Nr. 11/86 November
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

PEEKER

MAGAZIN FÜR MIKROCOMPUTER

Disketteneditor
Assembler-Unit für UCSD
Formatierte Pascal-Listings
CP/M 3.0
MS-DOS für IIe
Mark Williams C-Compiler

Jetzt mit regelmäßigem Atari-ST-Sonderteil

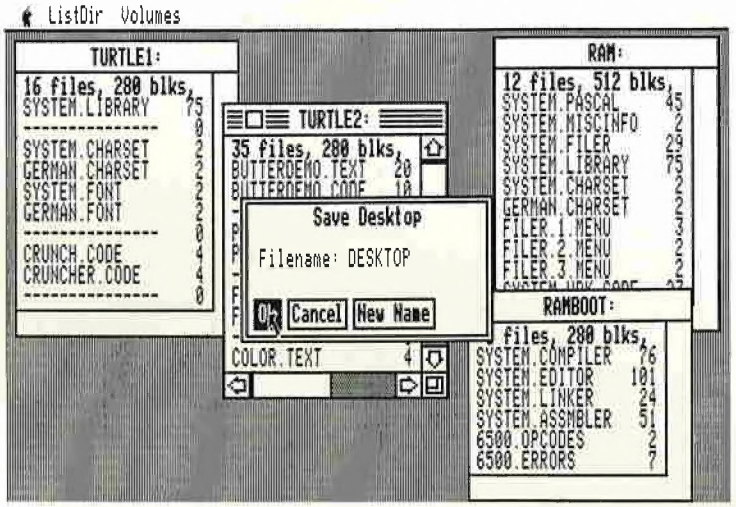
+ Großer Testbericht GFA-BASIC

Hüthig PUBLIKATION

TurtleGraphics-Library-Paket

von Dieter Geiß

Turtle-Utilities für Fenstertechnik und Apple-Maus in einfacher und doppelter Hires-Grafik für Pascal 1.1/1.2 auf Apple II+/e/c mit Maus oder Joystick.
2 Disketten mit umfangreichem Manual, DM 98,–



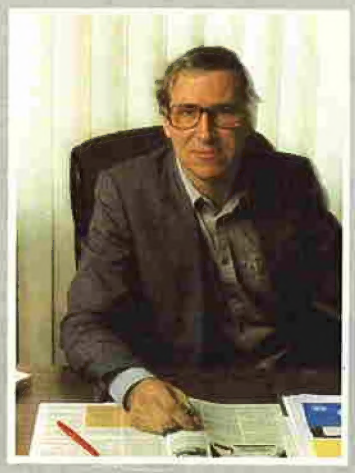
Das Utility-Paket besteht aus vier Modulen, die von Programmierern benutzt werden können, um professionelle grafische Anwendungsprogramme in Pascal zu schreiben.

Benötigt wird ein Apple Pascal Betriebssystem, entweder die Version 1.1 oder die neue Version 1.2. Bestehende Programme laufen ohne Einschränkung mit der neuen „TurtleGraphics", wenn diese nicht zu viel Speicherplatz verbraucht haben, da die neue „TurtleGraphics" umfangreicher als die alte ist. Für Fenstertechnik ist 128K-Pascal + Maus erforderlich.

Im einzelnen bietet das Paket folgende Möglichkeiten:

- volle Kompatibilität mit der alten „TurtleGraphics"
- lauffähig auf Pascal 1.1 und 1.2
- funktionsfähig mit angeschlossener UltraTerm-Karte
- alle zeitkritischen Funktionen in reinem Assembler programmiert
- Benutzung der zweiten Hires-Seite ($4000–$5FFF) möglich
- für Apple IIc und Apple IIe mit erweiterter 80-Zeichen-Karte Benutzung der doppelten Hires-Grafik mit 560 × 192 Punkten bzw. 16 neuen Farben möglich
- schnelle Prozeduren zum Zeichnen eines Punktes oder einer Linie
- Linearisierung von Teilen des Hires-Schirms
- Benutzung mehrerer Zeichensätze gleichzeitig
- Laden und Speichern von Hires-Bildern mit Ausdruck über Pascal-SUPERDUMP
- Scrolling des Hires-Schirms oder eines Teils in vier Richtungen
- drei verschiedene Schriftarten: Fett-, Breit- und Proportionalschrift, beliebig mischbar (acht Möglichkeiten)
- spezielle schnelle Ausgabe von Text
- Cursor bei Eingabe frei programmierbar
- Ein-/Ausgabe von INTEGER-, CHAR-, STRING- und REAL-Werten im Grafikmodus
- Menüzeile wie beim Macintosh (Die nachfolgenden Module benötigen Maus/Joystick) und 128K-Pascal
- Pull-down-Menüs
- Laden und Speichern von Fenstern (Windows)
- Öffnen von Fenstern
- Aktivieren und Deaktivieren von Fenstern
- Verschieben und Vergrößern/Verkleinern von Fenstern
- Scrolling von Fensterinhalten in allen vier Richtungen
- Umfangreiche Demos als Quelltexte.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg



EDITORIAL

Neues zum Apple GS

Ursprünglich sollte der Apple IIGS, der sich beim Booten übrigens mit „Apple IIgs" meldet, unmittelbar nach der Aussendung der Presserundschreiben am 15.9.86 erhältlich sein. Nunmehr wird es noch eine kleine Verzögerung geben, so daß größere Stückzahlen erst zum Jahresende zur Verfügung stehen werden.

Apple-II-Kompatibilität

Wir haben inzwischen eine Reihe von Kompatibilitätstests durchgeführt. Grundsätzlich muß man konstatieren, daß die Mehrzahl der Altprogramme auch auf dem Apple IIgs lauffähig ist. Dies gilt insbesondere für die für den Apple IIe mit neuen ROMs gedachten Programme, denn Software für den II+, die nicht auf dem IIe lief, wird auch auf dem IIgs nicht laufen, da der IIgs in dieser Hinsicht mehr dem IIe (mit neuen ROMs) als dem II+ oder IIc (mit alten ROMs) entspricht.

Auch wenn sich viele Altprogramme starten lassen, so heißt dies noch lange nicht, daß sie in allen Einzelheiten auf dem IIgs laufen. Dies gilt insbesondere dann nicht, wenn Programme von Monitorroutinen ($F800-$FFFF) Gebrauch machen. Beispielsweise funktioniert der spezielle Monitorbefehl unseres „Macroeditors" (M300L usw.) nicht auf dem IIgs; alle anderen Befehle scheinen zu funktionieren. Hier müßte man also mit einem Patch nachhelfen, denn die Monitorbefehle wurden beim IIgs erheblich erweitert. Ein weiteres Inkompatibilitätsbeispiel ist die Seite 2 ($0C00-$0FFF) des Textbildschirms, die sich nicht mehr mit $C055 oder entsprechenden Peeks einschalten läßt. Davon sind etliche Anwenderprogrammen tangiert, beispielsweise auch unser „Disk 40".

Accelerator und IIgs

Der IIgs ist theoretisch mit 2,8 MHz und praktisch mit etwa 2,5 MHz getaktet, während etwa die Accelerator von Titan mit theoretisch 3,6 MHz und praktisch mit 3,3 MHz gefahren wird. Deshalb wirbt Applied Engineering unter dem Konterfei von Steve Wozniak im Oktober-Heft der „Incider" damit, daß die neue 3,6-MHz-Transwarp-Karte, die in Deutschland über die Firma Weiß in Wilhelmshaven erhältlich ist, 40% schneller als der IIgs sei. Dies dürfte jedoch übertrieben sein. Einige Applesoft-Beispiele für die 3,3-MHz-Accelerator mit handgestoppten Zeiten (in Sekunden):

```
FOR X = 1 TO 1000:
Y = SQR(X): NEXT
Accel: 60s; IIgs: 74s.
FOR X = 1 TO 4000:
PRINT SQR(X);: NEXT
Accel: 93s; IIgs: 100s (in 80 Z/Z).
FOR X = 1 TO 8000:
PRINT X;: NEXT
Accel: 40s; IIgs: 36s (in 80 Z/Z).
```

Diese drei Beispiele zeigen deutlich: Solange überwiegend 6502-Befehle des auch im IIgs unveränderten Applesoft-Interpreters abgearbeitet werden, ist die Accelerator schneller als der IIgs. Überwiegen jedoch die 65816-Befehle des neuen Monitors, so ist der IIgs trotz der niedrigeren Taktfrequenz der Accelerator überlegen. Trotzdem kommt der 2,5-MHz-65816 natürlich niemals an die Leistungsfähigkeit eines 8-MHz-68000 heran.

IIgs- und Atari-Preise

Die Preise für einzelne IIgs-Komponenten wurden noch nicht festgelegt, doch liegen inzwischen wenigstens die Paketpreise vor. Danach muß man für das Grundgerät mit 256K RAM und einem 800K-Drive einschließlich eines Schwarzweißmonitors DM 3.990,- und einschließlich eines Farbmonitors DM 4.990,- bezahlen. Wie sich inzwischen herausgestellt hat, ist mindestens ein 800K-Drive unbedingt erforderlich, weil z.B. das 16-Bit-ProDOS auf einer 140K-Diskette gar nicht untergebracht werden konnte. Darüber hinaus dürfte auch eine Vielzahl der zu erwartenden IIgs-16-Bit-Programme mit 256K RAM nicht auskommen. So läßt sich beispielsweise das ProDOS-Entwicklungsprogramm bei einem 256K-Rechner erst gar nicht starten. Dies heißt jedoch, daß man zu den DM 4000,- für das Grundgerät noch mindestens DM 1000,- für die Speichererweiterung hinzurechnen muß. Damit dürfte eine Minimalkonfiguration des IIgs für unter DM 5000,- kaum erhältlich sein.*

Ganz anders verfährt hier die Firma Atari, die den Preis für den Atari 1040 ST mit 1 Megabyte RAM, 720K-Drive und Schwarzweißmonitor mit Wirkung ab 1.10.86 auf DM 2.500,- und somit auf den halben Preis einer vergleichbaren IIgs-Konfiguration gesenkt hat, nachdem vor kurzem der neue Schneider-PC mit 512K RAM, 360K-Drive und Schwarzweißmonitor zu einem Preis von DM 2.000,- vorgestellt worden ist.

Ulrich Stiehl

* Wie wir während der Drucklegung erfahren haben, hat man sich inzwischen eines anderen besonnen und wird voraussichtlich eine 256K-Erweiterung in den Paketpreis einschließen.

INHALT

Impressum

Peeker
3. Jahrgang 1986
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 11/1986

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 72,–, einschl. MwSt
und Versandkosten.
Jahresabonnement Ausland DM 72,– plus DM 18,–
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: Postgiro-
konten: Ludwigshafen 4799-673,
BLZ 545 100 67; Österreich: Wien 7555888;
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 Heidel-
berg 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

Disk-Monitor

Ein Disketteneditor für DOS 3.3

von Georg Maaß

Am Anschluß an den Disketteneditor EDIT (Peeker, Heft 5/86 ff.), der nur für ProDOS gedacht ist, wird mit dem „Disk-Monitor" nunmehr ein komfortabler Editor für DOS 3.3 vorgestellt.

1. Gerätevoraussetzung

Für das Funktionieren dieses Programms wird ein Apple IIe mit 80-Zeichenkarte benötigt. Das Programm läuft also nicht auf einem II+. Von der 80-Zeichenkarte wird nur die Textseite verwendet. Die restlichen 63K einer erweiterten 64K-Karte und die Language-Card sind damit frei.
Die RWTS des DOS 3.3 oder Diversi-DOS wird an der üblichen Adresse erwartet und über den Vektor auf Speicherseite 3 angesprungen. Die ersten 5 Bytes des Programms patchen DOS 3.3 bei $B942, damit auch ein Teil der kopiergeschützten Programme und Daten gelesen und editiert werden kann. Wer einen Original-Championship-LODE-RUNNER besitzt, kann dann z.B. die Bilder (Spuren $3 bis $C) lesen.
Da sehr viele ROM-Routinen benutzt werden, ist es möglich, daß das Programm auf dem IIc nicht funktioniert. Aus demselben Grund muß auch bei den neuen IIe-ROMs mit eventuellen Problemen gerechnet werden. Speziell die Kassetten-Features funktionieren auf dem IIc in keinem Fall. Die ROM-Routinen wurden alle mit den Labelnamen des Addendums zum Apple-IIe-Reference-Manual versehen. Man kann also dort nachsehen, was die jeweilige ROM-Routine macht.
Da sich die RDKEY-Routine im alten IIe-ROM mit dem Rechtspfeil gewisse Eigenmächtigkeiten erlaubt, haben wir sie durch eine eigene Routine ersetzt, damit wir im Editor den Cursor über die Pfeiltasten steuern können. Hier spielte anfänglich die „Del"-Taste verrückt; wir haben sie daher lahmgelegt, denn benutzt wird diese Taste ohnehin nicht.

```
   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F                  X  Slot
00:01 9D 88 C0 A5 2B 4A 4A 4A 4A 09 C0 85 3F A9 00  ___§%_____§__)_  D  Drive
10:85 3E D8 20 84 FE 20 2F FB 20 89 FE 20 93 FE 20  __X__ß__ä__ß__ß_  T  Spur
20:58 FC A0 00 B9 35 08 F0 06 20 F0 FD C8 D0 F5 20  _ö _9__p__püHPu_  S Sektor
30:0C FD 6C 3E 00 8D 8D D0 C5 C5 CB C5 D2 AD D3 C1  _ü_____PEEKER-SA  R  lesen
40:CD CD C5 CC C4 C9 D3 CB C5 D4 D4 C5 8D 8D C4 C9  MMELDISKETTE__DI  >nxtSekt
50:D3 CB C5 D4 D4 C5 A0 C5 CE D4 C8 C1 C5 CC D4 A0  SKETTE ENTHAELT  <vorSekt
60:CB C5 C9 CE A0 C4 CF D3 8D 8D C4 C1 C8 C5 D2 A0  KEIN DOS__DAHER  ^96vwles
70:D3 D9 D3 D4 C5 CD AD CD C1 D3 D4 C5 D2 AD C4 C9  SYSTEM-MASTER-DI  v96rwles
80:D3 CB C5 D4 D4 C5 8D C5 C9 CE CC C5 C7 C5 CE A0  SKETTE_EINLEGEN   Wschreib
90:D5 CE C4 A0 D2 C5 D4 D5 D2 CE A0 C4 D2 D5 C5 C3  UND RETURN DRUEC  B96schre
A0:CB C5 CE A0 00 C8 D5 C5 D4 C8 C9 C7 A0 D6 C5 D2  KEN _HUETHIG VER  Pdrucken
B0:CC C1 C7 A0 C8 C5 C9 C4 C5 CC C2 C5 D2 C7 A0 B1  LAG HEIDELBERG 1  H96druck
C0:B9 B8 B4 AF B1 B9 B8 B5 00 00 00 00 00 00 00 00  984/1985_____     E Editor
D0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ________________  CCassett
E0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ________________  MMonitor
F0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ________________  Ldisasse
                                                                      ^Clösche
Puffer: 1/0                                                           ^Iinvers
                                                                      ^Qquit
  Cursor durch Pfeile   Ctrl-A Hex <--> ASCII

  Ctrl-I inverse    Ctrl-Q quit
```

