manual besteht aus einem 71seitigen Ausdruck eines NLQ-Druckers im DIN-A-4-Format. Leider beeinträchtigen sprachliche Mängel das ansonsten recht ordentliche Manual: „Menu" und „formattierte Disketten" sind im Handbuch nahezu durchgehend vertreten.
Das Manual ist in drei Hauptabschnitte gegliedert.
Auf 30 Seiten wird zunächst eine ausführliche Einführung in den Umgang mit dem Programm gegeben, bei der alle Unterpunkte des Hauptmenüs erläutert werden. Im zweiten Hauptteil werden auf weiteren 34 Seiten die Kommandos nochmals im einzelnen besprochen. Der letzte Teil befaßt sich mit möglicherweise auftretenden Problemen und Fehlermeldungen.
Eine Anleitung, mit der vorgegebene Parameter und Systemdefinitionen verändert werden können, gibt dem Benutzer die Möglichkeit, das Programm stärker auf die eigenen Bedürfnisse abzustimmen.
Leider fehlt ein Stichwortverzeichnis im Anschluß an das Manual.

4. BUCH und das Finanzamt

Das Buchhaltungsprogramm BUCH entspricht den Regeln der ordnungsgemäßen Buchführung.
Alle Buchungsvorgänge müssen korrekt und eindeutig nachvollziehbar sein, also schriftlich festgehalten werden. Dabei ist die Form nicht entscheidend; Computerausdrucke sind ebenso akzeptabel wie herkömmliche Unterlagen.
Für alle Buchungsvorgänge gilt, daß sie nicht nachträglich korrigierbar sein dürfen. Diese Forderung wird von BUCH erfüllt. Nach dem Abspeichern der Buchungssätze auf Diskette ist keine Änderung mehr möglich. Fehler, die erst zu diesem Zeitpunkt aufgedeckt werden, müssen durch Stornobuchungen ausgeglichen werden.

Selbstverständlich gilt auch bei der Buchhaltung mit Buchhaltungsprogrammen die Regel „Keine Buchung ohne Beleg", d.h. für alle Buchungen müssen eindeutige Belege vorliegen.

Am Ende eine Geschäftsjahres werden alle Konten und das Journal ausgedruckt; diese Buchhaltungsunterlagen sind sorgsam aufzubewahren.

Ist man im Zweifel darüber, ob das zuständige Finanzamt die Form der EDV-verarbeiteten Buchhaltungsunterlagen akzeptiert, sollte man das anhand von einigen Testausdrucken direkt mit dem Finanzamt klären.

Fazit

Alles in allem erwirbt man mit BUCH ein Buchhaltungsprogramm, das ordentliche Arbeit leistet und für den professionellen Einsatz geeignet ist. BUCH erlaubt mit seinen Funktionen „Betriebsübersicht", „Summen Konten" und der Möglichkeit, den Stand einzelner Konten bis zu einem beliebigen Stichtag abzurufen, einen guten Überblick über den Stand der Buchhaltung.

Bei einem Preis von DM 660,– (inkl. Mwst) für Programmdiskette und Handbuch kann es im Vergleich zu anderen professionellen Buchhaltungsprogrammen noch als recht preiswert gelten.

Der Speedloader 2.0/R für Apple II+/e/c

Ein Erfahrungsbericht

von Willfried Wienholt

In einer Zeitschrift stolperte ich vor ca. einem Jahr über eine seitengroße Anzeige. Dort wurde der Speedloader als ein Programm angepriesen, das nur DM 59.– kosten sollte und Erstaunliches versprach:

- Lädt über 10x schneller als DOS 3.3
- Sehr einfach im Gebrauch
- Files können auf die RAM-Karte geladen werden
- Mehrere Files können gleichzeitig geladen werden
- Filewahl über eingebautes Menü möglich
- Ausführliche Gebrauchsanweisung

Soweit die Werbung. Aber was stimmt davon? Bei den heutigen Preisen für Softwareprodukte kommen doch Bedenken: Nur DM 59.–, was kann das wohl sein? Neugierig geworden bestellte ich ein Exemplar gegen Vorkasse über Bankanweisung und hielt knapp 14 Tage später das Erhoffte in den Händen.
Mittlerweile ist ca. ein Jahr vergangen, und ich möchte im folgenden meine Erfahrungen mit dieser Software schildern.

1. Gegenstand

Der Speedloader 2.0/R ist ein Softwareprodukt, das dem Anwender erlaubt, seine DOS-3.3-Disketten mit einem Schnelladeverfahren zu versehen, das Boot- bzw. Ladevorgänge erheblich verkürzt. Zu diesem Zweck werden DOS-Files von der jeweiligen DOS-Diskette auf eine „Fastbootdiskette" (FB) kopiert, die zusätzlich einen Schnellader enthält.
Die Lieferung umfaßte ein Manual im Format DIN A5 mit 14 Seiten Umfang und eine beidseitig beschriebene 5.25"-Diskette, die in einem Pappkarton gegen Transportschäden geschützt wurde.
Die mitgelieferte Systemdiskette ist kopiergeschützt. Der Schutz ist so ausgefeilt, daß bisher alle Nibble-Kopierer versagt haben. Dennoch braucht man keine Bedenken zu haben, seine Originaldiskette zu verwenden. Im Falle einer Beschädigung (z.B. RESET während des Bootvorgangs) sendet man die Systemdiskette wieder ein. Das Manual weist ganz zu Anfang auf die damit verbundene Kostenregelung hin. Kürzlich habe ich mir das Original „abgeschossen". Es half nichts mehr, ich mußte die Originaldiskette nebst Handbuch einsenden. Knapp zwei Wochen später erhielt ich die Diskette, meine alte und zusätzlich eine neue Anleitung zurück, und zwar umsonst! Die Anleitung hat jetzt 5.25"-Format und einen Umfang von 21 Seiten. An vielen Stellen wurde sie überarbeitet, Fehler wurden behoben, inhaltlich wurde aber nichts wesentlich Neues hinzugefügt.

2. Manual

Das mitgelieferte Manual ist übersichtlich und ausführlich geschrieben. Vor der Arbeit mit dem Speedloader sollte man es in Ruhe lesen. Auf den ersten Seiten begleitet das Manual den Anwender parallel zum Programm durch die einzelnen Menüpunkte. Jeder Menüpunkt wird ausführlich erklärt und ist leicht zu verstehen; auf Besonderheiten des Programms wird hingewiesen. An diese Kapitel schließen sich Installationshinweise für die Umwandlung von DOS-3.3- und Diversi-DOS-Disketten in Fastbootdisketten an. (Diversi-DOS gehört zum Lieferumfang des Speedloaders.) Möchte man DOS 3.3 auf eine FB übertragen, hilft ein Programm auf der Systemdiskette weiter.
Am Ende des Manuals wird auf Besonderheiten unter BASIC eingegangen, z.B. auf die Bedienung des Schnelladers, Fehlerbehandlungen usw. Verschiedene Beispiele zeigen, wie man das Color-Demo-Programm der DOS-3.3-Master-Diskette oder die Assembler MERLIN und BIG MAC in eine Fastbootdiskette umwandelt.

3. Systemprogramm

Das Systemprogramm ist benutzerfreundlich, denn auch Fehlbedienungen führen nicht zu einem Systemabsturz. Hat man versehentlich etwas falsch gemacht, genügt ein kurzer Blick ins Manual und man kann an der Stelle fortfahren, an der man zuvor steckengeblieben ist. Nach kurzer Eingewöhnungszeit kann man das Handbuch ganz zur Seite legen. Jeder Menüpunkt des Programms wird von zwei Statuszeilen am oberen und unteren Bildschirmrand begleitet. Die obere Statuszeile gibt Auskunft über den jeweiligen Arbeitszustand, in dem sich das Programm gerade befindet, während die untere Statuszeile dem Benutzer die erlaubten Eingaben bzw. bei Fehlbedienung die entsprechende Fehlermeldung anzeigt. Alle Angaben erfolgen im deutschen Klartext.