Abb. 1. Bildschirminhalt (hier ohne Kopfleiste)

```
Spy - Slot:06  Drive:01  Spur: 00  Sektor:00
   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
   ------------------------------------------------

00:01 9D 88 C0 A5 2B 4A 4A 4A 4A 09 C0 85 3F A9 00  ___§%_____§__)_
10:85 3E D8 20 84 FE 20 2F FB 20 89 FE 20 93 FE 20  __X__ß__ä__ß__ß_
20:58 FC A0 00 B9 35 08 F0 06 20 F0 FD C8 D0 F5 20  _ö _9__p__püHPu_
30:0C FD 6C 3E 00 8D 8D D0 C5 C5 CB C5 D2 AD D3 C1  _ü_____PEEKER-SA
40:CD CD C5 CC C4 C9 D3 CB C5 D4 D4 C5 8D 8D C4 C9  MMELDISKETTE__DI
50:D3 CB C5 D4 D4 C5 A0 C5 CE D4 C8 C1 C5 CC D4 A0  SKETTE ENTHAELT
60:CB C5 C9 CE A0 C4 CF D3 8D 8D C4 C1 C8 C5 D2 A0  KEIN DOS__DAHER
70:D3 D9 D3 D4 C5 CD AD CD C1 D3 D4 C5 D2 AD C4 C9  SYSTEM-MASTER-DI
80:D3 CB C5 D4 D4 C5 8D C5 C9 CE CC C5 C7 C5 CE A0  SKETTE_EINLEGEN
90:D5 CE C4 A0 D2 C5 D4 D5 D2 CE A0 C4 D2 D5 C5 C3  UND RETURN DRUEC
A0:CB C5 CE A0 00 C8 D5 C5 D4 C8 C9 C7 A0 D6 C5 D2  KEN _HUETHIG VER
B0:CC C1 C7 A0 C8 C5 C9 C4 C5 CC C2 C5 D2 C7 A0 B1  LAG HEIDELBERG 1
C0:B9 B8 B4 AF B1 B9 B8 B5 00 00 00 00 00 00 00 00  984/1985_____
D0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ________________
E0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ________________
F0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ________________
```

Abb. 2. Ausdruck des Bildschirminhalts

2. Programmübersicht

Das Programm „Spy the Disk-Monitor" oder kurz DISK.MONITOR ist ein komfortabler DOS-3.3-Sektor-Editor, der aufgrund seiner 80-Zeichendarstellung in der Lage ist, den ganzen Sektor auf einmal sowohl in Hex als auch in ASCII anzuzeigen. Man muß also nicht hin- und herblättern, um sich einen Überblick über einen DOS-Sektor zu verschaffen. Die Sektoren können komfortabel sowohl in Hex als auch in ASCII editiert werden, wobei in ASCII sowohl die Zeichen mit Bit 7 on wie auch mit Bit 7 off direkt per Tastatur eingegeben werden können.
Insgesamt stehen 96 Puffer zur freien Verfügung. Das dürfte wohl genügen! Diese können alle einzeln angesteuert und betrachtet bzw. editiert werden. Man kann sie aber auch komplett auf einmal laden bzw. abspeichern.
Jeder einzelne Puffer (und auch alle zusammen) kann auch ausgedruckt werden. Die dazu verwendeten Steuerzeichen stehen am Ende des Programms und enden mit $00 als Endmarke. Sie werden über STROUT ausgegeben und müssen daher mit $00 enden. Sie sind definiert für den *Epson RX-80 F/T* und bewirken Index-Modus (= 1/3 Zeichenhöhe), einen auf 5 Punkte verringerten Zeilenvorschub und Schmalschrift. Die Steuerzeichen lauten: Esc S1 Esc 3 Ctrl-O Ctrl-O.
Selbstverständlich können alle Befehle auch mit Kleinbuchstaben eingegeben werden. Alle Befehle des Auswahl-Modus sind rechts neben dem ASCII-Fenster auf dem Bildschirm zusehen. Das Befehlszeichen selbst ist invers dargestellt. Die Cursortasten sind durch folgende Zeichen symbolisiert:

↑ = Aufwärtspfeil = 96 Sektor lesen (aufwärts)
v = Abwärtspfeil = 96 Sektor lesen (abwärts)
< = Linkspfeil = vorigen Sektor lesen
> = Rechtspfeil = nächsten Sektor lesen

3. Befehlsübersicht

Drive (D) und Slot (X) der Diskette können frei gewählt werden. Spur (T) und Sektor (S) können einzeln festgelegt werden. Durch jeweils einen Tastendruck kann der vorige Sektor (Linkspfeil) bzw. der folgende Sektor (Rechtspfeil) eingelesen werden. Mit (R) kann man den aktuellen Sektor nochmals lesen, z.B. wenn man sich beim Editieren vertippt hat. Ebenfalls durch einen Tastendruck können gleich alle 96 Puffer in aufsteigender (Hochpfeil) oder absteigender (Tiefpfeil) Reihenfolge gelesen werden, manche Programme lesen ihre Daten nämlich in dekrementierender Reihenfolge (z.B. LODE RUNNER). Ein einzelner Puffer (W) oder alle definierten Puffer (B) können mit einem Tastendruck wieder dorthin auf die Diskette zurückgeschrieben werden, wo sie herkamen. Dies ist möglich, weil zu jedem Puffer in den TSB-Feldern (TSB = Track-Sektor-Buffer) die genaue Herkunft (Spurnummer und Sektornummer) vermerkt ist. Die Spurnummern im Speicher sind von $0B10 bis $0B6F vermerkt, die Sektornummern von $0D10 bis $0D6F. Der Bereich $0C10 bis $0C6F bzw. $0E10 bis $0E6F ist reserviert für eine mögliche Pro-DOS-Erweiterung. Angesichts des Pro-DOS-Disketteneditors EDIT von Arne Schäpers besteht jedoch dazu im Moment kein Bedarf.
Ein Backup auf Kassette ist durch (CW) oder (CS) ebenfalls möglich; dabei kann in einem Bemerkungsfeld ein Kommentar zu dem abgespeicherten Diskettenauszug (jeweils 6 Spuren) mit abgespeichert werden. Dies ist sehr nützlich, da es auf Kassetten keinen Catalog-File gibt. Das Bemerkungsfeld liegt von $0F00 bis $0FFF. Es wird genauso beschriftet wie ein normaler Sektor im Editier-Modus. Mit Ctrl-Q wird diese Arbeit beendet, und das Schreiben auf die Kassette beginnt. Auf der Kassette wird über die ROM-Routine der Speicherbereich von $0B00 bis $6FFF abgespeichert. Gelesen wird dieses Backup von Kassette durch CR („C" und „R"; nicht Return!) oder CL. Eine komplette Diskette benötigt etwa 15 Minuten, so daß bei einer guten Kassette 3 Disketten auf eine Seite passen. Das ist aber eher die Ausnahme, da die meisten Kassetten von Dropouts geradezu übersät sind.
Jeder Puffer kann ausgedruckt werden, einzeln (P) oder alle zusammen (H), dabei steht dann auch wieder das eben erwähnte Bemerkungsfeld zur Verfügung.
Die Puffer können wahlweise in Hex und ASCII (Standard) oder in Hex und Mnemonic (L) betrachtet werden. Bei letzterem kann nach dem „L" jeweils durch eine Zahl von 1 bis F der Anfang des Listings innerhalb dieses Puffers bestimmt werden.
Man kann auch direkt in den Monitor (M) springen, welcher nach Eingabe von „L" den aktuellen Puffer listet. Definiert man im Monitor die TSB-Felder auf Seite $B (Spur) und $D (Sektor), dann kann man mit Ctrl-Y direkt mit dem Einlesen der Sektoren in die 96 Puffer in der eben festgelegten Reihenfolge beginnen. Die Puffer gehen von Seite $10 bis Seite $6F. Puffer 1/0 (von $1000 bis $10FF) ist also in $0B10 (Spur) und $0D10 (Sektor) definiert. Entsprechend steht die Spurnummer von Puffer 1/1 in $0B11 bzw. die Sektornummer in $0D11 usw. Steht nun z.B. in $0B10 ein Wert größer als 34 bzw. in $0D10 ein Wert größer als 15, dann kann der Pufferinhalt nicht von einer Diskette stammen. Er wird daher mit der Backup-Funktion (B) nicht auf die Diskette geschrieben. Außerdem erscheint in der Topzeile an Stelle der Spur- bzw. Sektorangabe die Meldung „Puffer nicht von Disk". Um Fehlbedienungen zu vermeiden, ist im Monitor das DOS lahmgelegt.
Durch einfaches Reset-Drücken kommt man wieder bequem in das Programm zurück. Der Ampersand-Vektor ruft ebenfalls den DISK.MONITOR wieder auf. Im Editor kann man durch Ctrl-I (Tab) zwischen Bit 7 off bzw. Bit 7 on wählen. Auch in der Command-Ebene läßt sich mit Ctrl-I bzw. I die Anzeige der Inverszeichen ein- und ausschalten.

4. Quelltext

Der Vorteil dieses Programms dürfte wohl die Tatsache sein, daß es als Quelltext auf der Peeker-Sammeldiskette vorliegt, denn es kann dann für jeden Verwendungszweck schnell und leicht angepaßt werden. Wegen des Umfangs (über 1500 Zeilen) kann jedoch in diesem Heft selbst nur der Hex-Dump abgedruckt werden.
Der Merlin-Sourcefile nutzt den Speicher bis auf wenige Bytes voll aus. Vor einer Erweiterung muß der Quelltext entweder in mehrere Teile zerhackt oder der Merlin-Pro verwendet werden.

Kurzhinweise

1. Zweck:
Diskette neditor für DOS 3.3
2. Konfiguration:
IIe mit 80-Zeichenkarte; DOS 3.3 oder Diversi-DOS 2C; Epson RX-80
3. Test:
BRUN DISK.MONITOR
4. Sammeldisk:
T.DISK.MONITOR
DISK.MONITOR
5. Sonstiges: Im Listing bei Bedarf Druckersteuerzeichen ändern und Programm neu assemblieren.