Die Möglichkeiten des Speedloaders sind vielfältig:
Wird eine Diskette mit dem Speedloader initialisiert (Menüpunkt 1), dann wird zusätzlich eine VTOC (volume table of contents, Inhaltsverzeichnis der Diskette) angelegt, die ein späteres Benutzen mit DOS-Kommandos (SAVE, BSAVE etc.) ermöglicht. Der Schnellader selbst ist eine Maschinenroutine, deren Position im Speicher zwischen $200 und $B300 frei wählbar ist. Zusätzlich kann man seine FB mit Diversi-DOS versehen; das schränkt den Speicherbereich des Speedloaders auf $900 bis $8900 ein.
Mit der Option COPY (Menüpunkt 2) werden DOS-Files auf die FB übertragen. Durch die Angabe einer Startadresse kann der Speicherbereich eines Binärfiles verändert werden, sofern dies gewünscht wird. Ebenso hat man optional die Möglichkeit, eine I/O-Adresse (zweimal) referieren zu lassen. Damit ist es z.B. möglich, die LC mit $C081 in den „write-enable-Zustand" zu versetzen und dann eine Maschinenroutine (Integer-BASIC) in die LC zu laden.
Bei der Erstellung einer FB wird durch den ersten Buchstaben des FB-Files festgelegt, wie dieser File beim Booten gehandhabt werden soll: Ein „." läßt den entsprechenden File *nicht* im Auswahlmenü erscheinen. Die mit einem „#" versehenen Files werden automatisch geladen, während die mit einem „>" gekennzeichneten Files auch automatisch ausgeführt werden. *Ein* File kann mit einem „*" versehen werden. Er wird dann als letzter File geladen und ausgeführt. Auf diese Weise läßt sich z.B. sehr einfach ein „turnkey-System" installieren.
Jederzeit kann man sich den Inhalt der Fastbootdiskette oder den CATALOG der DOS-Diskette mit den Menüpunkten 4 (Directory) bzw. 7 (CATALOG) ansehen. Ebenso können Files auf der FB umbenannt oder gelöscht werden. Ein weiterer Menüpunkt gibt Auskunft über den noch frei zur Verfügung stehenden Speicherplatz auf der FB.

4. Anwendungsbereich

Für oft benötigte Programme (z.B. Assembler wie BIG MAC oder MERLIN, Textverarbeitungsprogramme, Utilities oder Nibble-Kopierer) ist der Speedloader eine sinnvolle und praktische Ergänzung. Schnell und effizient werden die benötigten Programme nach der Konvertierung auf eine FB in den Speicher geladen. Zusätzlich ist die FB als ganz normale DOS-Diskette weiterverwendbar. Natürlich kann man auch sein Lieblingsspiel auf diese Weise „speedloaden". Der Boot- und Ladevorgang ist einfach wesentlich schneller, bequemer und bei längeren Arbeiten damit auch nervenschonender als bei nicht bearbeiteten Disketten.

5. Nachteile

Wie überall gibt es auch hier Schattenseiten. Die Anzahl der FB-Files ist nicht unbegrenzt, sondern auf maximal 15 beschränkt. Bei der Arbeit mit dem Speedloader ist es sinnvoll, sich vorher mit Papier und Bleistift zu überlegen, was auf die FB soll. Durch die relativ langen Filekonvertierungszeiten beim Kopieren von DOS zur FB gerät man schnell aus dem Konzept. Dieser Zeitaufwand trübt ein wenig den sonst so guten Eindruck.
Der schwerwiegendste Fehler bei diesem Softwareprodukt aber liegt in der Unveränderbarkeit der ein-

mal frei ausgewählten Startadresse des Speedloaders. Weil diese Startadresse nachträglich nicht verändert werden kann, muß für eine Neuinstallation des Schnellladers die gesamte Diskette neu formatiert werden. Möchte man nachträglich ein Programm als FB-File deklarieren, muß darauf geachtet werden, daß der File nicht aufgrund seiner Länge mit dem Schnellader kollidiert. Das System weigert sich sonst beim Bootvorgang strikt, den File zu laden. Einerseits ist dies verständlich, da der Schnellader sonst während des Ladens überschrieben würde, andererseits wird beim Kopieren nicht auf diese Kollision hingewiesen.
Ein nachträgliches Verändern oder Löschen von FB-Files ist möglichst zu vermeiden, weil dadurch auf der Diskette eine Lücke entsteht, die nur von einem anderen File geringerer oder gleicher Länge aufgefüllt werden kann. (Das Prinzip des Schnelladers beruht gerade darauf, daß er den entsprechenden File „am Stück" als FB-File auf der Diskette anlegt.)

Fazit

Für DM 59.– leistet der Speedloader sehr viel. Die Benutzerführung ist ausgezeichnet. Die Konvertierungszeiten sind teilweise relativ lang, aber das Ergebnis einer FB rechtfertigt diesen Aufwand. Die Angaben in der Werbung sind nicht zu hoch gegriffen, sondern halten, was sie versprechen. Allerdings sollte sich der Anwender immer vorher darüber im klaren sein, welche Files und auf welche Art er seine Dateien auf der FB unterbringen will, um eine Kollision mit dem Schnellader zu vermeiden. Dann allerdings hat er ein Werkzeug gefunden, das ihn bei seiner täglichen Routinearbeit wirkungsvoll unterstützt.
Bezugsquelle:
CBWS Productions, Rotterdam

Wo sind meine Files?

Test des Programms Multi-Disk-Catalog III

von Franz-Josef Hüskens

Besitzen Sie eine umfangreiche Programm- und Datensammlung? Durchsuchen Sie des öfteren Ihre Disketten nach bestimmten Files? Wenn ja, dann sollten Sie sich einen Katalog der Disketteninhaltsverzeichnisse anfertigen, um langwieriges sowie disketten- und nervenaufreibendes Suchen in Zukunft zu umgehen. Der Einfachheit halber sollte dies natürlich der Computer selbst erledigen.
Es eignen sich dafür allgemein gehaltene Datenbank-Programme wie dBase oder Quickfile. Diese Programme haben jedoch den Nachteil, daß alle zu speichernden File-Daten von Hand eingegeben werden müssen. Es gibt allerdings auch spezielle Disketten-Bibliothek-Programme. Diese haben den Vorteil, daß die relevanten Diskettendaten durch das Programm direkt von der Diskette gelesen werden. MULTI DISK CATALOG III (MDC III) von Sensible Software ist eines dieser Programme.

1. File-Klassifikation

MDC III liest innerhalb weniger Sekunden alle wichtigen File-Informationen wie Name, Typ und Größe von der Diskette ein. Zusätzlich wird die Volumenummer und die Anzahl der freien Sektoren gespeichert. Der Benutzer wird vor dem Lesen der Diskette aufgefordert, jede zu dokumentierende Diskette mit einem dreistelligen Identifikationscode zu versehen. Dieser Code muß nicht mit der Disk-Volumenummer identisch sein. Man kann also seine Disketten nachträglich mit einem eigenen System durchnumerieren. Zusätzlich kann für jeden File ein Klassifikationsfeld, worin der Anwendungsbereich (Spiele, Utilities usw.) zweistellig anzugeben ist, definiert werden. Anhand einer einzugebenden Suchmaske (s.u.) werden dann die ausgewählten Files angezeigt und die Eingabe der Ordnungselemente verlangt. Eine Liste von häufig verwendeten Programmbereichen und deren Klassifikationskürzel bietet hierbei Unterstützung.

2. Listen und Sortieren

Zum Listen der gespeicherten Dateien benutzt MDC III eine Suchmaske. Hierin können Suchkriterien für die maximal fünf Unterscheidungsfelder, die pro File intern angelegt werden, angegeben werden. Anhand dieser fünf Kennzeichen erfolgt die Anzeige der in der Disketten-Datenbank gefundenen Files.
Das Sortieren aller gespeicherten Dateien kann anhand von mindestens einem und höchstens drei der fünf Unterscheidungsfelder vorgenommen werden. So ist es z.B. möglich, innerhalb einer Diskette (spezifiziert durch die dreistellige Identifikation) nach der File-Klassifikation und darin nach Filenamen zu sortieren.

3. Ausdrucken

Mit der List-Funktion können die gespeicherten Files angezeigt bzw. ausgedruckt werden, wenn zuvor PR#1 eingegeben wurde. Aufgrund fehlender Steuerroutinen für den Drucker wird die Liste der Files durchgehend (ohne Format) gedruckt. Man kann jedoch mit einem eigenen Programm auf die gespeicherten „Bibliotheken" zugreifen und damit u.a. auch eine eigene Druckroutine schreiben. Im Handbuch zeigt ein dreiseitiges Programm-Listing in Applesoft-BASIC, wie man dabei vorgehen kann.
Zusätzlich erläutert das Handbuch auf vier Seiten die einzelnen Befehle, stellt auf zwei Seiten das Konzept der Suchmaske vor und erklärt auf einer Seite, wie man ein Treiberprogramm für ein nichtstandardisiertes Drucker-Interface in MDC III einbindet. Ein dreiseitiges Beispiel zeigt kurz und bündig, wie man das Programm benutzt. Die Handhabung von MDC III ist sehr einfach; das Handbuch reicht trotz seiner Kürze voll und ganz aus.

4. Speicherumfang

Im RAM dürfen maximal 950 Filenamen und die dazugehörigen Daten gleichzeitig gespeichert sein. Die „gesammelten" Files können auf einer beliebigen Diskette in maximal neun sogenannten „Master Files" (MF1 bis MF9) untergebracht werden. Die Kapazität von über 8500 dokumentierbaren Files dürfte für einen „normalen" Anwender ausreichen.

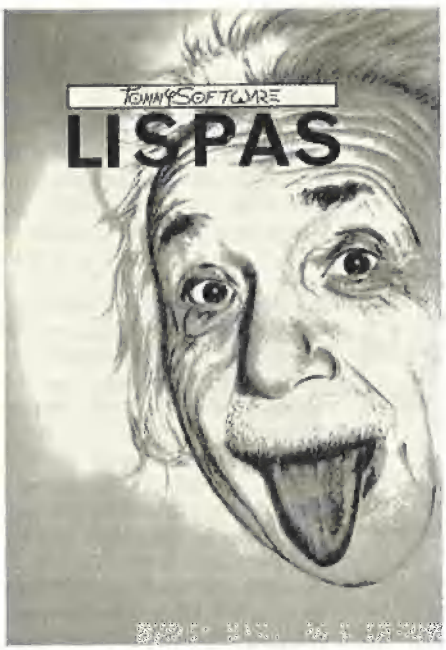
Fazit

Nachteilig ist bei dem Disketten-Dokumentationsprogramm, daß das Programm nur DOS-3.X-Files (3.1 bis 3.3) lesen kann. CP/M- und Pascal-Files können genausowenig gelesen werden wie Pro-DOS-Dateien. Damit ist das Programm für den Anwender von mehreren Betriebssystemen unbrauchbar.
Der Preis von ca. DM 100,– ist unter diesen Umständen etwas zu hoch, insbesondere deshalb, weil eine bessere Druckroutine wünschenswert wäre.

LISPAS II ST

Ein LISP-Interpreter für den Atari ST

getestet von Kai Oliver Tiffany



LISPAS II ist ein LISP-Interpreter, der seit August 1986 von der Firma TommySoftware in Frankfurt speziell für den ATARI ST angeboten wird. Der Interpreter läuft auf jedem ATARI ST mit mindestens 512K und einem beliebigen Laufwerk. Bei einem ST mit nur 512K sollten jedoch ROMs vorhanden sein.

1. Die Programmiersprache LISP

LISP ist eine Programmiersprache, die aus derselben historischen Situation entstanden ist wie FORTRAN, das vom Standpunkt der modernen Informatik wenig mehr als eine Katastrophe ist. LISP ist zwar bis heute fast nur im universitären Bereich verbreitet, wird dort allerdings oftmals sogar als erste Programmiersprache gelehrt. Dies ist nicht zuletzt der enormen und individuellen Kreativität zu verdanken, die durch LISP angeregt wurde: Sowohl die Zahl der implementierten LISP-Dialekte als auch die Zahl der auf der Grundlage von LISP entwickelten KI-Sprachen gehen mittlerweile ins Dreistellige. Es wird deshalb ein interessantes Schauspiel geben, wenn mit Rechnern der Kapazität des Atari ST erstmals Mikrocomputer zur Verfügung stehen, die LISP-Programmieren sinnvoll erscheinen lassen. Schauspiel deshalb, weil die Mikrocomputerszene neben der der Großcomputer eher bunt zusammengewürfelt ist.

2. Der LISPAS-Sprachumfang

Der Sprachumfang von LISPAS II bietet etwa 140 Standardfunktionen, die alle fest programmiert sind und nicht z. T. in LISP definiert gebootet werden müssen. Laut Handbuch handelt es sich bei LISPAS II um einen InterLisp und FranzLisp ähnlichen Dialekt. Der Benutzer kann auf einfache Weise eine Standardfunktion über einen eigenen Namen aufrufen, ohne den Umweg über eine selbstdefinierte Funktion zu wählen, womit man den LISP-Dialekt in großen Teilen ohne Aufwand und Geschwindigkeitsverlust an eigene Wünsche anpassen kann. Interessant ist, daß es in LISPAS kein „GO in PROG“ gibt, an seine Stelle tritt eine REPEAT-Funktion, die mit beliebig vielen UNTIL-Bedingungen etwa die Möglichkeiten des LOOP-Konstrukts der Sprache MODULA 2 bietet. Ein Pretty-Printer ist fest in die Sprache eingebaut. Selbstdefinierte Funktionen können sowohl „nospread“ als auch „spread“ sein. An die Stelle von „DE“ und „DF“ tritt in LISPAS „DEFUN“; NLAMBDA-Parameter können einzeln durch QUOTE deklariert werden (Ein Beispiel: (DEFUN f ('Text x y) ...)). An Datentypen gibt es 32-Bit-Integer und beliebig lange Strings.
Nachteilig zu bewerten ist das Fehlen der Floating-Point-Arithmetik. Sie soll jedoch in der Anfang 1987 herauskommenden Version LISPAS III ST implementiert sein, die allen registrierten Benutzern zu Selbstkosten zugeht.

3. Vom Umgang mit LISPAS

Da LISP eine sehr interaktive Sprache ist, muß die GEM-Schnittstelle hier besonders hervorgehoben werden. Im Gegensatz zum LISP des Softwarehauses Metacomco wurde hier der Weg beschritten, dem Benutzer das Einarbeiten in die GEM-Programmierung durch möglichst kraftvolle Befehle unnötig erscheinen zu lassen. LISPAS II kann mit Textfenstern, Alertboxen und der Menüzeile umgehen. Das Schöne ist, daß von der Definition eines Textfensters bis zum Bedrucken des geöffneten Fensters nur zwei LISP-Befehle benötigt werden. Um beispielsweise eine neue Menüzeile zu definieren, muß nur ein LISP-Befehl gegeben werden.

Die folgenden vier Beispiele zeigen das sehr anschaulich:

```
(TEXTWINDOW 'MeinFenster 3Ø 2Ø)
(MAPC (WINDOWS) 'CLOSEWINDOW)
(MAKEMENU '((Titel1 Punkt1 Punkt2 ...)
(Titel2 ...) ... ))
(OKBOX '(Wollen Sie wirklich alles löschen?)
'(Ja Nein Abbruch))
```

Das Anwählen eines Fensters oder z.B. eines Menüpunktes geschieht vom LISP aus über LISP-Atome, die gleichzeitig auch als Fenstertitel bzw. Menüpunktnamen erscheinen. Mausaktionen des Benutzers können relativ einfach überwacht werden, ohne daß der Event-Manager umständlich programmiert werden muß, da die Event-Funktionen von LISPAS (KEYSTROKE und INTERACT) oft schon den fertigen LISP-Code zurückgeben, der z.B. zum Schließen eines Fensters nötig ist.

Der Interpreter selbst ist 87K groß und stellt den gesamten vorhandenen Arbeitsspeicher für LISP-Code zur Verfügung. Der Garbage-Collector von LISPAS macht kein Sweeping, bleibt also nicht wie die meisten LISP-Interpreter von Zeit zu Zeit für einige Sekunden für „Aufräumarbeiten“ stehen. Der Interpreter ist etwa 15% schneller als das als Shareware verbreitete XLISP 1.5.

Es gibt 21 Fehlermeldungen, die sämtlich mit einer POSTMORTEM-Funktion vom LISP selbst abgefangen werden können. Leider ist kein Tracing möglich, aber mit einer kurzen selbstgeschriebenen POSTMORTEM-Funktion, wie sie als Beispiel auf der LISP-Diskette mitgeliefert wird, kann man z.B. mit einem Tastendruck jederzeit das Programm anhalten, lokale Variablen betrachten und anschließend wieder in das LISP-Programm zurückzukehren.

LISP-Code kann in Textfenstern editiert werden (Cut, Copy, Clear, Paste). Unter den Demoprogrammen befindet sich auch ein kleines LISP-Programm, mit dem das Editieren mehrerer Funktionen in mehreren Fenstern erleichtert wird. Wählt man in diesem Programm vom Menü aus HELP, wird eine einfache Art von ELIZA geladen, mit dem der hilflose Benutzer einen kleinen Schatz von erklärenden Sätzen erfragen kann.