SPY.MONITOR

```
BSAVE SPY.MONITOR, A$8000,
L$E9C

$8000: A9 18 8D 42 B9 A9 00 85
$8008: 06 A9 0B 85 07 A2 10 A9
$8010: A0 8D 56 82 20 53 82 E6
$8018: 07 E4 07 D0 F7 A9 00 8D
$8020: 56 82 AD 28 89 85 06 AD
$8028: 29 89 85 07 A9 4C 8D F8
$8030: 03 8D F5 03 A9 51 8D F6
$8038: 03 8D F2 03 A9 80 8D F7
$8040: 03 8D F3 03 20 6F FB A9
$8048: 88 8D FA 03 A9 FD 8D F9
$8050: 03 A9 03 20 95 FE 20 27
$8058: 87 A9 00 8D 7B 05 A9 01
$8060: 20 5B FB 20 28 86 A9 B0
$8068: 20 59 86 20 ED 84 A9 01
$8070: 20 5B FB A9 00 8D 7B 05
$8078: 20 D0 87 C9 91 D0 03 4C
$8080: 27 82 C9 83 D0 13 20 53
$8088: 82 A9 00 8D 7B 05 A9 02
$8090: 20 5B FB 20 ED 84 4C 6E
$8098: 80 C9 8B D0 03 4C C0 86
$80A0: C9 8A D0 03 4C 6D 88 C9
$80A8: 95 D0 03 4C 3F 83 C9 88
$80B0: D0 03 4C 70 83 C9 89 F0
$80B8: 06 09 A0 C9 E9 D0 03 4C
$80C0: B0 8B C9 E8 D0 45 20 BE
$80C8: 87 A9 2F 8D 7B 05 A9 3B
$80D0: A0 8A 20 3A DB 20 D0 87
$80D8: C9 EA F0 06 20 BE 87 4C
$80E0: 6E 80 20 D8 88 A9 0F 85
$80E8: 07 A9 00 8D 7B 05 A9 13
$80F0: 20 5B FB A9 75 A0 8B 20
$80F8: 3A DB A5 07 20 89 86 20
$8100: DA 85 E6 07 C9 70 D0 E1
$8108: 4C 51 80 C9 E3 D0 44 20
$8110: BE 87 A9 2F 8D 7B 05 A9
$8118: 00 85 3C A9 FF 85 3E A9
$8120: 0B 85 3D A9 6F 85 3F 20
$8128: D0 87 C9 F7 F0 04 C9 F3
$8130: D0 09 20 D8 88 20 CD FE
$8138: 4C 4C 81 C9 F2 F0 04 C9
$8140: EC D0 06 20 FD FE 4C 4C
$8148: 81 20 D8 88 A9 10 85 07
$8150: 4C 66 80 C9 E2 D0 4B 20
$8158: BE 87 A9 00 20 5B FB A9
$8160: 2F 8D 7B 05 A9 05 A0 8A
$8168: 20 3A DB 20 D0 87 C9 EA
$8170: F0 06 20 BE 87 4C 6E 80
$8178: A0 10 84 07 A9 00 8D 7B
$8180: 05 A9 13 20 5B FB A9 75
$8188: A0 8B 20 3A DB A5 07 20
$8190: 89 86 20 BE 87 20 89 82
$8198: A4 07 C8 C0 70 D0 DB 4C
$81A0: 6E 80 C9 F2 D0 06 20 5D
$81A8: 82 4C 6E 80 C9 F7 D0 06
$81B0: 20 62 82 4C 6E 80 C9 E4
$81B8: D0 03 4C E5 82 C9 F4 D0
$81C0: 03 4C F3 82 C9 F3 D0 03
$81C8: 4C 1C 83 C9 E5 D0 06 20
$81D0: D5 83 4C 6E 80 C9 ED D0
$81D8: 03 4C 3C 82 C9 F0 D0 06
$81E0: 20 B8 85 4C 51 80 C9 F8
$81E8: D0 03 4C C4 82 C9 EC D0
$81F0: 03 4C 7B 87 C9 B7 B0 16
$81F8: C9 B1 90 12 0A 0A 0A 0A
$8200: 8D 3C 89 20 D0 87 C9 BA
$8208: B0 07 C9 B0 B0 0B 4C 6E
$8210: 80 C9 E1 90 F9 C9 E7 B0
$8218: F5 AC 3C 89 84 07 20 59
$8220: 86 20 ED 84 4C 6E 80 20
$8228: 58 FC A0 89 A9 69 20 3A
$8230: DB A9 00 8D 01 08 8D 02
$8238: 08 4C 84 9D 20 58 FC A0
$8240: 89 A9 69 20 3A DB AD 28
$8248: 89 85 3A AD 29 89 85 3B
$8250: 4C 65 FF A0 00 A9 00 91
$8258: 06 C8 D0 FB 60 A9 01 4C
$8260: 8B 82 20 BE 87 A9 00 20
$8268: 5B FB A9 2F 8D 7B 05 A0
$8270: 8A A9 05 20 3A DB 20 D0
$8278: 87 29 DF C9 D9 F0 0A C9
$8280: CA F0 06 20 BE 87 4C C3
$8288: 82 A9 02 8D 2C 89 AD 25
$8290: 89 C9 10 B0 2E AD 24 89
$8298: C9 23 10 27 A0 20 A9 89
$82A0: 20 D9 03 A9 00 85 48 8D
$82A8: 7B 05 A9 02 20 5B FB 20
$82B0: ED 84 20 92 83 A4 07 AD
$82B8: 24 89 99 00 0B AD 25 89
$82C0: 99 00 0D 60 A9 0B 8D 7B
$82C8: 05 A9 00 20 5B FB 20 D0
$82D0: 87 C9 B8 B0 0A 38 E9 B0
$82D8: 0A 0A 0A 0A 8D 21 89 20
$82E0: 59 85 4C 6E 80 AD 22 89
$82E8: 49 03 8D 22 89 20 59 85
$82F0: 4C 6E 80 A9 00 20 5B FB
$82F8: A9 1E 8D 7B 05 20 D0 87
$8300: 8D 00 02 20 D0 87 8D 01
$8308: 02 A0 00 8C 02 02 20 A7
$8310: FF A5 3E 8D 24 89 20 59
$8318: 85 4C 6E 80 A9 00 20 5B
$8320: FB A9 29 8D 7B 05 20 D0
$8328: 87 8D 00 02 A0 00 8C 01
$8330: 02 20 A7 FF A5 3E 8D 25
$8338: 89 20 59 85 4C 6E 80 AD
$8340: 25 89 C9 0F 90 19 A9 00
$8348: 8D 25 89 AD 24 89 C9 22
$8350: 90 07 A9 00 8D 24 89 F0
$8358: 09 EE 24 89 4C 62 83 EE
$8360: 25 89 20 59 85 A9 8D 20
$8368: ED FD 20 5D 82 4C 6E 80
$8370: AD 25 89 D0 17 A9 0F 8D
$8378: 25 89 AD 24 89 F0 06 CE
$8380: 24 89 4C 62 83 A9 22 8D
$8388: 24 89 D0 D6 CE 25 89 4C
$8390: 62 83 A9 00 20 5B FB A9
$8398: 2F 8D 7B 05 A0 2D 89 C9
$83A0: 10 F0 13 C9 20 F0 16 C9
$83A8: 40 F0 19 C9 80 F0 1C A0
$83B0: 89 A9 AD 4C 3A DB A0 89
$83B8: A9 C3 4C CF 83 A0 89 A9
$83C0: D9 4C CF 83 A0 89 A9 EF
$83C8: 4C CF 83 A0 8A A9 0F 20
$83D0: 3A DB 4C DD FB A0 00 B9
$83D8: 94 8B 8D 7B 05 B9 94 8C
$83E0: 8C 38 89 20 5B FB 20 D0
$83E8: 87 AC 38 89 C9 88 F0 4B
$83F0: C9 95 F0 41 C9 8B F0 37
$83F8: C9 8A F0 2D C9 91 F0 25
$8400: C9 81 F0 22 8C 38 89 8D
$8408: 00 02 20 D0 87 8D 01 02
$8410: A0 00 8C 02 02 20 A7 FF
$8418: A5 3E AC 38 89 A2 80 8E
$8420: 39 89 4C B2 84 60 4C 51
$8428: 84 20 4B 84 4C D7 83 20
$8430: 45 84 4C D7 83 20 43 84
$8438: 4C D7 83 20 41 84 4C D7
$8440: 83 88 60 C8 60 98 38 E9
$8448: 10 A8 60 98 18 69 10 A8
$8450: 60 A9 00 8D 39 89 B9 94
$8458: 8D 8D 7B 05 B9 94 8C 8C
$8460: 38 89 20 5B FB 20 D0 87
$8468: AC 38 89 C9 95 F0 2A C9
$8470: 88 F0 20 C9 8B F0 28 C9
$8478: 8A F0 2A C9 81 F0 2D C9
$8480: 91 F0 28 C9 89 D0 03 4C
$8488: C5 88 2D 3F 89 0D 40 89
$8490: 4C B2 84 20 41 84 4C 51
$8498: 84 20 43 84 4C 51 84 20
$84A0: 45 84 4C 51 84 20 4B 84
$84A8: 4C 51 84 60 4C D7 83 4C
$84B0: 51 84 91 06 B9 94 8B 8D
$84B8: 7B 05 B1 06 20 DA FD B9
$84C0: 94 8D 8D 7B 05 B1 06 09
$84C8: 80 C9 FF F0 0C C9 A0 90
$84D0: 08 B1 06 20 ED FD 4C DE
$84D8: 84 A9 DF 20 ED FD AC 38
$84E0: 89 20 43 84 AD 39 89 C9
$84E8: 80 F0 C1 D0 C2 A9 00 8D
$84F0: 35 89 20 80 FE AD 35 89
$84F8: 20 DA FD 20 84 FE A9 BA
$8500: 20 ED FD A0 00 B1 06 20
$8508: DA FD EE 35 89 A9 A0 20
$8510: ED FD C8 C0 10 90 EE A9
$8518: A0 20 ED FD A0 00 B1 06
$8520: 0D 3E 89 C9 7F F0 0A C9
$8528: FF F0 06 C9 A0 B1 06 B0
$8530: 02 A9 DF 20 ED FD C8 C0
$8538: 10 90 E3 A9 8D 20 ED FD
$8540: 18 A5 06 69 10 85 06 AD
$8548: 35 89 C9 00 D0 A4 AD 28
$8550: 89 85 06 AD 29 89 85 07
$8558: 60 A9 00 20 5B FB A9 00
$8560: 8D 7B 05 A0 89 A9 41 20
$8568: 3A DB AD 21 89 4A 4A 4A
$8570: 4A 20 DA FD A0 89 A9 4D
$8578: 20 3A DB AD 22 89 20 DA
$8580: FD AD 24 89 C9 23 B0 25
$8588: AD 25 89 C9 10 B0 1E A0
$8590: 89 A9 56 20 3A DB AD 24
$8598: 89 20 DA FD A0 89 A9 5F
$85A0: 20 3A DB AD 25 89 20 DA
$85A8: FD 20 BE 87 60 A9 7E A0
$85B0: 8B 20 3A DB 20 BE 87 60
$85B8: 20 BE 87 A9 2F 8D 7B 05
$85C0: A0 8A A9 3B 20 3A DB 20
$85C8: D0 87 29 DF C9 CA F0 0A
$85D0: C9 D9 F0 06 20 BE 87 4C
$85D8: 6E 80 A9 01 20 95 FE A9
$85E0: 94 A0 8E 20 3A DB 20 59
$85E8: 85 A9 8D 20 ED FD 20 28
$85F0: 86 A9 8D 20 ED FD A9 A0
$85F8: 20 ED FD 20 ED FD 20 ED
$8600: FD A9 AD A0 00 20 ED FD
$8608: C8 C0 2F 90 F8 A9 8D 20
$8610: ED FD 20 ED 84 A9 8D 20
$8618: ED FD 20 ED FD A9 00 20
$8620: 95 FE A9 03 20 95 FE 60
$8628: A9 A0 20 ED FD 20 ED FD
$8630: 20 ED FD 20 80 FE A9 00
$8638: 20 DA FD A9 01 8D 38 89
$8640: A9 A0 20 ED FD AD 38 89
$8648: 20 DA FD AD 38 89 18 69
$8650: 01 C9 10 D0 E8 20 84 FE
$8658: 60 8D 38 89 A9 00 8D 7B
$8660: 05 A9 13 20 5B FB A0 8B
$8668: A9 75 20 3A DB AD 38 89
$8670: C9 BA 90 06 38 E9 D7 4C
$8678: 7C 86 29 0F 8D 38 89 A5
$8680: 07 29 F0 18 6D 38 89 85
$8688: 07 8D 29 89 4A 4A 4A 4A
$8690: 20 E3 FD A9 AF 20 ED FD
$8698: AD 29 89 29 0F 20 E3 FD
$86A0: AC 29 89 B9 00 0B 8D 24
$86A8: 89 B9 00 0D 8D 25 89 20
$86B0: 59 85 A9 8D 20 ED FD 20
$86B8: 28 86 A9 8D 20 ED FD 60
$86C0: A0 10 8C 3C 89 AC 3C 89
$86C8: AD 25 89 C9 0F D0 1A A9
$86D0: 00 8D 25 89 AD 24 89 C9
$86D8: 22 F0 06 EE 24 89 4C EC
$86E0: 86 A9 00 8D 24 89 4C EC
$86E8: 86 EE 25 89 AD 24 89 99
$86F0: 00 0B AD 25 89 99 00 0D
$86F8: C8 8C 3C 89 C0 70 D0 C5
$8700: A2 10 86 07 A9 00 8D 7B
$8708: 05 A9 13 20 5B FB A9 75
$8710: A0 8B 20 3A DB A5 07 20
$8718: 89 86 20 5B FB 20 E2 A6
$8720: E0 70 D0 DE 4C 6E 80 A9
$8728: 8D 20 ED FD A0 01 8C 38
$8730: 89 8C 37 89 98 20 5B FB
$8738: A9 46 8D 7B 05 AE 38 89
$8740: EE 38 89 BD 50 8A F0 06
$8748: 20 ED FD 4C 3D 87 EE 37
$8750: 89 AC 37 89 C0 15 D0 DC
$8758: A9 02 8D 7B 05 A9 15 20
$8760: 5B FB A0 8B A9 2B 20 3A
$8768: DB A9 02 8D 7B 05 A9 17
$8770: 20 5B FB A0 8B A9 57 20
$8778: 3A DB 60 A5 06 85 3A A5
$8780: 07 85 3B 20 58 FC A9 0F
$8788: 20 63 FE A9 80 20 ED FD
$8790: 20 D0 87 C9 91 F0 21 C9
$8798: CD F0 20 8D 00 02 A0 00
$87A0: 8C 01 02 20 A7 FF A5 3E
$87A8: 0A 0A 0A 0A 18 65 06 85
$87B0: 3A A5 07 85 3B 4C 86 87
$87B8: 4C 51 80 4C 3C 82 A9 2F
$87C0: 8D 7B 05 A9 00 20 5B FB
$87C8: A0 8A A9 25 20 3A DB 60
$87D0: AC 7B 05 8C 3D 89 4E 3D
$87D8: 89 AC 3D 89 B0 06 AD 55
$87E0: C0 4C E7 87 AD 54 C0 B1
$87E8: 28 8D 3A 89 AD 54 C0 A2
$87F0: 02 AD 00 C0 30 50 CA CA
$87F8: CA D0 F6 AC 7B 05 8C 3D
$8800: 89 4E 3D 89 AC 3D 89 B0
$8808: 06 AD 55 C0 4C 12 88 AD
$8810: 54 C0 AD 3A 89 29 7F 91
$8818: 28 AD 54 C0 A2 02 AD 00
$8820: C0 30 23 CA CA CA D0 F6
$8828: AC 7B 05 8C 3D 89 4E 3D
$8830: 89 AC 3D 89 B0 03 AD 55
$8838: C0 AD 3A 89 09 80 91 28
$8840: AD 54 C0 4C F1 87 8D 3B
$8848: 89 8D 10 C0 AC 7B 05 8C
$8850: 3D 89 4E 3D 89 AC 3D 89
$8858: B0 03 AD 55 C0 AD 3A 89
$8860: 91 28 AD 54 C0 AD 3B 89
$8868: C9 FF F0 B2 60 A0 6F 8C
$8870: 3C 89 AC 3C 89 AD 25 89
$8878: C9 0F D0 1A A9 00 8D 25
$8880: 89 AD 24 89 C9 16 F0 06
$8888: EE 24 89 4C 99 88 A9 00
$8890: 8D 24 89 4C 99 88 EE 25
$8898: 89 AD 24 89 99 00 0B AD
$88A0: 25 89 99 00 0D 88 8C 3C
$88A8: 89 C0 0F D0 C5 4C 00 87
$88B0: AD 3E 89 D0 08 A9 80 8D
$88B8: 3E 89 4C 6E 80 A9 00 8D
$88C0: 3E 89 4C 6E 80 A9 80 4D
$88C8: 3F 89 8D 3F 89 A9 80 4D
$88D0: 40 89 8D 40 89 4C 51 84
$88D8: A9 0F 85 07 A9 00 8D 7B
$88E0: 05 A9 13 20 5B FB A9 75
$88E8: A0 8B 20 3A DB A5 07 20
$88F0: 89 86 20 ED 84 20 D5 83
$88F8: A9 10 85 07 60 A9 03 20
$8900: 95 FE 20 27 87 A9 00 8D
$8908: 7B 05 A9 01 20 5B FB 20
$8910: 28 86 A9 00 20 59 86 20
$8918: ED 84 20 EA 03 4C 00 87
$8920: 01 60 01 00 00 00 31 89
$8928: 00 10 00 00 01 00 00 60
$8930: 01 00 01 EF D8 00 00 00
$8938: 00 00 00 00 00 00 00 FF
$8940: 80 D3 F0 F9 A0 AD A0 D3
$8948: EC EF F4 BA 00 A0 A0 C4
$8950: F2 E9 F6 E5 BA 00 A0 A0
$8958: D3 F0 F5 F2 BA A0 00 A0
$8960: A0 D3 E5 EB F4 EF F2 BA
$8968: 00 DA F5 F2 FD E3 EB A0
$8970: FA F5 A0 8F A0 D3 A0 D0
$8978: A0 D9 A0 A0 F4 E8 E5 A0
$8980: A0 C4 A0 C9 A0 D3 A0 CB
$8988: A0 AD A0 CD A0 CF A0 CE
$8990: A0 C9 A0 D4 A0 CF A0 D2
$8998: A0 8E A0 E4 F5 F2 E3 E8
$89A0: A0 C3 F4 F2 EC AD D2 E5
$89A8: F3 E5 F4 8D 00 AA AA AA
$89B0: A0 E7 F5 F4 E5 F2 A0 D3
$89B8: E5 EB F4 EF F2 A0 A0 AA
$89C0: AA AA 00 AA AA AA A0 D3
$89C8: E3 E8 F2 E5 E9 E2 F3 E3
$89D0: E8 F5 F4 FA A0 AA AA AA
$89D8: 00 AA AA A0 E6 E1 EC F3
$89E0: E3 E8 E5 A0 D6 EF EC AD
$89E8: CE F2 AE A0 AA AA 00 A0
$89F0: E1 EC EC E7 E5 ED E5 E9
$89F8: EE E5 F2 A0 C6 E5 E8 EC
$8A00: E5 F2 A0 A1 00 F3 E9 E3
$8A08: E8 E5 F2 A0 BF A0 00 AA
$8A10: AA AA AA A0 CC E5 F3 E5
$8A18: AD C6 E5 E8 EC E5 F2 A0
$8A20: AA AA AA AA 00 A0 A0 A0
$8A28: A0 A0 A0 A0 A0 A0 A0 A0
$8A30: A0 A0 A0 A0 A0 A0 A0 A0
$8A38: A0 A0 00 C9 F3 F4 A0 E4
$8A40: E5 F2 A0 C4 F2 F5 E3 EB
$8A48: E5 F2 A0 8F E1 EE 8E BF
$8A50: 00 8F D8 8E A0 A0 A0 D3
$8A58: EC EF F4 00 8F C4 8E A0
$8A60: A0 C4 F2 E9 F6 E5 00 8F
$8A68: D4 8E A0 A0 A0 D3 F0 F5
$8A70: F2 00 8F D3 8E A0 D3 E5
$8A78: EB F4 EF F2 00 8F D2 8E
$8A80: A0 A0 EC E5 F3 E5 EE 00
$8A88: 8F BE 8E EE F8 F4 D3 E5
$8A90: EB F4 00 8F BC 8E F6 EF
$8A98: F2 D3 E5 EB F4 00 8F DE
$8AA0: 8E B9 B6 F6 F7 EC E5 F3
$8AA8: 00 8F F6 8E B9 B6 F2 F7
$8AB0: EC E5 F3 00 8F D7 8E F3
$8AB8: E3 E8 F2 E5 E9 E2 00 8F
$8AC0: C2 8E B9 B6 F3 E3 E8 F2
$8AC8: E5 00 8F D0 8E E4 F2 F5
$8AD0: E3 EB E5 EE 00 8F C8 8E
$8AD8: B9 B6 E4 F2 F5 E3 EB 00
$8AE0: 8F C5 8E A0 C5 E4 E9 F4
$8AE8: EF F2 00 8F C3 8E C3 E1
$8AF0: F3 F3 E5 F4 F4 00 8F CD
$8AF8: 8E CD EF EE E9 F4 EF F2
$8B00: 00 8F CC 8E E4 E9 F3 E1
$8B08: F3 F3 E5 00 8F DE C3 8E
$8B10: EC FC F3 E3 E8 E5 00 8F
$8B18: DE C9 8E E9 EE F6 E5 F2
$8B20: F3 00 8F DE D1 8E F1 F5
$8B28: E9 F4 00 C3 F5 F2 F3 EF
```

```
$8B3Ø: F2 AØ E4 F5 F2 E3 E8 AØ
$8B38: DØ E6 E5 E9 EC E5 AØ AØ
$8B4Ø: AØ C3 F4 F2 EC AD C1 AØ
$8B48: C8 E5 F8 AØ BC AD AD BE
$8B5Ø: AØ C1 D3 C3 C9 C9 ØØ C3
$8B58: F4 F2 EC AD C9 AØ E9 EE
$8B6Ø: F6 E5 F2 F3 E5 AØ AØ AØ
$8B68: AØ C3 F4 F2 EC AD D1 AØ
$8B7Ø: F1 F5 E9 F4 ØØ DØ F5 E6
$8B78: E6 E5 F2 BA AØ ØØ DØ F5
$8B8Ø: E6 E6 E5 F2 AØ EE E9 E3
$8B88: E8 F4 AØ F6 EF EE AØ C4
$8B9Ø: C9 D3 CB ØØ Ø3 Ø6 Ø9 ØC
$8B98: ØF 12 15 18 1B 1E 21 24
$8BAØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8BA8: ØF 12 15 18 1B 1E 21 24
$8BBØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8BB8: ØF 12 15 18 1B 1E 21 24
$8BCØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8BC8: ØF 12 15 18 1B 1E 21 24
$8BDØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8BD8: ØF 12 15 18 1B 1E 21 24
$8BEØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8BE8: ØF 12 15 18 1B 1E 21 24
$8BFØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8BF8: ØF 12 15 18 1B 1E 21 24
$8CØØ: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C08: ØF 12 15 18 1B 1E 21 24
$8C1Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C18: ØF 12 15 18 1B 1E 21 24
$8C2Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C28: ØF 12 15 18 1B 1E 21 24
$8C3Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C38: ØF 12 15 18 1B 1E 21 24
$8C4Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C48: ØF 12 15 18 1B 1E 21 24
$8C5Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C58: ØF 12 15 18 1B 1E 21 24
$8C6Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C68: ØF 12 15 18 1B 1E 21 24
$8C7Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C78: ØF 12 15 18 1B 1E 21 24
$8C8Ø: 27 2A 2D 3Ø Ø3 Ø6 Ø9 ØC
$8C88: ØF 12 15 18 1B 1E 21 24
$8C9Ø: 27 2A 2D 3Ø Ø2 Ø2 Ø2 Ø2
$8C98: Ø2 Ø2 Ø2 Ø2 Ø2 Ø2 Ø2 Ø2
$8CAØ: Ø2 Ø2 Ø2 Ø2 Ø3 Ø3 Ø3 Ø3
$8CA8: Ø3 Ø3 Ø3 Ø3 Ø3 Ø3 Ø3 Ø3
$8CBØ: Ø3 Ø3 Ø3 Ø3 Ø4 Ø4 Ø4 Ø4
$8CB8: Ø4 Ø4 Ø4 Ø4 Ø4 Ø4 Ø4 Ø4
$8CCØ: Ø4 Ø4 Ø4 Ø4 Ø5 Ø5 Ø5 Ø5
$8CC8: Ø5 Ø5 Ø5 Ø5 Ø5 Ø5 Ø5 Ø5
$8CDØ: Ø5 Ø5 Ø5 Ø5 Ø6 Ø6 Ø6 Ø6
$8CD8: Ø6 Ø6 Ø6 Ø6 Ø6 Ø6 Ø6 Ø6
$8CEØ: Ø6 Ø6 Ø6 Ø6 Ø7 Ø7 Ø7 Ø7
```

```
$8CE8: Ø7 Ø7 Ø7 Ø7 Ø7 Ø7 Ø7 Ø7
$8CFØ: Ø7 Ø7 Ø7 Ø7 Ø8 Ø8 Ø8 Ø8
$8CF8: Ø8 Ø8 Ø8 Ø8 Ø8 Ø8 Ø8 Ø8
$8DØØ: Ø8 Ø8 Ø8 Ø8 Ø9 Ø9 Ø9 Ø9
$8DØ8: Ø9 Ø9 Ø9 Ø9 Ø9 Ø9 Ø9 Ø9
$8D1Ø: Ø9 Ø9 Ø9 Ø9 ØA ØA ØA ØA
$8D18: ØA ØA ØA ØA ØA ØA ØA ØA
$8D2Ø: ØA ØA ØA ØA ØB ØB ØB ØB
$8D28: ØB ØB ØB ØB ØB ØB ØB ØB
$8D3Ø: ØB ØB ØB ØB ØC ØC ØC ØC
$8D38: ØC ØC ØC ØC ØC ØC ØC ØC
$8D4Ø: ØC ØC ØC ØC ØD ØD ØD ØD
$8D48: ØD ØD ØD ØD ØD ØD ØD ØD
$8D5Ø: ØD ØD ØD ØD ØE ØE ØE ØE
$8D58: ØE ØE ØE ØE ØE ØE ØE ØE
$8D6Ø: ØE ØE ØE ØE ØF ØF ØF ØF
$8D68: ØF ØF ØF ØF ØF ØF ØF ØF
$8D7Ø: ØF ØF ØF ØF 1Ø 1Ø 1Ø 1Ø
$8D78: 1Ø 1Ø 1Ø 1Ø 1Ø 1Ø 1Ø 1Ø
$8D8Ø: 1Ø 1Ø 1Ø 1Ø 11 11 11 11
$8D88: 11 11 11 11 11 11 11 11
$8D9Ø: 11 11 11 11 34 35 36 37
$8D98: 38 39 3A 3B 3C 3D 3E 3F
$8DAØ: 4Ø 41 42 43 34 35 36 37
$8DA8: 38 39 3A 3B 3C 3D 3E 3F
$8DBØ: 4Ø 41 42 43 34 35 36 37
$8DB8: 38 39 3A 3B 3C 3D 3E 3F
$8DCØ: 4Ø 41 42 43 34 35 36 37
$8DC8: 38 39 3A 3B 3C 3D 3E 3F
$8DDØ: 4Ø 41 42 43 34 35 36 37
$8DD8: 38 39 3A 3B 3C 3D 3E 3F
$8DEØ: 4Ø 41 42 43 34 35 36 37
$8DE8: 38 39 3A 3B 3C 3D 3E 3F
$8DFØ: 4Ø 41 42 43 34 35 36 37
$8DF8: 38 39 3A 3B 3C 3D 3E 3F
$8EØØ: 4Ø 41 42 43 34 35 36 37
$8EØ8: 38 39 3A 3B 3C 3D 3E 3F
$8E1Ø: 4Ø 41 42 43 34 35 36 37
$8E18: 38 39 3A 3B 3C 3D 3E 3F
$8E2Ø: 4Ø 41 42 43 34 35 36 37
$8E28: 38 39 3A 3B 3C 3D 3E 3F
$8E3Ø: 4Ø 41 42 43 34 35 36 37
$8E38: 38 39 3A 3B 3C 3D 3E 3F
$8E4Ø: 4Ø 41 42 43 34 35 36 37
$8E48: 38 39 3A 3B 3C 3D 3E 3F
$8E5Ø: 4Ø 41 42 43 34 35 36 37
$8E58: 38 39 3A 3B 3C 3D 3E 3F
$8E6Ø: 4Ø 41 42 43 34 35 36 37
$8E68: 38 39 3A 3B 3C 3D 3E 3F
$8E7Ø: 4Ø 41 42 43 34 35 36 37
$8E78: 38 39 3A 3B 3C 3D 3E 3F
$8E8Ø: 4Ø 41 42 43 34 35 36 37
$8E88: 38 39 3A 3B 3C 3D 3E 3F
$8E9Ø: 4Ø 41 42 43 1B 53 Ø1 1B
$8E98: 33 ØF ØF ØØ ØØ ØØ ØØ ØØ
```