4. Handbuch und Service

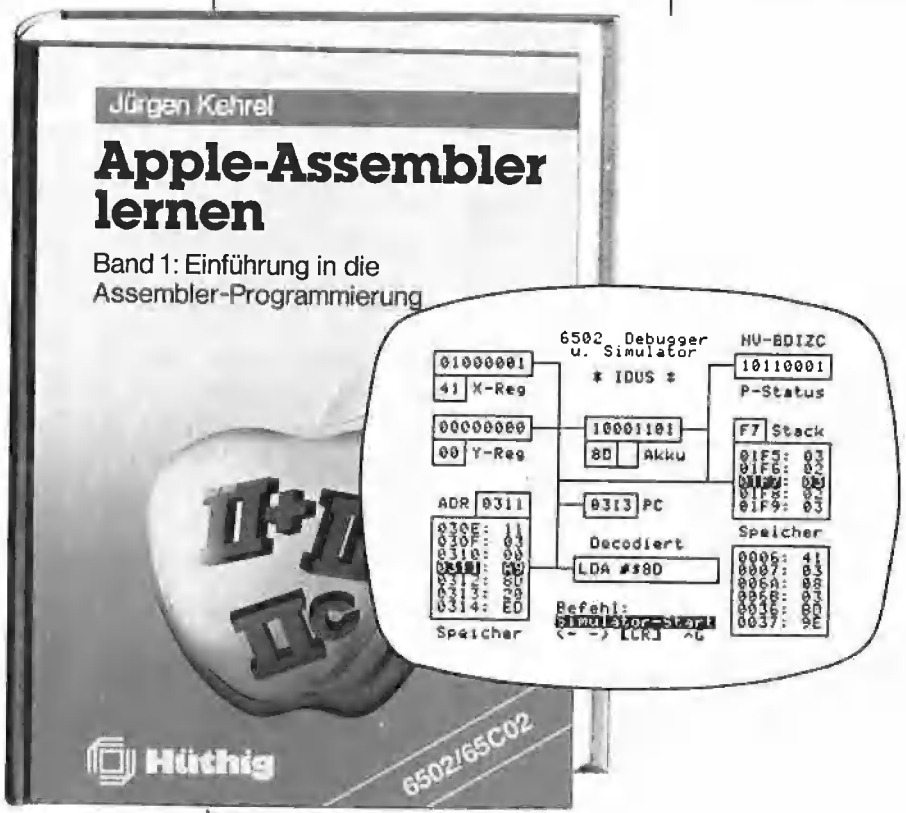
Das Handbuch ist etwas knapp geraten, dafür aber recht übersichtlich. Es gliedert sich in einen deutschen und einen englischen Teil. Der deutsche Teil enthält neben den alphabetisch angeordneten Funktionsbeschreibungen ein 8seitiges LISP-Wörterbuch, das einige englische LISP-Termini erklärt.
TommySoftware liefert auf der Diskette einen Textfile mit Informationen zum Programm mit. Darin wird u.a. darauf hingewiesen, daß die alte Version 1.1.0. beim Laden vom LISP aus dem Interpreter keine falschen Dateinamen verkraftet.

Noch ein Wort zum Thema Service: Neuere LISPAS-Versionen werden an Besitzer von LISPAS II gegen eine Kopiergebühr ausgeliefert. Die umfangreichere Version LISPAS III mit Floating-Point-Arithmetik, Grafikfunktionen und einem LISP-sensiblen Editor wurde für Anfang 1987 angekündigt.

5. Urteil

LISPAS II ist ein brauchbarer LISP-Interpreter. Zwar reicht die mitgelieferte Dokumentation nicht dazu aus, LISP zu erlernen, aber dafür ist LISPAS II durch seine Benutzeroberfläche und den LISP-spezifischen GEM-Zugriff ein sehr bedienungsfreundliches System. LISPAS II kostet DM 298,–; dieser Preis scheint mir für dieses Produkt durchaus vertretbar.

FACHBUCHTIP Hüthig



Apple-Assembler lernen

von Jürgen Kehrel

Band 1: Einführung in die Assembler-Programmierung des 6502/65C02

1985, 234 S., kart., DM 38,—
ISBN 3-7785-1151-3
Begleitdiskette zu Bd. 1:
DM 44,—
ISBN 3-7785-1243-9

Band 2: Nutzung besonderer Apple-Eigenschaften

1986, 276 S., kart., DM 38,—
ISBN 3-7785-1170-X
Begleitdiskette zu Bd. 2:
DM 44,—
ISBN 3-7785-1244-7

Das zweibändige Werk „Apple Assembler lernen" ist ein kompletter Kurs in der Assemblerprogrammierung des 6502 oder 65C02 auf dem Apple II, der nicht da aufhört, wo andere Einführungen auf „weiterführende Literatur" verweisen. Starkes Gewicht wird auf die praktische Anwendung gelegt. Deshalb gehört zum Kurs ein vollwertiger 2-Pass Assembler, der als einer von wenigen die erweiterten Befehle der neuen IIc und IIe Prozessoren 65C02 verarbeitet und der auch lange Programme von mehr als 1000 Zeilen in wenigen Sekunden übersetzt. Zu seinen professionellen Eigenschaften gehören neben 14 Pseudo-Opcodes, die die Arbeit mit Strings und Tabellen zu einem Kinderspiel werden lassen, ein Zeileneditor mit viel Komfort und die Fähigkeit, Quellcode in verschiedenen Formaten zu schreiben und zu lesen. Ein interaktiver Debugger und Simulator hilft Ihnen, eigene und fremde Maschinenprogramme zu verstehen. Mit seinen vielfältigen und mächtigen Möglichkeiten läßt er Sie hinter die Kulissen Ihres Rechners schauen. Sie können „sehen", was abläuft, Ihre Vorstellungskraft wird angeregt und nicht nur einfach Ihr Gedächtnis strapaziert.

Alle Programme sind 100% kompakter Maschinencode, nicht einfach compiliertes Basic. Selbstverständlich lernt der Leser sämtliche Maschinenbefehle des Apple und die wichtigen Grundalgorithmen. Aber auch der Umgang mit den eingebauten ROM-Routinen wird ausführlich geübt. Grafik, Sound Stringverwaltung, Fließkommaarithmetik werden ebenso behandelt oder das Zusammenwirken von Applesoft und Assemblerprogrammen.

BESTELLCOUPON

Bitte ausfüllen und an Hüthig Vertriebsservice, Postfach 1028 69 · 6900 Heidelberg schicken.

Buchtitel

Name

Straße

Ort

Unterschrift

Leserbriefe

Fragen und Antworten

Hardbreaker für Apple

Im Artikel „Hardbreaker für Apple II+ und IIe" (Peeker 9/86, S. 34 ff.) spricht Herr Porten unter Punkt 3 („Das Programm Hardbreaker") an, daß die LC möglichst blockiert werden sollte und verweist dabei auf die Schaltung in Abbildung 1. Daraus ist aber nur die Entprellung der Tastatur und die Aktivierung des NMI zu entnehmen. Ich besitze einen Kompatiblen mit auf der Hauptplatine integrierter LC. Ich möchte nicht auf der Hauptplatine herumbasteln. Könnte ich das Problem dadurch lösen, daß ich eine Experimentierplatine in Slot 0 stekke und DEVICE-SELECT mit +5V über einen manuellen Schalter verbinde? Ist dabei ein Kurzschluß möglich?

A. Lausch, Hildesheim

Antwort von W. Porten:

Wie Sie richtig bemerken, gibt die Abbildung 1 nicht die Schaltung zum Schutz der Language-Card (LC) wieder. Diese Schaltung habe ich dem Verlag zusammen mit der Schaltung zur Entprellung des NMI-Tasters zugeschickt, sie ist aber durch ein Versehen nicht veröffentlicht worden. Die Schaltung zum Schutz der LC muß folgendermaßen aussehen:

Auf der LC muß der vom Apple-Slot kommende DEVICE-SELECT unterbrochen werden. Mit einem Schalter kann diese Unterbrechung wahlweise wieder überbrückt werden, oder es wird die Leitung auf der Karte, auf der vorher der DEVICE-SELECT lag, mit einem Pull-UP (1k-Widerstand auf +5V) auf High gezogen. Es ist dabei aber wichtig, diese Unterbrechung vorzunehmen! Man darf den DEVICE-SELECT nicht einfach auf +5V ziehen, da sonst der Treiber im Apple beschädigt werden könnte. Da Sie einen Apple mit auf der Hauptplatine integrierter LC besitzen, müssen Sie diese Unterbrechung auch unbedingt auf der Hauptplatine vornehmen. Dies dürfte, falls Sie einen Schaltplan dieses Nachbaus besitzen, jedoch einfach sein. Wenn Sie andernfalls einfach eine Experimentierkarte in Slot 0 stecken und den DEVICE-SELECT dann auf +5V ziehen, kann es passieren, daß der Treiber-IC, welcher den DEVICE-SELECT generiert, durchbrennt.