AUGE-Regionalgruppen

Die Apple User Group Europe e.V. mit Sitz in 4200 Oberhausen (Tel. 0208/675141) hat allein in Deutschland etwa 50 Regionalgruppen, die sich einmal oder teilweise auch zweimal im Monat in Schulen oder anderen öffentlichen Gebäuden zwecks Erfahrungsaustausch treffen. Für diejenigen Peeker-Leser, die entsprechende Kontakte suchen und Mitglied in der AUGE werden wollen (Jahresbeitrag DM 100,–), nennen wir nachfolgend PLZ, Ort, Name und Telefonnummer des jeweiligen Regionalleiters.

1000 Berlin, T. Schneider, Tel. 030/8539341
2000 Hamburg, M.Stang, Tel. 040/6400563
2300 Kiel, U. Hannemann, Tel. 0431/803021
2390 Flensburg, D. Schwarz, Tel. 04467/388
2400 Lübeck, R. Oelzen, Tel. 0451/505290
2800 Bremen, H. Klein, Tel. 0421/641643
2850 Bremerhaven, K. Birkner, Tel. 04743/6413
2940 Wilhelmshaven, J. Weikert, Tel. 04421/83147
3000 Hannover, R. Schneider, Tel. 0511/734833
3300 Braunschweig, W. Sühring, Tel. 0531/335987
3392 Clausthal, N. Molitor, Tel. 05323/3037
3400 Göttingen, Dr. U. Hoppe, Tel. 0551/706398
3500 Kassel, A. Stimpel, Tel. 0561/877776
4000 Düsseldorf, U. Julius, Tel. 0211/445454
4100 Duisburg, H. Schröder, Tel. 0203/705827
4200 Oberhausen, B. Schritz, Tel. 02831/4364
4400 Münster, A. Schimek, Tel. 02504/3518
4420 Coesfeld, A. Kock, Tel. 02594/6426
4500 Osnabrück, S. Pfeiffer, Tel. 05473/2777
4600 Dortmund, M. Lucas, Tel. 0231/144920
4630 Bochum, M. Harbold, Tel. 02327/57292
4708 Kamen, C.W. Fürst, Tel. 02307/71105
4800 Bielefeld, H. Baldewein, Tel. 05202/71553
4930 Detmold, H. Scheunemann, Tel. 05231/20665
5000 Köln, H. Rothkegel, Tel. 0221/5994456
5100 Aachen, B. Stolte-Wilke, Tel. 02408/4249
5300 Bonn, U. Berger, Tel. 02244/6556
5400 Koblenz, R. Reckenthäler, Tel. 06771/7459
5600 Wuppertal, B. Schnippering, Tel. 0202/80302
5650 Solingen, O. Holthausen, Tel. 0212/45539
5960 Olpe, A. Hase, Tel. 02762/7870
6000 Frankfurt, C. Bohmann, Tel. 06007/1243
6100 Darmstadt, T. Oppermann, Tel. 06151/27285
6200 Wiesbaden, H. Papenkort, Tel. 06198/2498
6300 Giessen, J.P. Fuchs, Tel. 06441/47705
6450 Hanau, M. Mickan, Tel. 06051/71169
6600 Saarbrücken, I. Dickert, Tel. 06894/52323
6700 Ludwigshafen, Dr. H. Böhm, Tel. 06236/53777
6800 Mannheim, R. Augspurger, Tel. 06229/7814
7000 Stuttgart, Dr. G. Prigge, Tel. 0711/535794
7110 Hohenlohe, U. Schönbohm, Tel. 07944/2488
7400 Tübingen, S. Kleindienst, Tel. 07121/57747
7470 Albstadt, V. Krippinger, Tel. 07431/66915
7750 Konstanz, G. Spohn, Tel. 07542/7469
7800 Freiburg, B. Vollmer, Tel. 0761/74765
7900 Ulm, A.Herold, Tel. 0731/712408
8000 München, E. Grassl, Tel. 089/616318
8360 Deggendorf, L. Daas, Tel. 0911/24743
8480 Weiden, J. Gampert, Tel. 0961/42058/7285
8500 Nürnberg, P. Bieberich, Tel. 0911/221383

Disk-Sucher

Eine Utility zum Durchsuchen von DOS-3.3-Disketten

von Jörg Windheim

Mit diesem kleinen Programm kann man eine ganze Diskette nach einem einzigen String durchsuchen. Der String wird vorwärts, rückwärts, invers (INVERSE) und blinkend (FLASH) gesucht. Der Disk-Sucher ist somit ideal zum Übersetzen oder Ändern von bestehenden Programmen.
Das Programm wird mit
BRUN DISK.SUCHER
gestartet. Zuerst wird nach dem Suchstring gefragt. Vor dem Drücken der Returntaste muß bereits die zu durchsuchende Diskette in das Laufwerk 1 gelegt worden sein. Wenn das Programm nun gestartet wird, zeigt es ständig die Spur und den Sektor an, der gerade gelesen wird. Wird der String gefunden, so werden Spur und Sektor sowie der String, wie er auf dem Sektor gespeichert ist, auf dem Bildschirm wiedergegeben.

Die Begrenzung auf 12 Zeichen lange Strings erscheint mir sinnvoll, da bei längeren Strings die Wahrscheinlichkeit, daß der String auf zwei Files verteilt ist, schon viel zu groß ist. Bei Bedarf kann man den Quelltext ändern.
Beim Suchen eines Strings verfährt man am besten so:

– Programm starten mit BRUN DISK.SUCHER bzw. nachher mit CALL 8192,
– das zu suchende Wort eingeben,
– Diskette einlegen und Return betätigen.
– mit ESC wird der gleiche String nochmals gesucht, und
– mit CR wird das Programm neu gestartet.
Das Programm verlassen Sie mit Ctrl-Reset.

Hat nun der DISK.SUCHER nichts gefunden, so ist der String bestimmt auf zwei Sektoren verteilt. Dazu teilt man nun diesen String in zwei in etwa gleiche Hälften, und sucht nun beide Hälften getrennt für sich. Bleibt auch diese Suche erfolglos, so kann man auf jeden Fall sicher sein, daß dieser String nicht so zusammenhängend auf der Diskette steht.

Kurzhinweise

1. Zweck:
Utility zum Durchsuchen von DOS-3.3-Disketten
2. Konfiguration:
II+/e/c; zu durchsuchende Diskette in Slot 6, Drive 1
3. Test:
BRUN DISK.SUCHER
Dann z.B. Sammeldisk in S6,D1 einlegen und „HUETHIG“ eingeben.
4. Sammeldisk:
T.DISK.SUCHER
DISKSUCHER
5. Sonstiges:
Das Programm bricht bei Lesefehlern nicht ab, z.B. wenn gar keine Diskette im Laufwerk liegt. In diesem Fall ziehen Sie alle Disketten heraus und drücken Sie Ctrl-Reset.

DISK.SUCHER

```
BSAVE DISK.SUCHER, A$2000, L$01D1

$2000: 4C 18 20 01 60 01 00 00
$2008: 00 14 20 00 30 00 00 01
$2010: 00 00 60 01 00 01 EF D8
$2018: 20 58 FC A9 0C 20 5B FB
$2020: A2 00 BD 99 21 20 ED FD
$2028: E8 E0 18 D0 F5 A9 0C 85
$2030: 24 20 6F FD E0 0D 10 E0
$2038: E0 00 F0 21 86 06 20 62
$2040: FC 20 60 20 A2 00 86 24
$2048: A9 A0 20 ED FD E8 E0 14
$2050: D0 F6 20 18 FD C9 9B F0
$2058: E5 C9 8D F0 BB 4C D3 03
$2060: A9 22 8D 07 20 A9 0F 8D
$2068: 08 20 A9 B2 8D B7 21 8D
$2070: B8 21 A9 C6 8D C4 21 4C
$2078: BD 20 CE C4 21 AD C4 21
$2080: C9 C0 D0 05 A2 B9 8E C4
$2088: 21 C9 AF D0 20 A9 C6 8D
$2090: C4 21 CE B8 21 AD B8 21
$2098: C9 C0 D0 05 A2 B9 8E B8
$20A0: 21 C9 AF D0 08 A9 C6 8D
$20A8: B8 21 CE B7 21 CE 08 20
$20B0: 10 0B A9 0F 8D 08 20 CE
$20B8: 07 20 10 01 60 A2 00 86
$20C0: 24 BD B1 21 20 ED FD E8
$20C8: E0 14 D0 F5 A9 20 A0 03
$20D0: 20 D9 03 20 D9 20 4C 7A
$20D8: 20 A9 00 38 E5 06 85 07
$20E0: A2 00 BD 00 02 9D C5 21
$20E8: E8 E4 06 D0 F5 20 4D 21
$20F0: 20 1D 21 20 4D 21 20 3C
$20F8: 21 20 4D 21 A2 00 A4 06
$2100: 88 BD 00 02 99 C5 21 E8
$2108: 88 E4 06 D0 F4 20 4D 21
$2110: 20 1D 21 20 4D 21 20 3C
$2118: 21 20 4D 21 60 A2 00 BD
$2120: C5 21 38 E9 80 C9 40 30
$2128: 0A 38 E9 40 C9 20 30 03
$2130: 38 E9 20 9D C5 21 E8 E4
$2138: 06 D0 E4 60 A2 00 BD C5
$2140: 21 18 69 40 9D C5 21 E8
$2148: E4 06 D0 F2 60 AD 0B 20
$2150: 85 19 AD 0C 20 85 1A A0
$2158: 00 B9 C5 21 D1 19 D0 0B
$2160: C8 C4 06 D0 F4 20 87 21
$2168: 20 74 21 E6 19 A5 19 C5
$2170: 07 D0 E4 60 20 62 FC A2
$2178: 00 86 24 BD B1 21 20 ED
$2180: FD E8 E0 14 D0 F5 60 A9
$2188: 19 85 24 A2 00 BD C5 21
$2190: 20 ED FD E8 E4 06 D0 F5
$2198: 60 D3 F5 E3 E8 F3 F4 F2
$21A0: E9 EE E7 BA A0 AE AE AE
$21A8: AE AE AE AE AE AE AE AE
$21B0: AE D3 F0 F5 F2 BA A0 B2
$21B8: B2 A0 A0 A0 D3 E5 EB F4
$21C0: EF F2 BA A0 C6 00 00 00
$21C8: 00 00 00 00 00 00 00 00
$21D0: 00 00 00 00 00 00 00 00
```

ATARI ST

ASSEMBLER-PRAXIS AUF ATARI ST

Roland Löhr
...ein Altmeister der Assembleranwendung, Herausgeber des Mikrocomputer-Magazins MICRO MAG, veröffentlicht bei te-wi seine souveräne Darstellung der Assemblerprogrammierung auf ATARI STs.

Erklärt Grundlagen:

Begriffe und Werkzeuge der Assemblerprogrammierung...erforderliche Systemkenntnisse...systembezogene Erläuterung der 68000er Befehlsfunktionen.

Zeigt Anwendungen:

Hantieren mit Assemblern: Aufruf von Assemblern; Steuern ihrer Optionen über Direktiven; Stellungnahme zu realen ATARI-ST-Assemblern.
Arbeiten in der ATARI-ST-Programmierumgebung: Textprogramme zur Programmentwicklung; ein Editor; ein Parser; das Betriebssystem; BIOS-Funktionen; BIOS-Toolbox; GEMDOS Toolkit; das erweiterte XBIOS.
Anwenden des Befehlssatzes in Musterprogrammen für: E/A-Routinen, Rekursionen, dez/bin Rechenarten, Stackverwaltung, Adressverwaltung, Entscheidungen, Schleifenkonstrukte, Unterprogramme, numerierte Traps, Bedienen von Interfacebausteinen, Texterkennung, Textverarbeitung, Tastaturdekodierung, memory dumps, Floppy-Tests/Funktionen, serielle RS232-Datenübertragung usw.

Entwickelt Hilfsprogramme:

BIOS-Toolbox; GEMDOS-Toolkits; ein Editor; ein Parser; Arbeiten mit Toolkits.
Die Programme des Buchs sind auf Diskette vom Autor erhältlich.

Ein Fachtext in klarer Sprache mit leserfreundlichem Druckbild, guter Bilddokumentation und umfangreichen Listings von Musterprogrammen (auf Diskette beim Autor erhältlich).

ca. 300 Seiten, Softcover, DM 59,–

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

Weitere te-wi-Bücher

NEU

DAS „C"-BUCH
(Herold /Unger)
Ein „C"-Kurs der Industrie. Für sämtliche C-Konstrukte. Über 100 Beispiele. Anspruchsvoll in Text/Bildmaterial, ca. 500 Seiten, Softcover, DM 79,–

UNIX
(Yates/Thomas) US-Standardwerk der UNIX-Promoterin Yates. Eine sachkundige Übersicht und Einführung in die Anwendung, 550 Seiten, Softcover, DM 79,–

LOGO –
Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den USA.
Best-rezensiert von Pädagogen und deutschen Kultusministerien. Ein bildreicher Führer durch u. a. ATARI's LOGO. Von Papert's Schüler D. Watt.
384 Seiten, A4, DM 59,–

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,–

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,–

Mein ATARI Computer
Best-rezensiertes Standardwerk deutscher ATARI-User-Groups. Kompakte ATARI 400-/800-System/Peripheriebeschreibung.
Von Poole/McNiff/Cook, 500 Seiten, Softcover, DM 59,–

Sprühende Ideen mit ATARI-GRAPHIK
Fröhlicher Lehrstoff in Geometrie und Farbenlehre eines amerikanischen Lehrers mit ATARI Graphikmöglichkeiten.
Von Tom Rowley, 224 Seiten, Softcover, DM 49,–

6502 – Programmieren in Assembler
Dieses Buch behandelt ausführlich die Assemblersprachen-Programmierung für den weitverbreiteten Mikroprozessor 6502.
Lance Leventhal, 704 Seiten, Softcover, DM 59,–

fm 4645

Noch im Programm: Einführung in die Mikrocomputer-Technik, DM 66,–
Computer für Kinder, ATARI, DM 29,80

String-Catalog

DOS-3.3-Dateinamen im String-Array

von Waldemar Kupisch

Will man ein Applesoft-Programm, das mit vielen Dateien arbeiten muß, halbwegs komfortabel gestalten, so steht man nicht selten vor dem Problem, die Namen der einzelnen DOS-3.3-Files zur weiteren Bearbeitung in Strings einzulesen. (Unter ProDOS ist dies im Gegensatz zu DOS 3.3 denkbar einfach. Vgl. „CAT.ARRAY“ in Peeker, 4/85, S. 44.) Bleibt man beim Applesoft, dann kommt man nicht umhin, mit vielen Peeks und Pokes und RWTS-Aufrufen und noch viel mehr String-Operationen die Filenamen mühsam aus den Catalog-Sektoren auszulesen. Wer das einmal ausprobiert hat, kann verstehen, warum ich versucht habe, dieses Problem auf Assemblerebene zu lösen. Herausgekommen ist dabei ein kurzes und dennoch effizientes, vollkommen relozierbares Maschinenprogramm, das folgende Kriterien erfüllt:

1. Es ist relozierbar und kann somit an jedem beliebigen Speicherplatz abgelegt werden. Außer bei Systemaufrufen werden keine absoluten Befehle verwendet. (Für das Programmlisting STRCAT wurde Origin $8000 gewählt.)
2. Es schreibt die Filenamen in ein vorher vom Applesoft-Programm dimensioniertes Stringfeld. Um Verträglichkeit mit dem Applesoft-Programm zu sichern, werden Interpreter-Routinen für die Stringpool-Verwaltung benutzt. Die vom Maschinenprogramm erzeugten Strings werden damit vollständig in die Applesoft-Umgebung eingebunden.
3. Das Programm kann mit einem CALL-Befehl aufgerufen werden. Damit werden die anderen Möglichkeiten wie USR und Ampersand (&) für den Programmierer nicht eingeschränkt.

STRCAT.DEMO1

```
10Ø ANFANG = 32768: REM $8ØØØ
11Ø HIMEM: ANFANG
12Ø PRINT CHR$ (4);"BLOAD STRCAT,A";ANFANG
13Ø DIM CA$(1Ø5): REM FELD FUER DIE FILENAMEN
14Ø A1 = PEEK (ANFANG + 17Ø):A2 = PEEK (ANFANG + 171): REM ALTE WERTE MERKEN
15Ø POKE ANFANG + 17Ø,234: POKE ANFANG + 171,234: REM BRANCHBEFEHL
    WIRD DURCH NOPS UEBERSCHRIEBEN. DAMIT KOENNEN ALLE FILES EINGELESEN WERDEN
16Ø CALL ANFANG
17Ø POKE ANFANG + 17Ø,A1: POKE ANFANG + 171,A2
18Ø IF PEEK (8) = Ø THEN 8Ø: REM WENN ANZAHL FILES =Ø DANN KEINE ZUWEISUNG
19Ø FOR I = Ø TO PEEK (8) - 1: PRINT CA$(I): NEXT : REM FILENAMEN LISTEN
```

STRCAT.DEMO2

```
10Ø ANFANG = 32768: REM $8ØØØ
11Ø HIMEM: ANFANG
12Ø PRINT CHR$ (4);"BLOAD STRCAT,A";ANFANG
13Ø DIM CA$(1Ø5): REM FELD FUER DIE FILENAMEN
14Ø DIM C$(1Ø5,2): REM FELD FUER NACH FILETYPE SORTIERTE FILENAMEN
15Ø DIM L(2): REM ANZAHL DER EINZELNEN FILETYPEN
16Ø POKE ANFANG + 169,ØØ: REM TEXTFILES
17Ø CALL ANFANG: REM MASCHINENPROGRAMM AUFRUFEN
18Ø L(Ø) = PEEK (8)
19Ø IF L(Ø) = Ø THEN 23Ø: REM UM DIE ERSTE ZUWEISUNG IN DER FOLGENDEN
    FOR-NEXT SCHLEIFE ZU VERMEIDEN
2ØØ FOR I = Ø TO L(Ø) - 1
21Ø C$(I,Ø) = CA$(I): REM TEXTFILES EINTRAGEN
22Ø NEXT
23Ø POKE ANFANG + 169,2: REM APPLESOFT FILES
24Ø CALL ANFANG
25Ø L(1) = PEEK (8)
26Ø IF L(1) = Ø THEN 3ØØ
27Ø FOR I = Ø TO L(1) - 1
28Ø C$(I,1) = CA$(I): REM APPLESOFT FILES EINTRAGEN
29Ø NEXT
3ØØ POKE ANFANG + 169,4: REM BINARY FILES
31Ø CALL ANFANG
32Ø L(2) = PEEK (8)
33Ø IF L(2) = Ø THEN 37Ø
34Ø FOR I = Ø TO L(2) - 1
35Ø C$(I,2) = CA$(I): REM BINARY FILES EINTRAGEN
36Ø NEXT
37Ø HOME
38Ø HTAB 1Ø: VTAB 8: PRINT "( 1 )  TEXTFILES"
39Ø HTAB 1Ø: VTAB 1Ø: PRINT "( 2 )  APPLESOFT FILES"
4ØØ HTAB 1Ø: VTAB 12: PRINT "( 3 )  BINARY FILES"
41Ø HTAB 1Ø: VTAB 14: PRINT "( 4 )  ENDE"
42Ø HTAB 1Ø: VTAB 17: PRINT "WELCHEN TYP ";
43Ø GET A$
44Ø IF A$ < "1" OR A$ > "4" THEN 43Ø: REM EINGABEKONTROLLE
45Ø A = VAL (A$)
46Ø HOME
47Ø IF A = 4 THEN 54Ø
48Ø IF L(A - 1) = Ø THEN 52Ø
49Ø FOR I = Ø TO L(A - 1)
5ØØ PRINT C$(I,A - 1)
51Ø NEXT : REM FILENAMEN LISTEN
52Ø VTAB 24: PRINT "BITTE TASTE DRUECKEN ";: GET A$
53Ø GOTO 37Ø
54Ø END
```

1. Variablenverwaltung

Um das Programm zu verstehen, muß man wissen, wie der Applesoft-Interpreter die Variablen anlegt. Grundsätzlich werden alle Variablen in dem Speicherbereich abgelegt, der durch die Zeropage-Pointer $0069-$006A (LOMEM) und $0073-$0074 (HIMEM) begrenzt wird. Als erstes werden die einfachen Variablen von LOMEM ab aufwärts angelegt. Darauf folgen die Feldvariablen, deren Bereich durch die Pointer $006B-$006C (Anfang Array-Space) und $006D-$006E (Ende Array-Space + 1) bestimmt ist. Strings werden von HIMEM abwärts bis zum Ende des Array-Speichers angelegt. Wird ein String neu definiert, so weist eine Interpreter-Routine diesem String einen neuen Platz zu und reduziert gleichzeitig den Pointer $006F-$0070 (Anfang String-Space). Der alte Stringinhalt bleibt also im Speicher stehen. So werden mit der Zeit immer mehr String-Leichen geschaffen. Wenn der Pointer $006F-$0070 das Ende des Array-Speichers erreicht, dann wird der Stringspeicher aufgeräumt. Das übernimmt die sog. Garbage-Collection-Routine im Interpreter. Bei einer großen Anzahl von Strings braucht diese Routine mehrere Minuten. (Vgl. aber hierzu die schnelle GC-Routine in Peeker, Heft 1/85.)

In früheren Peeker-Beiträgen wurde schon auf die Speicherbelegung der Variablen eingegangen. Hier nun nochmals eine Zusammenfassung:

Einfache Variablen belegen immer 7 Bytes. Die ersten beiden Bytes enthalten den Variablennamen. Die Bit 7 der Namensbytes stellen in codierter Form den Variablentyp dar. Bei Real-Variablen sind beide 0, bei Integer-Variablen sind beide 1. Stringnamen haben im ersten Byte, Bit 7 = 0 und im zweiten Byte, Bit 7 = 1. Die restlichen Bytes enthalten den Variablen-Wert:
Real: 1 Byte Exponent und 4 Bytes für Mantisse.
Integer: Low-Byte und High-Byte für Zahl und 3 Bytes = 0 (unbenutzt).
String: 3 Bytes Stringdeskriptor (Stringlänge, Pointer auf String) und 2 Bytes = 0 (unbenutzt).
Feld-Variablen sind wie folgt aufgebaut:
Variablenname wie oben.
Länge der gesamten Feld-Variablen mit Low-Byte, High-Byte.
Anzahl der Dimensionen in einem Byte, also maximal 255.
Größe des letzen Index mit High-Byte, Low-Byte!
... usw. bis
Größe des ersten Index mit High-Byte, Low-Byte.

STRCAT

BSAVE STRCAT, A$8000, L286

```
                          1           ORG  $8000
                          2  *
                          3  * String-Catalog
                          4  * ==============
                          5  *
                          6  * von Waldemar Kupisch, 1985
                          7  *
                          8  * Dieses relokative Programm liest Filenamen in das
                          9  * zuvor von einem BASIC-Programm dimensionierte Stringfeld
                         10  * CA$(105) ein.
                         11  * Filetype-Auswahl ist durch POKE ANFANG+169,TYPE möglich.
                         12  * TYPE = (0,1,2,4,8,16) für (T,I,A,B,S,R) TYPE FILES.
                         13  * T Type Files sind voreingestellt. (s. Zeile 147)
                         14  * Mit  POKE ANFANG+170,234 : POKE ANFANG+171,234 :REM NOP
                         15  * können alle Files eingelesen werden.
                         16  * Die Anzahl der eingelesenen Files steht in $08.
                         17  * Das Programm benötigt Applesoft im ROM und 48K-DOS 3.3
                         18  *
                         19  COUT     EQU  $FDED      ;PRINT
                         20  BASIC    EQU  $E003      ;BASIC-Warmstart
                         21  PRERR    EQU  $FF2D      ;Print "ERR"
                         22  READDIR  EQU  $B011      ;48K-DOS: Catalogsektor lesen mit
                         23  *  CLC : 1. Catsektor
                         24  *  SEC : nächster Catsektor
                         25  GETIOB   EQU  $3E3       ;IO-Block-Adresse holen
                         26  BUFFER   EQU  $FA        ;Pointer: 256-Byte-Sektorpuffer
                         27  DCT      EQU  $FC        ;Pointer auf DCT
                         28  COUNT    EQU  $08        ;Filezähler
                         29  FCOUNT   EQU  $09        ;Zähler
                         30  DIROFF   EQU  $0B        ;Offset zum ersten Catalogeintrag
                         31  FILEOFF  EQU  $23        ;Offset zum folg. Catalogeintrag
                         32  GETSP    EQU  $E452      ;Platz für neuen String schaffen
                         33  *Input: Accu=Strlänge
                         34  *Output: A,X,Y = neuer Deskriptor
                         35  BEGINAR  EQU  $6B        ;Pointer Anfang Array-Space
                         36  ENDAR    EQU  $6D        ;Pointer Ende Array-Space+1
                         37  BEGINST  EQU  $71        ;Pointer auf Stringposition
                         38  ARHEAD   EQU  $94        ;Pointer Feldvar. Kopf
                         39  LENGTH   EQU  $8F        ;Länge der einz. Feldvar
                         40  DESCR    EQU  $5E        ;Feldvar Deskriptor-Pointer
                         41  *
                         42  * Feldvar CA$() suchen und DESCR bestimmen
                         43  *
8000: A5 6B              44           LDA  BEGINAR    ;Feldanfang vorbereiten
8002: A6 6C              45           LDX  BEGINAR+1
8004: 85 94              46           STA  ARHEAD     ;Pointer auf Feldvar Kopf
8006: 86 95              47           STX  ARHEAD+1   ;auf Feldanfang setzen
8008: A9 03              48           LDA  #$03       ;Länge der Feldelemente
                         49  *= 3 für String
800A: 85 8F              50           STA  LENGTH     ;merken
800C: A5 94              51  CA1      LDA  ARHEAD
800E: A6 95              52           LDX  ARHEAD+1
8010: E4 6E              53           CPX  ENDAR+1    ;Feldende erreicht?
8012: D0 06              54           BNE  CA2        ;nein: -> CA2
8014: C5 6D              55           CMP  ENDAR      ;LByte vergleichen
8016: D0 02              56           BNE  CA2
8018: F0 3E              57           BEQ  ERROR      ;Feldende erreicht -> ERROR
801A: 85 5E              58  CA2      STA  DESCR      ;Deskriptor-Pointer auf
801C: 86 5F              59           STX  DESCR+1    ;Feldvar Kopf setzen
801E: A0 00              60           LDY  #$00
8020: B1 5E              61           LDA  (DESCR),Y  ;1. Namensbyte vergl.
8022: C9 43              62           CMP  #'C'       ;mit 'C' MSB = 0
8024: D0 09              63           BNE  CA3        ;nicht gleich -> nächste Feldvar
8026: C8                 64           INY
8027: B1 5E              65           LDA  (DESCR),Y  ;2. Namensbyte vergl.
8029: C9 C1              66           CMP  #"A"       ;mit "A" MSB = 1
802B: F0 15              67           BEQ  CA4        ;CA$ gefunden -> CA4
802D: D0 01              68           BNE  CA5        ;nächste Feldvar
                         69  *
802F: C8                 70  CA3      INY
8030: C8                 71  CA5      INY             ;ARHEAD auf nächsten Feldvar Kopf
8031: 18                 72           CLC
8032: B1 5E              73           LDA  (DESCR),Y  ;Länge der Feldvar
8034: 65 94              74           ADC  ARHEAD     ;zu Pointer addieren
8036: 85 94              75           STA  ARHEAD
8038: C8                 76           INY
8039: B1 5E              77           LDA  (DESCR),Y  ;High-Bytes addieren
803B: 65 95              78           ADC  ARHEAD+1
803D: 85 95              79           STA  ARHEAD+1
803F: 18                 80           CLC
8040: 90 CA              81           BCC  CA1        ;unbedingter Sprung
                         82  *
```

Elemente der Reihe nach (0,0,..), (1..n,0,..), (1..n,1..n,..),..
Die Länge der Elemente beträgt 5, 2, 3 bei Real-, Integer- und String-Variablen.
Einfache Integer-Variablen sparen also keinen Speicherplatz, sondern nur Integer-Feld-Variablen!