CP/M 3.0 und ALS-Karte

Im Artikel „CP/M 3.0 beim Apple II" (Peeker 11/86, S. 34ff.) bemerken die Autoren auf S. 40, 2. Spalte, daß der Common-Bereich für die ALS-Karte bei $8000 beginnt. Ich besitze selbst jedoch eine Original-ALS-Karte, bei der dieser Bereich bei $E000 anfängt. Da ich annehme, daß zu dem Test eine Z80+-Karte verwendet wurde, so bedeutet dies, daß die Z80+-Karte (bzw. die verwendete Karte) doch nicht 100% kompatibel zur ALS-Karte ist! Also kann es durchaus vorkommen, daß ein für die ALS-Karte geschriebenes CP/M-3.0-System oder ein sonstiges Programm nicht auf der Z80+-Karte läuft. Dies dürfte insbesondere bei Grafik-Anwendungen der Fall sein, wie z.B. bei meinem Grafik-Paket, das ich unter Turbo-Pascal benutze.

Da der Common-Bereich der ALS-Karte bei $E000-$FFFF liegt, steht dem Programmierer, wenn er auf Bank 0 geschaltet hat, der Apple-Bereich $1000-$6FFF (auf den Z80-Adressen $8000 bis $DFFF) zur Verfügung. Das bedeutet, daß man auch, wenn man mit der Z80 arbeitet, auf den Grafikspeicher zugreifen kann. Im Gegensatz dazu muß man, wenn der Common-Bereich schon ab $8000 beginnt, extra auf die 6502 umschalten. Dies kann aber sehr zeitaufwendig sein, wenn man z.B. auf einer Hires-Seite malen will. 1000 Punkte gemalt bedeutet dann, daß man tausendmal auf die 6502 und zurück auf die Z80 schalten muß. Das von ALS geschriebene BIOS weist viele Mängel auf, wenn man es z.B. mit dem BIOS für den Basis 108 vergleicht:

– ALS nutzt die Möglichkeit der IO-Redirektion nicht. So kann man das Kommando DEVICE nicht benutzen, mit dem man jedem physikalischen Device, z.B. CRT (Bildschirm), V24 (serielle Schnittstelle), PARALL (paralleles Interface) die logischen Devices zuordnen kann (CONIN:, CONOUT:, AUXIN:, AUXOUT:, LST:). Es ist also beispielsweise nicht möglich zu sagen, daß der Rechner von nun an nicht nur Befehle von der Tastatur, sondern auch von der seriellen Schnittstelle annehmen soll. So wäre es auf einfache Weise möglich, den Rechner z.B. über ein Modem oder einen anderen Rechner zu bedienen.

– ALS nutzt die Möglichkeit des Multi-Sector-IOs nicht. Dadurch ist es beim Basis-108-CP/M-3.0 möglich, einen ganzen Track in einer Umdrehung zu lesen. Wie phantastisch dies ist, hat jeder schon gemerkt, der auf einem Basis unter CP/M 3.0 mit PIP Files kopiert hat und danach das gleiche z.B. unter CP/M 2.2 tat.

– Es ist noch eine Menge Speicher beim 6502 übrig, der nicht benutzt wird. Diesen freien Speicher könnte z.B. ein Drucker-Spooler ausnutzen, wie dies beim Basis-108-CP/M-3.0 gemacht wird.

Fazit: Die schönsten Möglichkeiten des CP/M 3.0 werden nicht ausgenutzt, weil diese vom BIOS nicht unterstützt werden. Schade!

W. Porten, Köln

Save-Befehl

Im gleichen Beitrag wird die Verwendung des Save-Befehls als umständlich beschrieben (Peeker 11/86, S. 38, Spalte 2). Leider wird dies auch im CP/M Plus User's Guide (Seite 5-85) bestätigt. Bei der Erläuterung des SID-Befehls (Seite 5-106) wird jedoch folgende Möglichkeit aufgeführt, eine Datei von SID aus abzuspeichern:

Befehle „Wfilespec", „Wfilespec, s,f" Inhalt: „write the contents of a contiguous block of memory to filespec, s is the start address, f is the finish address."

Beispiele: „Wa:turbo.com": Die Datei turbo.com wird in Laufwerk a: gespeichert, „Wb:test.txt,100, 300": Der Speicherbereich 100-300 wird in Laufwerk b: gespeichert.

Turbo-Pascal

Beim ALS-CP/M-3.0 können die Prozeduren Blockread und Blockwrite nicht verwendet werden, da Turbo-Pascal direkte BIOS-Aufrufe verwendet. Abhilfe für Turbo-Pascal 3.0:

1. Auf eine eigene Diskette folgende Dateien speichern:
– CP/M-Systemdateien SID.COM und SUBMIT.COM
– Turbo-Pascal: TURBO.COM
2. Erstellung der nachfolgenden Datei „TURBO.SUB" (z.B. mit dem Turbo-Pascal-Editor)
3. Aufruf durch „SUBMIT TURBO [Return]"
4. Die bisherige Datei TURBO.COM wird in ALTTURBO.COM umbenannt und die neue veränderte Datei als TURBO.COM abgespeichert.

```
; Name dieser Datei: TURBO.SUB
; Turbo-Pascal-3.0 BIOS-Patch
;
REN ALTTURBO.COM,TURBO.-COM
SID ALTTURBO.COM
<S1FEA
<ED
<43
<7
<1
<CD
<D
<1
<.
<S010D
<7B
<3C
<32
<5
<1
<E
<32
<11
<5
<1
<CD
<5
<0
<C9
<.
<WTURBO.COM
<G0
;
; Turbo-Pascal angepasst
```

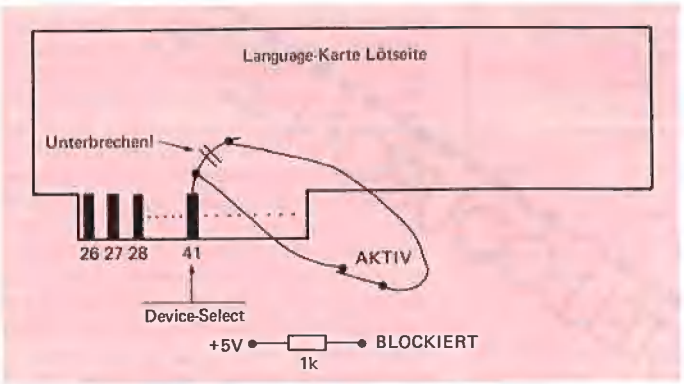
MBASIC und GBASIC

Neben den Grafikbefehlen funktionieren leider auch nicht VTAB und HTAB sowie alle direkten Zugriffe auf Speicheradressen, die in der anderen Bank liegen (z.B. Joystick).

J. Vievers, Niederkirchen

Drucker-Initialisierung unter CP/M 2.2

Ich besitze einen Apple-II+-kompatiblen Rechner mit einer Apple Serial Card plus 80-Zeichenkarte und Z80-Prozessor, mit dem ich CP/M 2.2 fahren kann. Dabei stehe



Schaltplan zum Schutz der Language-Card

ich vor folgendem Problem: Kann man den Drucker (Epson-FX-80-kompatibel) über CP/M mit einem bestimmten Zeichensatz, z.B. komprimierte Schrift, initialisieren, indem man z.B. das AUTORUN-Programm patcht, so daß es als Stapeldatei für Anwenderprogramme (z.B. Multiplan) benutzt werden kann? Oder gibt es eine andere Möglichkeit, vor dem Start eines CP/M-Anwenderprogramms automatisch den Drucker mit einem bestimmten Zeichensatz zu belegen?
T. Hochstetter, Siegburg

Zweiter Videx-Zeichensatz

Jörg Bliesener schlägt im Peeker 6/86, Seite 69 vor, zum Umschalten auf den zweiten Zeichensatz der Videxkarte die Folge „[ESC] (" zu verwenden. Nach meinen Erfahrungen ist dies bei manchen Videx-kompatiblen Karten erfolglos, weil sie aufgrund ihrer geänderten Firmware unter CP/M einige Steuerzeichen nicht erkennen, v.a. die CTRL-Z-Kommandos. Dieses Problem tritt vorwiegend bei Karten ohne Kabel zu Position F14 auf der Hauptplatine auf, für deren Betrieb vom Hersteller das Firmware-EPROM angepaßt wurde. Abhilfe kann hier mit einem direkten Zugriff auf das Inversflag der Karte geschaffen werden:

1. DDT starten
2. Diskette mit WS.COM einlegen
3. Zeile für Zeile abtippen:

```
IWS.COM
R
A2A4
JMP 2E0
JMP 2E9

A2E0
LDA F7FB
ORI 1
STA F7FB
RET
LDA F7FB
ANI FE
STA F7FB
RET

↑C (CONTROL-C)
SAVE 66 WSNEU.COM
```

Die Version WSNEU.COM schaltet jetzt jede Videx-kompatible Karte, die wenigstens intern das gleiche Inversflag benutzt, beim Start auf den zweiten Zeichensatz um und beim Verlassen mit „X" wieder zurück auf den ersten Zeichensatz.
D. Rackow, Wunstorf

Fehler in PRODOS.LIB

Im Programm PRODOS.LIB im Peeker 7/86, Seite 51, steckt ein schier banaler Fehler: In den Zeilen 976ff. soll ja getestet werden, ob BLEN = 0 ist, ggf. soll eine entsprechende Fehlermeldung erfolgen. Der Test auf Null ist einfach falsch, denn wenn bei der Folge „CLC, LDA BLEN, ADC BLEN+1" das Zero-Flag gesetzt wird, heißt das lediglich ((BLEN)+(BLEN+1)) mod 256 = 0. Im Längenbereich bis 32K gibt es 127 mögliche Werte für die Längenangabe, die eine irrtümliche Fehlermeldung produzieren. Mein Vorschlag für den Nulltest, der auch noch ein Byte kürzer ist, lautet:

```
BSAVE4    LDA BLEN
          ORA BLEN+1
          BNE BSAVE5
          LDA #$84
          ...
```

R. Pfitzer, Murrhardt

Proportionalschrift bei Appleworks

Mit Appleworks ist es möglich, auf dem Imagewriter einen beidseitigen Randausgleich bei der Proportionalschrift durchzuführen. Leider unterstützt das Anpassungsmenü von Appleworks diese schöne Möglichkeit nicht bei anderen Druckern (in meinem Fall Brother HR 15 XL). Wie kann man nun Appleworks klarmachen, daß ein anderer Drucker auch Proportionalschrift besitzt – wenn auch mit anderen Buchstabenbreiten als der Imagewriter? Da ich absoluter ProDOS-Laie bin, habe ich keine Ahnung, wie man das Problem angeht. Vielleicht können Sie dieses Problem in einem Artikel lösen oder mir konkrete Hinweise, Literatur oder ein kommerzielles Programm nennen, das mir weiterhilft.
M. Duttge, Hamburg

Apple-Atari-Kopplung

Ich bin Apple- und Atari-User (Apple IIc, Atari 260ST) und suche eine Möglichkeit der Datenübertragung von einem Rechner auf den anderen (ohne Modem oder Akustikkoppler). Wer kann mir helfen, z.B. mit Software oder einem Bauplan für ein Kabel?
S. Butz, Ebernburg
Antwort: Es gibt sowohl für den Apple als auch für den Atari ein Datenübertragungsprogramm namens KERMIT, das von amerikanischen Universitäten entwickelt und kostenlos erhältlich ist. Für den Atari befindet es sich beispielsweise auf den Atari-Entwicklerdisketten und für den Apple auf diversen Public-Domain-Disketten, z.B. von Firma W. Muhle, 2105 Seevetal 3, Waldwinkel 3. Hardwaremäßig benötigt man für den Apple eine Super-Serial-Card sowie ein Verbindungskabel zur bereits eingebauten seriellen Schnittstelle des Atari-Rechners. Die Peeker-Aufsätze und -Programme über den Atari werden zur Zeit noch auf diesem Wege vom Atari zum Apple übertragen und dann für die Setzerei kodiert.

ProDOS-1.0.1-Patch für 640K-Laufwerke mit Ehring-Controller – Erweiterung auf ProDOS 1.1.1

Lassen Sie mich zunächst noch einige Anmerkungen zum Patch von Herrn Hüneke (Peeker 11/85, Seite 29ff.) machen. Herr Hüneke hat mit seinem Patch alle am Markt befindlichen Laufwerkkonfigurationen von 1 x 40-Track- über 2 x 40-Track- bis hin zu 2 x 80-Track-Laufwerken unterstützt und mußte aus diesem Grund die ProDOS-RAM-Disk opfern. Ohne es genau zu wissen, nehme ich an, daß zum Einsatz von RAM-Karten (RAM-Erweiterungen > 128K) die Installation der ProDOS-RAM-Disk unumgänglich ist. Damit muß man sich derzeit für eine der beiden Alternativen entscheiden – entweder RAM-Disk oder 640K-Diskette. Würde Herr Hüneke auf die Unterstützung der 2 x 40-Track-Laufwerke verzichten, könnte er ohne weiteres 1 x 40-(1 x 35)Track-, 1 x 80-Track-, 2 x 80-Track-Laufwerke und die RAM-Disk unterstützen.

Dies geht, weil er sowieso einen 40/80-Track-Schalter zur Step-Umschaltung an den Laufwerken vorschreibt. Demzufolge muß ein ProDOS-Patch nur noch den „gewünschten Track" mit 80 vergleichen, um eine Kopfumschaltung vorzunehmen; der Modus „Kopfumschalten bei größer 40 Tracks"

```
ProDOS-1.1.1-Patch für 640K-Laufwerke mit Ehring-Controller

      $5000: D8          CLD
             20 BE D6    JSR $D6BE
*     $5004: 20 D0 D6    JSR $D6D0       ; Aufruf Blocktest
*            8E 56 D3    STX $D356       ; Tracknummer
*            B0 28       BCS $5034       ; Carry gesetzt: ERR
*     $500C: A5 46       LDA $46         ; Akku mit Block Nr.
*            A0 05       LDY #$05        ; low laden
*            0A          ASL
               ...          ...

      $50BC: D0 0E       BNE $50CC
             AC 6F D3    STY $D36F
*     $50C1: CC 56 D3    CPY $D356       ; Tracknummer
             F0 0F       BEQ $50D5
*     $50C6: AD 56 D3    LDA $D356       ; Tracknummer
             48          PHA
             98          TYA
               ...          ...

      $5109: 4C F7 D0    JMP $D0F7
*     $510C: EA          NOP
*            20 DB D6    JSR $D6DB       ; Track-Nr. prüfen
*            20 25 D1    JSR $D125       ; ggf. SL-Kopf-Um-
               ...          ...             schaltung

*** Blocktest ***

      $56D0: A5 42       LDA $42         ; Test Kommando Nr.
             C9 04       CMP #$04        ; = 4?
*            B0 04       BCS $56DA       ; Kommando >= 4
*            A6 47       LDX $47         ; Block Nr. (hi)
*            E0 05       CPX #$05        ; = 5? (>1280 Blk)
*     $56DA: 60          RTS             ; Carry gesetzt ->
                                           Fehler!

*** "Track gewünscht" prüfen ***

*     $56DB: A4 3E       LDY $3E         ; Slot Nr. x 16
*            C9 50       CMP #$50        ; Akku = Track gewünscht
*            B0 12       BCS #56F3       ; Track gew. >= 80
*            48          PHA             ; Slot Nr. gewinnen
*            98          TYA             ; "Kopfrückschaltung"
*            4A          ASL             ; 60 für Slot 6
*            4A          ASL             ; /10 =
*            4A          ASL
*            4A          ASL             ; 06 für Slot 6
*            09 C0       ORA #$C0        ; C6 für Slot 6
*            8E EE D6    STA $D6EE       ; C6
*     $56EC: 8D 00 00    STA $0000       ; $Cx00 für Slot x
*            68          PLA
*            18          CLC
*            90 08       BCC $56FB       ; immer
*     $56F3: 99 88 C0    STA $C088,Y     ; Kopfumschaltung
*            99 89 C0    STA $C089,Y     ; Motor Aus/Ein
*            E9 50       SBC #$50        ; "Tack gew." - 80
*     $56FB: 0A          ASL             ; "Track gew." * 2
*            8D 6F D3    STA $D36F       ; Akku = für
*     $56FF: 60          RTS             ; "Track gew." >= 80 ->
                                           ; ("Track gew." - 80)*2
                                           ; "Track gew." <= 79 ->
                                           ; "Track gew." * 2
```

entfällt ja. Für 35/40-Track- und 1 x 80-Track-Laufwerke entfällt eine Kopfumwandlung automatisch, da „Track gewünscht" nie größer 80 wird. Daraus ergibt sich die Einsparung der Drive-Tabelle und der „Bedienungssoftware" für die Tabelle, so daß – durch die Speicherplatzeinsparung – die RAM-Disk erhalten bleiben kann.
Nach dieser Theorie nun auch ein praktisches Beispiel, welches ich mit ProDOS 1.1.1 durchgeführt habe. Wenn Sie diesen Patch testen wollen, gehen Sie bitte wie folgt vor:

1. ProDOS 1.1.1 booten (Original)
2. ProDOS 1.1.1 mit „UNLOCK PRODOS" unlocken
3. ProDOS 1.1.1 mit „BLOAD PRODOS,TSYS,A$2000" laden
4. Mit CALL -151 den Monitor aufrufen und
5. ab $5000 alle mit „*" gekennzeichneten Änderungen eingeben; alle anderen Zeilen dienen nur der Orientierung und entsprechen dem Original.
6. ProDOS mit „BSAVE PRODOS,TSYS,A$2000" wieder abspeichern
7. ProDOS mit „LOCK PRODOS" ggf. wieder sichern
8. ProDOS neu booten.