2. System-Routinen

Bevor wir zur Programmbeschreibung übergehen, müssen wir noch einige DOS- und Interpreter-Routinen besprechen. Ausführlichere Informationen enthalten (1) und (2).
DOS-Routine **READDIR** ($B011): Im DOS 3.3 gibt es ein Unterprogramm zum Einlesen der Catalog-Sektoren. Springt man in dieses Unterprogramm mit gelöschtem Carry (CLC), dann holt sich das Programm die Track- und Sektornummer des ersten Catalog-Sektors aus der VTOC (Volume Table of Contents). Geschieht der Einsprung mit gesetztem Carry (SEC), dann holt sich das Programm die Track- und Sektornummer des nächsten Catalog-Sektors aus dem vorher eingelesenen Catalog-Sektor und liest diesen in den DOS-Puffer. Vorher wird die Pufferadresse in die RWTS-Parameterliste eingetragen. Das Programm bricht mit Errorcode (End of Directory) ab, wenn die Tracknummer des nächsten Sektors 0 ist.
DOS-Routine **GETIOB** ($03E3): Mit einem JSR GETIOB erhält man im Y-Register und im Akkumulator die Adresse des I/O-Control-Blocks (IOB). Damit erfährt man die Adresse des 256 Byte langen Sektorpuffers und kann auf den Inhalt des eingelesenen Sektors zugreifen.
Interpreter-Routine **GETSP** ($E452): Dieses Unterprogramm schafft Platz für einen neuen String. Beim Einsprung muß im Akkumulator die Stringlänge stehen. Dabei wird geprüft, ob noch genügend Platz für den String im Stringpool vorhanden ist, und gegebenenfalls die Garbage-Collection aufgerufen. Ist danach nicht genügend Platz vorhanden, wird die Programmausführung mit „Out of Memory“ abgebrochen. Andernfalls kehrt das Unterprogramm mit dem neuen Stringdeskriptor in Akkumulator, X- und Y-Register (A: Länge, X: Low-Byte, Y: High-Byte) zum aufrufenden Programm zurück.

3. Programmablauf

Schaut man das Programm STRCAT zum ersten Male an, so drängt sich der Eindruck auf, als wenn hier ziellos in der Gegend „herumverzweigt“ würde. Die Ursache dafür liegt in der begrenzten Verzweigungsfähigkeit der relativen Branchbefehle des 6502 (-126 bis +129 Bytes um den Branch herum). Bei relokativer =

```
8042: C8          83  CA4     INY             ;Pointer auf Deskr. von CA$(Ø)
8043: C8          84          INY
8044: C8          85          INY
8045: B1 5E       86          LDA  (DESCR),Y  ;Anz Dimensionen Lesen
8047: C9 Ø1       87          CMP  #$Ø1       ;Vergl. mit 1
8049: DØ ØD       88          BNE  ERROR      ;-> ERROR falls <> 1
804B: ØA          89          ASL             ;*2
804C: 69 Ø5       9Ø          ADC  #$Ø5       ;+5 (Carry = Ø)
804E: 65 5E       91          ADC  DESCR      ;= Pointer auf Deskrip.von CA$(Ø)
8050: 85 5E       92          STA  DESCR
8052: 9Ø 21       93          BCC  DIR
8054: E6 5F       94          INC  DESCR+1    ;High-Byte inkrementieren
8056: DØ 1D       95          BNE  DIR        ;weiter bei DIR
                  96  *
                  97  ERROR                   ;Fehlerbehandlung
8058: A9 C3       98          LDA  #"C"       ;Print "CA$-ERR"
805A: 2Ø ED FD    99          JSR  COUT
805D: A9 C1       1ØØ         LDA  #"A"
805F: 2Ø ED FD    1Ø1         JSR  COUT
8062: A9 A4       1Ø2         LDA  #"$"
8064: 2Ø ED FD    1Ø3         JSR  COUT
8067: A9 AD       1Ø4         LDA  #"-"
8069: 2Ø ED FD    1Ø5         JSR  COUT
806C: 2Ø 2D FF    1Ø6         JSR  PRERR
806F: 68          1Ø7         PLA             ;Pop Returnadresse
8Ø7Ø: 68          1Ø8         PLA
8Ø71: 4C Ø3 EØ    1Ø9         JMP  BASIC      ;BASIC Warmstart
                  11Ø *
8074: 6Ø          111 ENDE    RTS
                  112 *
                  113 * Catalog auslesen und Filenamen übertragen
                  114 *
8Ø75: A9 ØØ       115 DIR     LDA  #$ØØ
8Ø77: 85 Ø8       116         STA  COUNT      ;Zähler initialisieren
8Ø79: 2Ø E3 Ø3    117         JSR  GETIOB     ;IO-BLock-Adresse holen
8Ø7C: 84 FC       118         STY  DCT
8Ø7E: 85 FD       119         STA  DCT+1      ;und ablegen
8Ø8Ø: 18          12Ø         CLC             ;vorbereiten zum 1. Catalogsektor
8Ø81: A9 ØØ       121 DIR1    LDA  #$ØØ
8Ø83: 85 Ø9       122         STA  FCOUNT     ;Filecount initialisieren
8Ø85: 2Ø 11 BØ    123         JSR  READDIR    ;Catalogsektor lesen
8Ø88: 18          124         CLC
8Ø89: AØ Ø8       125         LDY  #$Ø8       ;Pufferadresse holen
8Ø8B: B1 FC       126         LDA  (DCT),Y
8Ø8D: 69 ØB       127         ADC  #DIROFF    ;und Offset addieren
8Ø8F: 85 FA       128         STA  BUFFER
8Ø91: C8          129         INY
8Ø92: B1 FC       13Ø         LDA  (DCT),Y
8Ø94: 69 ØØ       131         ADC  #$ØØ       ;High-Byte addieren
8Ø96: 85 FB       132         STA  BUFFER+1
                  133 *
8Ø98: AØ ØØ       134 FILE    LDY  #$ØØ
8Ø9A: B1 FA       135         LDA  (BUFFER),Y ;Check, ob gültiger Eintrag
8Ø9C: FØ D6       136         BEQ  ENDE       ;wenn Ø, dann kein weiterer
                  137 *Eintrag -> ENDE
8Ø9E: C9 FF       138         CMP  #$FF       ;Deleted?
8ØAØ: FØ 18       139         BEQ  WEITER     ;Ja -> nächster Eintrag
8ØA2: C8          14Ø         INY
8ØA3: C8          141         INY
8ØA4: B1 FA       142         LDA  (BUFFER),Y ;File-Type Byte
8ØA6: 29 7F       143         AND  #$7F       ;Clear MSB (Lock Bit)
8ØA8: C9 ØØ       144         CMP  #$ØØ       ;Check, ob Textfile
                  145 *Hier kann Type geändert werden
8ØAA: DØ ØE       146         BNE  WEITER     ;wenn nicht dann weiter
                  147 *Hier NOPs, falls alle Files
8ØAC: AØ 2Ø       148         LDY  #$2Ø       ;Y auf letztes Namensbyte
8ØAE: B1 FA       149 GO      LDA  (BUFFER),Y
8ØBØ: C9 AØ       15Ø         CMP  #$AØ       ;Space?
8ØB2: DØ 1C       151         BNE  LESEN      ;kein Space dann einlesen
8ØB4: 88          152         DEY             ;sonst vorhergehendes Zeich.
8ØB5: DØ F7       153         BNE  GO         ;prüfen
                  154 *
8ØB7: 38          155 NEXTSEC SEC             ;vorber. zum nächsten Catsektor
8ØB8: BØ C7       156         BCS  DIR1
                  157 *
8ØBA: E6 Ø9       158 WEITER  INC  FCOUNT     ;Filecount erhöhen
8ØBC: A5 Ø9       159         LDA  FCOUNT
8ØBE: C9 Ø7       16Ø         CMP  #$Ø7       ;schon 7.ter Eintrag?
8ØCØ: FØ F5       161         BEQ  NEXTSEC    ;dann nächsten Sektor einlesen
8ØC2: 18          162         CLC
8ØC3: A5 FA       163         LDA  BUFFER     ;Fileoffset zu Puffer addieren
8ØC5: 69 23       164         ADC  #FILEOFF
8ØC7: 85 FA       165         STA  BUFFER
8ØC9: 9Ø Ø2       166         BCC  WEITER1
8ØCB: E6 FB       167         INC  BUFFER+1
8ØCD: 18          168 WEITER1 CLC
```

speicherplatzunabhängiger Programmierung ist man aber auf diese relativen Verzweigungsmöglichkeiten angewiesen. Nur festliegende Routinen im DOS bzw. im Interpreter können mit absolut adressierten Befehlen aufgerufen werden.
Im ersten Teil des Programms wird die Adresse des zuvor vom Applesoft-Programm dimensionierten Stringfeldes CA$() gesucht. Wird das Stringfeld nicht gefunden, dann wird mit einer Fehlermeldung die Programmausführung abgebrochen und ein Warmstart durchgeführt. Das erfolgt auch, wenn CA$() mehrdimensional angelegt worden ist. Im Erfolgsfalle wird der Pointer DESCR auf die Adresse des ersten Stringdeskriptors gerichtet.
Anschließend werden der Filezähler auf 0 gesetzt, die Adresse des IOB eingelesen und abgelegt, der erste Catalog-Sektor eingelesen und die Adresse des 256-Byte-Sektorpuffers im IOB ermittelt. Zu der Pufferadresse wird als nächstes ein Offset addiert, um den Pointer auf den ersten Catalog-Eintrag zu richten.

In den ersten beiden Bytes eines Catalog-Eintrags stehen die Track- und Sektornummer des ersten TSL-Sektors des Files (TSL = Track Sector List). Ist die Tracknummer 0, dann existieren keine weiteren Einträge im Directory mehr. Ist die Tracknummer $FF, dann ist dieser File gelöscht.

Im dritten Byte steht die Information über den Filetyp. Genaueres darüber findet man im DOS-Handbuch und in (2). An dieser Stelle wird entschieden, ob der Filename in das Stringfeld übertragen wird oder der nächste Catalog-Eintrag untersucht werden soll. Durch einen Poke-Befehl, der das Argument des CMP-Befehls ändert (s. Listing 9, Zeile 144), kann der Filetyp verändert werden. Überschreibt man den nachfolgenden Branch (BNE) mit NOPs, dann werden alle Filenamen in das Stringfeld übertragen. Im folgenden wird der Zähler FCOUNT erhöht und gegebenenfalls der nächste Catalog-Sektor eingelesen. Die Pufferadresse zum nächsten Catalog-Eintrag wird errechnet, und das Programm verzweigt zum Abarbeitungsteil. Das Spiel wiederholt sich so lange, bis eine Tracknummer 0 gefunden wird, d.h. kein weiterer Eintrag mehr existiert.
Beim Übertragen des Filenamens wird zunächst überprüft, ob er kürzer oder ebensolang ist wie derjenige Name, der beim letzten Programmaufruf an dieser Stelle eingetragen wurde. Ist dies der Fall, dann wird der gleiche Speicherplatz für den neuen Namen benutzt. Damit wird verhindert, daß bei mehrfachem Auslesen des gleichen Catalogs ständig neuer Stringplatz geschaffen wird und es entsprechend früher zur Garbage-Collection kommt. Ferner muß beachtet werden, daß der Filename erst bei (BUFFER)+3 beginnt. Das wird bei der Übertragung des Filenamens berücksichtigt.

```
80CE: 90 C8     169          BCC  FILE
                170 * Hier werden die Filenamen in das Stringfeld eingelesen
                171 *
80D0: 98        172 LESEN    TYA             ;Y sichern
80D1: 48        173          PHA
80D2: 38        174          SEC
80D3: E9 02     175          SBC  #$02       ;-2 = Länge
80D5: A0 00     176          LDY  #$00
80D7: D1 5E     177          CMP  (DESCR),Y  ;vergleiche mit alter Länge
80D9: F0 17     178          BEQ  LESEN1     ;wenn gleich oder kleiner dann
80DB: 30 15     179          BMI  LESEN1     ;keinen neuen Platz schaffen
80DD: 20 52 E4  180          JSR  GETSP      ;Platz für String schaffen
80E0: 48        181          PHA             ;Akku sichern
80E1: 98        182          TYA
80E2: A0 02     183          LDY  #$02
80E4: 91 5E     184          STA  (DESCR),Y  ;Deskriptor übertragen
80E6: 88        185          DEY
80E7: 8A        186          TXA
80E8: 91 5E     187          STA  (DESCR),Y
80EA: 88        188          DEY
80EB: 68        189          PLA
80EC: 91 5E     190          STA  (DESCR),Y
80EE: 68        191          PLA             ;Y zurückholen
80EF: A8        192          TAY             ;Y = Länge also > 0
80F0: D0 0E     193          BNE  LOOP       ;weiter bei LOOP
80F2: 91 5E     194 LESEN1   STA  (DESCR),Y  ;Länge in Deskriptor eintragen
80F4: C8        195          INY
80F5: B1 5E     196          LDA  (DESCR),Y  ;alten Wert aus Deskriptor
80F7: 85 71     197          STA  BEGINST    ;in Stringposition
80F9: C8        198          INY             ;übertragen
80FA: B1 5E     199          LDA  (DESCR),Y
80FC: 85 72     200          STA  BEGINST+1  ;Stringpointer setzen
80FE: 68        201          PLA
80FF: A8        202          TAY
8100: B1 FA     203 LOOP     LDA  (BUFFER),Y ;Filenamen übertragen
8102: 88        204          DEY             ;Achtung: Namensanfang
8103: 88        205          DEY             ;= BUFFER + 3
8104: 88        206          DEY             ;also Y-3
8105: 29 7F     207          AND  #$7F       ;High-Bit löschen
8107: 91 71     208          STA  (BEGINST),Y;und in Stringbereich übertragen
8109: C8        209          INY
810A: C8        210          INY             ;= Y-1 insgesamt
810B: C0 02     211          CPY  #$02       ;Ende Filename?
810D: D0 F1     212          BNE  LOOP       ;nein, dann nächst. Zeichen
810F: E6 08     213          INC  COUNT      ;Filezähler erhöhen
8111: 18        214          CLC             ;DESCR auf nächsten
                215 * Descriptor setzen
8112: A5 8F     216          LDA  LENGTH     ;Var Länge
8114: 65 5E     217          ADC  DESCR      ;zu DESCR addieren
8116: 85 5E     218          STA  DESCR
8118: 90 A0     219          BCC  WEITER
811A: E6 5F     220          INC  DESCR+1
811C: D0 9C     221          BNE  WEITER     ;DESCR+1 immer > 0
```

4. Applesoft-Demos

Die beiden kurzen Demoprogramme STRCAT.DEMO1 und STRCAT.DEMO2 erklären sich selbst und sollen lediglich zeigen, wie das Maschinenprogramm vom Applesoft-Programm modifiziert und aufgerufen wird.