Anmerkung: Der Bereich $D6.. entspricht $56..(unverschoben)!
Dieser ProDOS-1.1.1-Patch paßt gerade noch in ProDOS hinein, ohne die RAM-Disk-Routinen entfernen zu müssen. Nach Umschaltung meines Laufwerks 2 in den 40-Track-Modus (SS) arbeitet ProDOS genauso zuverlässig und fehlerfrei wie mit einem zweiten 2 x 40-Track-Laufwerk. Da in ProDOS 1.0.1 von $FEBE-$FEFF noch freier Platz vorhanden ist, müßte es möglich sein, die Routinen von Herrn Hüneke dort unterzubringen; ggf. stehen dann ab $FFEC-$FFF9 noch einige Bytes zusätzlich zur Verfügung.
J. Grimm, Zwiesel

Umwandlung in Großbuchstaben bei dBase II

Ich bin User von dBase II und habe das Problem der Umwandlung von Klein- in Großbuchstaben bei der Verwendung der GET-Befehle. Die Befehlsfolge:

```
erase
store " " to m
§ 10,10 get m picture "!!!"
read
return
```

liefert in der Variablen m bei der Eingabe von deutschen Kleinbuchstaben keine Großbuchstaben.
Mit dem Befehl „display memory" kommt als Antwort:
m (c) äöüß
usw. Wie kann ich dBase dazu bewegen, auch die deutschen Sonderzeichen richtig umzuwandeln? Gibt es wie bei Wordstar eine Tabelle, mit Hilfe derer dBase die Umwandlung vornimmt? Kann mir ein Peeker-Leser bei diesem Problem helfen?
Ch. Waller, Frankfurt

Leserbriefe zur Peeker-Umfrage

IBM hat Apple abgelöst

Kann man den technischen Reizen eines IBM-PC und seiner kompatiblen Verwandten auch zwiespältig gegenüberstehen, so handelt es sich dabei doch aufgrund ihrer Verbreitung und Einsatzgebiete wohl eher um die Rechnerfamilie, die den Apple in seinen vielfältigen Einsatzbereichen abgelöst hat.
Und hier tritt meiner Meinung nach die Problematik Ihrer redaktionellen Entscheidung zugunsten des Atari zutage. Peeker verband bisher sehr gelungen professionelle Anwendung und Hobby. Die Anwendbarkeit von Atari-Programmen und Rechnerinterna in meinem zukünftigen Berufsleben als Ingenieur in der Industrie ist wohl ziemlich begrenzt. Ich möchte Sie daher bitten, Ihre Entscheidung für Atari noch einmal ausführlicher in einem Artikel darzulegen und dabei auch aus Ihrer Sicht einmal vertieft darzustellen, was einem „innovationswilligen" Apple-II+-Anwender den Übergang auf den Atari schmackhaft machen könnte und warum er der IBM-Welt fernbleiben sollte.
T. Jacobs, Hamburg

IBM an Schulen

Ich hatte gehofft, daß sich Ihre Zeitschrift in Zukunft auch mit dem Betriebssystem MS-DOS beschäftigen würde. Leider stellte ich bei der Lektüre des September-Editorials fest, daß Ihre Zeitschrift in Richtung Atari erweitert wird. Die meisten Schulen (Gymnasien) in Nordrhein-Westfalen besitzen noch (!) Apple-Computer. Alle Neubeschaffungen und Neueinrichtungen sind Computer, die IBM-kompatibel sind. Für einen Atari gibt es von der Düsseldorfer Schulaufsichtsbehörde weder Unterstützung noch Zuschüsse.
Dr. G. Burri, Xanten

Flexibilität durch Slots

Auch Sie haben sicher bemerkt, daß sich seit geraumer Zeit die Tendenz weg von Apple durch die Geschäftspolitik dieser Firma ergibt. Trotzdem ist es doch nicht zu verstehen, daß eine Zeitschrift, die sich dem Apple verschrieben hat, nicht längst versucht hat, das Naheliegende zu tun: Über Steckkarten den Rechner so zu erweitern, daß er modernen Standards genügt. Der größte Vorteil des Apple waren immer seine Slots, die ihn erweiterungsfähig machen. Das haben Sie anscheinend vergessen. Damit haben Sie den gleichen Fehler gemacht wie die Firma Apple.
P. Messing, Lüdenscheid

Vom Apple zum Atari

Ich habe die Absicht, mir in den nächsten Tagen einen Atari zuzulegen, nachdem ich bisher mit einem Apple II+ programmiert habe. Ich würde mir daher eine deutschsprachige Zeitschrift mit einem ähnlich hohen Informationsgehalt wünschen wie den Peeker für Apple-Computer.
G. Grammel, Wiernsheim

Atari als Spielgerät

Der Ausgabe 9/86 des Peeker entnehme ich, daß sich die Zeitschrift für Atari öffnen wird, die MS-DOS-Maschinen (IBM und Kompatible) dagegen unberücksichtigt bleiben. Es ist anzunehmen, daß die Atari-Spezies da beginnt, wo wir als Apple(r) anfangs der 80er Jahre standen (ROM entdecken etc.), und daß sie den Computer weniger als Werkzeug, sondern als Spielmaschine und Hobbygerät nutzt, dem es Geheimnisse zu entlocken gilt.
O. Strasser, Kreuzlingen/Schweiz

Mehr Mac

Anbei übersende ich Ihnen die Umfragekarte, da ich einer der oft so geschmähten Macintosh-Besitzer bin und gern in Ihrer Zeitschrift mehr über den Apple Macintosh Plus lesen möchte. Ein bißchen verwundert mich oft, mit welcher Arroganz manche Apple-II-Besitzer über Mac-Besitzer schreiben. Noch nie habe ich einen solchen Leserbrief mit dem umgekehrten Sinn bei Ihnen gelesen. Ich schreibe doch auch keinem vor, welchen Computer er sich kaufen soll. Das muß doch jeder mit sich selbst ausmachen. Den „besten Computer" gibt es halt nicht, und so sollte sich jeder kaufen dürfen, was er will.

Ich fände es auf jeden Fall schade, wenn sich der Peeker nur als „Apple-IIe-Zeitschrift" verstehen würde.
N. Rothhaas, Santiago de Chile

Zunächst IIgs behandeln

Ich bin der Meinung, der Apple II ist so ergiebig, daß es der Peeker nicht nötig hat, sich opportunistisch an neuere Rechner zu hängen. Es gibt schon zuviele Zeitschriften, die sich an jeden Trend anhängen und von allem etwas und nichts Genaues bringen. Schauen Sie sich einmal Chip (die Bild-Zeitung unter den Computerzeitschriften), Heft 9/86 zum Thema Apple IIx an! Wenn nun der Apple IIx oder IIgs herauskommt, gibt es für den Peeker viel zu tun. Der Peeker sollte sich zunächst einmal diesem Thema zuwenden.
W. Oertling, Neumünster

Wen unterstützt Apple?

Ich möchte mich zu einer Sache äußern, die Herr Stiehl in letzter Zeit öfters in seine Berichterstattung einfließen läßt: Die Quälereien und Anstellereien bei der Firma Apple. Wenn Apple den Anspruch (qualitativ und preislich) erhebt, zu der gehobenen Schicht der PCs zu gehören, dann sollte man in München sowohl der Kundschaft als auch besonders einem Informationsmedium wie dem Ihren entgegenkommen und sich nicht selber ins Abseits bugsieren. Oder ist man bei Apple wirklich so borniert? Dieses Thema sollte in einem Erfahrungsbericht Ihrerseits, aber auch durch Erfahrungsberichte der Peeker-Leser gesondert in einer Peeker-Ausgabe aufgegriffen werden.
A. Grün, Krefeld

Aktuelle Computerbücher

1985, 305 S., 6 Abb., kart.,
DM 58, –, ISBN 3-7785-1150-5
Begleitdiskette DM 48, –
doppelseitig beschrieben
ISBN 3-7785-1290-9

Dieses Buch wendet sich als lehrbuchhafter Kurs an alle, die professionelle hochaufgelöste Grafiken auf dem Apple erzeugen wollen. Der erste Teil beginnt mit einem Abriß des Aufbaus der HGR-Seiten aus der Sicht des Programmierers. Danach wird das Programm HRCG (HI-RES Character Generator, Apple, Inc.) eingehend analysiert, und es werden sinnvolle Ergänzungen vorgestellt.
Schrittweise wird die Nutzung des HRCG erarbeitet bis hin zur beliebigen Bewegung eines statistischen Objekts auf einer der HGR-Seiten.
Der zweite Teil baut auf dem ersten auf und führt über die Definition mehrerer Objekte und simultaner Bewegung hin zu einem Arcade-Spiel, das für die meisten kauflichen Action-Spiele in der meisterhaften Grafik als Vorbild dienen kann. Grundkenntnisse in 6502-Assembler sollten vorhanden sein.

1985, 470 S., kart., DM 68, –
ISBN 3-7785-1134-3

„Die ProDOS Analyse" ist die umfangreichste und detaillierteste Darstellung, die jemals ein Apple-Betriebssystem erfahren hat.
Wer die „Innereien" von ProDOS bis zum letzten Byte, z.T. bis ins letzte Bit kennenlernen möchte, braucht dieses Buch. Das komplette Betriebssystem (Urlader, MLI, Disk-Driver, RAM-Disk-Driver und Uhr-Routine) mit Ausnahme des BASIC-SYSTEM wird mit umfangreichen Kommentaren und Übersichtstabellen disassembliert.
Auch die nicht im „Technical Reference Manual" aufgeführten Eigenschaften von ProDOS werden analysiert und beschrieben, z.B. die vertrackten eingebauten Testroutinen zur Identifikation der verschiedenen Apple II Modelle und *eventueller Nachbaugeräte*.
Programmierer, die ProDOS versionsabhängig „patchen" möchten, erhalten hier den genauen Überblick, wo was geändert werden muß, damit dies keine negativen Konsequenzen hat.

Arne Schäpers, Das BASIC SYSTEM, 1986, ca. 300 S., kart., DM 54, – ISBN 3-7785-1407-5
Sie wollen die erweiterten Möglichkeiten von ProDOS und dem BASIC-SYSTEM voll ausnutzen? Nur hier finden Sie

- *das Lesebuch:* eine schrittweise Erklärung der Zusammenarbeit zwischen Applesoft und dem Monitor, des Aufbaus der Stringverwaltung, mögliche Umleitungen der Ein-/Ausgabe und der Eingriffsmöglichkeiten in diese Abläufe. Die gezeigten Mechanismen werden durch zahlreiche (und sehr kurze) Beispielprogramme untermauert;
- *die Analyse:* eine minutiöse Sezierung des BASIC-SYSTEM zusammen mit der Kommandoschnittstelle zu ProDOS. Die Hauptfunktionen (Vektorbehandlung, TRACE-Kontrolle, FRE-Kommando, Global Page und benutzte Speicherstellen) sind jeweils in eigenen Abschnitten erläutert, ein kommentiertes Listing schließt diesen Teil ab;
- *Tips & Tricks:* die verborgenen Möglichkeiten des BASIC-SYSTEM („Input Anything", andere Dateinamen als STARTUP, RESET ohne CLEAR), der Aufbau „externer Kommandos". Dieser Teil enthält ein komplettes Rahmenprogramm für die Erstellung eigener Kommandos sowie die Erweiterung MONITOR, mit der (analog zu DOS 3.3) die Ausgabe von Kommandos und der Dateiverkehr auf dem Bildschirm sichtbar gemacht werden können.

☐ Schäpers, **Bewegte Apple Grafik**, ISBN 3-7785-1150-5, DM 58, –

☐ Begleitdiskette, DM 48, – ISBN 3-7785-1290-9

☐ Schäpers, **Pro-DOS-Analyse** ISBN 3-7785-1134-3, DM 68, –

☐ Schäpers, **Das BASIC-SYSTEM**, ISBN 3-7785-1407-5, DM 54, –

BESTELLCOUPON

Gewünschte Bücher bitte ankreuzen und an Dr. Alfred Hüthig Verlag, Postfach 102869, 6900 Heidelberg, schicken.

Name _____

Straße _____

Ort _____

Datum _____ Unterschrift _____

Hüthig

3 starke Bücher für Ihren Atari ST

Dieter und Jürgen Geiß

Logo auf dem Atari ST

1986, 146 S., DM 35, –,
ISBN 3-7785-1262-5

LOGO ist die erste Sprache auf dem Atari ST. Hier treffen sich die hervorragenden grafischen Fähigkeiten einer Programmiersprache und die überlegenen Leistungen des neuen Rechners. Das Atari-LOGO unter der Benutzeroberfläche des GEM verfügt über den zur Zeit größten LOGO-Sprachumfang und macht vollen Gebrauch von Fenstern, der Maus als Eingabegerät und den sogenannten Drop-Down-Menüs. Dieses Buch zeigt das Planen und Schreiben von faszinierenden und nützlichen Programmen. Das gesamte LOGO-System – nämlich Bedienung und Sprache – wird vorgestellt. Hier stehen die Antworten auf Fragen, die im Original-Handbuch offen geblieben sind. Der Leser lernt die gesamte LOGO-Sprache mit strukturierter Top-Down-Programmierung, Prozeduren, Rekursionen usw. Einige beispielhafte Projekte zeigen, daß LOGO weit mehr ist als eine anschauliche Lernsprache für Kinder.

Hajo Lemcke, Volker Dittmar und Michael Sommer

Programmierlexikon für Atari ST

1986, ca. 500 S., DM 48, –,
ISBN 3-7785-1412-1

Wie jedes Lexikon ist auch dieses vollständig nach Stichworten sortiert. Im Gegensatz zu einem normalen Lexikon findet der Leser hier jedoch nicht nur eine Beschreibung, sondern gleich eine Programmieranleitung. Ebenso sind mehrere Tabellen enthalten, die das Auffinden weiterer Stichworte erleichtern und zusätzliche Informationen beinhalten. Die meisten anderen Bücher enthalten entweder eine Dokumentation über die GEM-Fähigkeiten des Rechners oder die Beschreibung der einfachen Betriebssystemroutinen. Hier jedoch findet der Leser alles. Es gibt nicht nur Hinweise zur Programmierung von Dialogboxen, Fenstern oder Kommandointerpreten, sondern es werden auch alle systeminternen Fragen beantwortet. Dies umfaßt sowohl die Programmierung der im Rechner benutzten Chips, als auch eine Beschreibung der Schnittstellen und deren Benutzung. Es wird auf alle grafischen Möglichkeiten des ST eingegangen. Gleichgültig, ob nach den deutschen oder nach den englischen Begriffen gesucht wird, es sind alle vorhanden und verweisen gegebenenfalls aufeinander.

Software-Entwicklung auf dem Atari ST

1986, 388 S., DM 54, –, ISBN 3-7785-1339-7

In diesem Buch findet der ernsthafte Programmierer alles was er braucht, um gute und professionelle Software auf dem Atari ST zu entwickeln.
Der vollständige Arbeitsablauf sowohl in einer C- als auch in einer PASCAL-Umgebung wird beschrieben, die notwendigen BATCH-Programme zum Compilieren, Assemblieren und Linken sind gelistet.
Das Kapitel 3 beschäftigt sich mit der Entwicklung von reinen TOS-Programmen. An dieser Stelle werden sowohl das Betriebssystem und der Aufruf aller GEMDOS-, BIOS- und XBIOS-Funktionen als auch die Bedeutung der System-Variablen erklärt.
Kapitel 4 ist das Herzstück des Buches: die GEM-Programmierung. Alle Funktionen der beiden großen GEM-Bibliotheken (VDI, AES) werden behandelt.
Zwei komplette Sitzungen mit dem Recourse-Construction-Set werden in Kapitel 5 dargestellt.
Im Kapitel 6 werden dem Leser an zwei vollständigen, sehr sauber programmierten und kommentierten Beispielen mit einigen hundert Zeilen fast alle Probleme vor Augen geführt und gelöst, die bei der Fensterprogrammierung auftreten. Diese Programme können als Muster für eigene Applikationen oder Desk-Accessories benutzt werden.

☐ D. u. J. Geiß, **Logo auf dem Atari ST,**
ISBN 3-7785-1262-5, DM 35, –

☐ Lemcke, Dittmar und Sommer, **Programmierlexikon für den Atari ST,**
ISBN 3-7785-1412-1, DM 48, –

☐ D. u. J. Geiß, **Software-Entwicklung auf dem Atari ST,**
ISBN 3-7785-1339-7, DM 54, –

BESTELLCOUPON

Gewünschte Bücher bitte ankreuzen und an Dr. Alfred Hüthig Verlag, Postfach 102869, 6900 Heidelberg, schicken.

Name _____

Straße _____

Ort _____

Datum _____ Unterschrift _____

Hüthig