Literaturhinweise

(1) Matthias Buck: Apple II ROM-Listing, Aachen 1984
(2) Don Worth, Pieter Lechner: Beneath Apple DOS, Reseda 1981

Kurzhinweise

1. Zweck:
Einlesen von Catalog-Dateinamen in String-Array unter DOS 3.3
2. Konfiguration:
II+/e/c; 48K-DOS 3.3 oder Diversi-DOS 2C (kein in die LC geschobenes DOS!).
3. Test:
RUN STRCAT.DEMO1 oder STRCAT.DEMO2
4. Sammeldisk:
T.STRCAT
STRCAT
STRCAT.DEMO1
STRCAT.DEMO2
5. Sonstiges: STRCAT ist relokativ und kann deshalb an eine andere Stelle als $8000 eingeladen werden.

TIPS UND TRICKS IN PASCAL

Teil 8: Eine Utility-Unit mit schnellen und nützlichen Prozeduren

von Dieter Geiß

Nach der vielen „Theorie" der letzten Teile soll hier wieder einmal die praktische Seite zum Zuge kommen. Die nachfolgend vorgestellte Unit beinhaltet 16 Prozeduren und Funktionen, die in vielen Programmen nützlich eingesetzt werden können. Bei der Implementierung kam es mir vor allem auf Geschwindigkeit und Einzigartigkeit an. Daher sind eine Reihe der Prozeduren in Assembler geschrieben. Wiederum andere greifen auf Variablen des Pascalsystems zu, die gelesen oder verändert werden können.

1. Implementierung

Wer die Unit benutzen möchte, muß folgende Schritte beachten:
1. Mit Hilfe von GETDOS den File UTILITY.TEXT, den File UTIL.ASS.TEXT und den File DEMO.TEXT auf eine Pascal-Diskette überspielen.
2. UTILITY.TEXT zu UTILITY.CODE compilieren.
3. UTIL.ASS.TEXT zu UTIL.ASS.CODE assemblieren.
4. Linken der beiden Files: Der Host-File UTILITY.CODE wird mit dem Lib-File UTIL.ASS.CODE gelinkt. Als Output-File gibt man z.B. „UTIL.LIB." oder auch „SYSTEM.LIBRARY." an, wenn dieser nicht schon auf der Diskette existiert und damit zerstört werden könnte.

Achtung: Der File UTIL.LIB enthält aber nun nicht nur das Codesegment UTILITY, sondern auch das Segment PASCALSYSTEM, welches als Übersetzungsrahmen integriert werden mußte. Will man nun die Utility-Unit in seine SYSTEM.LIBRARY integrieren, so startet man das Programm LIBRARY.CODE und gibt als Output Code File und als Link Code File jeweils SYSTEM.LIBRARY an. Nachdem alle Segmente mit Hilfe des „=" Befehls aus der alten SYSTEM.LIBRARY in die neue kopiert wurden, holt man als neuen Link File UTILITY.LIB, von dem man nur Segment 27, also nicht Segment 0, in einen freien Slot kopiert. Dann kann man das Library-Programm verlassen.
Für das kurze Demoprogramm zum Testen der Unit gehen wir davon aus, daß die Schritte 1 bis 4 durchgeführt wurden und der Name der fertig gelinkten Unit SYSTEM.LIBRARY heißt. Dann kann das Demoprogramm DEMO.TEXT compiliert und ausgeführt werden.
Die Tricks, die in der Unit angewendet werden, wurden schon in vorausgegangenen Teilen erklärt. Aus Teil 4 sind die Wirkungen der beiden Compiler-Optionen {$U-} und {$E+} bekannt. Mit Hilfe der ersten kann man Systemprogramme oder -units übersetzen lassen; die zweite erlaubt private Files in Units. Die Definitionen der Konstanten und Typen für Directory und Files wurden im Artikel „Pascal Directory unter der Lupe" in Heft 1/1985 bzw. im Teil 6 dieser Serie dargelegt. Auf sie muß deshalb nicht näher eingegegangen werden, so daß nur noch die Wirkung der einzelnen Prozeduren erklärt werden soll.

2. Die Funktionen und Prozeduren

Funktion Addr

Um die Adresse einer beliebigen Variablen zu erhalten, ruft man die Funktion Addr auf. Sie liefert einen Integer-Wert, der als Zeiger (Adresse) der an sie übergebenen Variablen dient. Das Beispiel zur Prozedur Append zeigt einen Aufruf von Addr.

Prozedur UpperChar

Sie wandelt einen an sie übergebenen Kleinbuchstaben, der also im Bereich von „a" bis „z" liegt, in einen Großbuchstaben um. Andere Zeichen werden nicht verändert.

Prozedur UpperString

Sie wandelt einen gesamten String in Großbuchstaben um. Ist der aktuelle Parameter ein String, der kürzer als 80 Zeichen ist, muß vor dem Aufruf die {$V-} Option eingeschaltet werden, da sonst der Fehler 175 auftritt.

Prozedur EatSpaces

Sollen aus einem String alle Leer- und Ctrl-Zeichen entfernt werden, so benutzt man EatSpaces. Der erste Parameter ist ein String, für den das gleiche gilt wie bei der Prozedur UpperString. Als zweiter Parameter entscheidet der Boolean-Wert „Upper", ob gleichzeitig alle Buchstaben in Großschrift umgewandelt werden sollen. Die Prozedur kann z.B. dazu benutzt werden, einen Filenamen zu analysieren, da das System beim Öffnen eines Files keine Leer- und Ctrl-Zeichen erlaubt. EatSpaces, UpperString, UpperChar und Addr sind in Assembler geschrieben und daher besonders effizient.

Prozedur AddSuffix

Sie hat zwei String-Parameter, der erste (S) ist sowohl Ein- als auch Ausgabe, der zweite (Suffix) eine Eingabe. An den String S wird der Parameter Suffix angehängt, wenn er nicht schon vorhanden ist. Zuvor werden aus beiden Eingabestrings alle Leer- und Ctrl-Zeichen entfernt und die Strings in Großbuchstaben umgewandelt. Ist danach der letzte Buchstabe in S ein Punkt, so wird die Anhängung des Suffixes unterbunden und der Punkt eliminiert. Ist S leer, so wird ebenfalls kein Suffix angehängt. Fehlt hinter einer Gerätenummer der Doppelpunkt, so wird er angehängt. Eine vorhandene Längenspezifikation für einen File wird ebenfalls berücksichtigt. An einen Gerätenamen oder einen Diskettennamen wird kein Suffix angehängt. Die Prozedur wird vor allem für die Analyse von interaktiv erfragten Datei- oder Gerätenamen benutzt. Beispiele für alle Sonderfälle:

```
S := 'test.code';
AddSuffix (S, '.code');
Ergebnis: S = 'TEST.CODE';

S := 'FILE.';
AddSuffix (S, '.text');
Ergebnis: S = 'FILE';

S := '   ';
AddSuffix (S, '.FOTO');
Ergebnis: S = '';

S := '#1';
AddSuffix (S, '.text');
Ergebnis: S = '#1:';

S := 'BILD[16];
AddSuffix (S, '.GRAF');
Ergebnis: S = 'BILD.GRAF[16]';

S := 'systerm:';
AddSuffix (S, '.text');
Ergebnis: S = 'SYSTERM:'
```

Prozedur Append

Wie schon in Teil 6 angekündigt, wird nun eine Prozedur vorgestellt, mit deren Hilfe man Daten an einen File anhängen kann. Die beiden Parameter sind Faddr, die Adresse eines Files, und Ftitle, d.h. der Name des Files, an den etwas angehängt werden soll. Das Problem, an einen vorhandenen Textfile etwas anzuhängen, kennen Sie sicher. Es gibt die zwei Möglichkeiten, den Textfile mit RESET oder REWRITE zu öffnen. Im ersten Fall steht der interne Filepointer auf dem ersten oder zweiten Zeichen des Textfiles, je nachdem, ob es mit Hilfe eines Files vom Typ TEXT oder eines Files vom Typ INTERACTIVE geöffnet wurde. Im zweiten Fall ist der ursprüngliche File gar nicht mehr zugreifbar. Würde man auf konventionelle Weise ein Append implementieren, so müßte die Routine etwa folgende Gestalt haben:

```
reset (F, Name);
while not eof (F) do get (F);
```

Dabei ist „F" eine Filevariable und „Name" der Filename. Bei einem langen File würde dieser Algorithmus in die Minuten, fast sogar in die Stunden gehen. Eine einfachere Möglichkeit ist das „Umbiegen" der Filepointer, namentlich „Fnxtblk" und „Fnxtbyte". Der interne Aufbau von Files wurde schon in Teil 6 erklärt. Der Algorithmus in der Prozedur Append baut darauf auf und muß noch einige Sonderfälle unterscheiden, die im Kommentar angegeben sind.
Die Benutzung der Prozedur ist einfach. Statt eines RESET wird einfach ein „Append" verwendet. Falls es einen Fehler beim Öffnen des Files gibt, kann dieser mit Hilfe der Funktion IORESULT abgefragt werden. Die folgenden Beispiele sind syntaktisch nicht immer vollständig:

```
var F : text;
Append (Addr (F), 'TEXTFILE.TEXT');
if IOresult <> 0 then
 rewrite (F, 'TEXTFILE.TEXT');

var F : file of integer;
Append (Addr (F), 'INTEGERFILE');

var F : file;
Append (Addr (F), 'BLOCKFILE');
```

Wie man sieht, wird die Funktion „Addr" benutzt, weil „Append" kein Maschinenprogramm ist, an welches wie bei „Addr" beliebige Parameter übergeben werden können. Ein Parameter von irgendeinem der vielen Filetypen wäre aber inkompatibel zu allen anderen Filetypen, so daß lediglich die Adresse einer Filevariablen übergeben werden kann. Diese wird in der Prozedur in eine „Fibp"-Variable geschoben, die wiederum in einen beliebigen Filetyp umgewandelt wird, damit ein RESET ausgeführt werden kann. Statt „interactive" hätte man auch „text" oder „file of integer" usw. schreiben können.

Prozedur GetFileinfo

An sie wird ebenfalls die Adresse eines Files übergeben. Als Ausgabe sind die vier wichtigen Parameter „Fisblkd", „Funit", „Fvid" und „Ftid" vorgesehen. Mit ihnen kann z.B. festgestellt werden, ob ein File ein Disk- oder ein Gerätefile ist, um damit schnell auf dieses Gerät etwas ausgeben zu können. Beispiel:

```
var F : interactive;
    S : string;
    usw.
readln (S);
reset (F, S);
S := 'Hallo';
GetFileinfo (Addr (F), Fisblkd,
  Funit, Fvid, Ftid);
if Fisblkd
then write (F, S)
else unitwrite (Funit, S [1],
  length (S));
```

Prozedur ListDir

In vielen Anwenderprogrammen und Editoren benötigt man bisweilen ein Inhaltsverzeichnis einer Diskette. An "ListDir" übergibt man die Unitnummer eines Diskettenlaufwerks, also 4, 5, 9, 10, 11 oder 12, und eine Menge von Filesorten, die man aufgelistet haben möchte. Ist die Art "Untypedfile" oder "Securedir" dabei, so handelt es sich um einen Diskettennamen, der dann auch ausgegeben wird. Beispiel:

```
write ('Von welchem Laufwerk
  wollen Sie ein Textfile einlesen?');
readln (Unitnum);
ListDir (Unitnum, [Untypedfile,
  Textfile]);
```

Prozedur GetDate

Sie holt das Systemdatum und speichert es im Parameter „Date".

Prozedur SetDate

Ein Aufruf setzt das Systemdatum auf den Wert im Parameter „Date".

Prozedur GetWork
Die sechs Strings, die bei einem Aufruf von „GetWork" übergeben werden, enthalten die Namen der Workfiles (Text und Code) und die Diskettennamen, auf denen sie sich befinden. „Workvid" und „Worktid" geben Disketten- und Filename des Workfiles an, wenn es mit dem Get-Befehl des Filers geholt wurde (ohne Suffix). „Symvid" und „Symtid" geben Disketten- und Filename des Textfiles an, das als Workfile definiert ist (mit Suffix, auch SYSTEM.WRK). „Codevid" und „Codetid" geben Disketten- und Filename des Codefiles an, der als Workfile definiert ist (mit Suffix, auch SYSTEM.WRK). Die Prozedur kann benutzt werden, um in einem Anwenderprogramm (z.B. einem Editor) den Workfile zu bevorzugen (Default Files).

Prozedur GetErrorstuff
Wenn jemand einen Editor schreiben und ihn nicht als Systemprogramm implementieren will, so braucht er die Fehlerinformation, die der Compiler dem Editor übergibt, also den Ort und die Art des Fehlers. Nur das Programm im File SYSTEM.EDITOR kann auf diese Fehler zugreifen, wenn man ein „E" im Compiler eingibt, nachdem ein Fehler aufgetreten ist. Wird der Editor nicht aufgerufen, so wird die Fehlerinformation immer wieder gelöscht. „Errsym" gibt die Byteposition des Fehlers innerhalb eines Blocks an, „Errblk" gibt die Blocknummer des Fehlers im Textfile an, und „Errnum" ist die Fehlernummer laut SYSTEM.SYNTAX.

Prozedur GetPrefix
Durch sie kann man das Präfix erfahren, das man im Filer mittels des Prefix-Befehls setzen kann.

Prozedur SetPrefix
Der String, der als Parameter nicht mehr als sieben Buchstaben umfassen darf, gibt das neue Präfix des Systems an.

Prozedur GetEmptyheap
Um die Variable „Emptyheap" des Pascalsystems zu erhalten, ruft man GetEmptyheap mit einem Integer-Parameter auf. Die Bedeutung und Anwendung dieser Prozedur wird gleich bei „SetEmptyheap" erklärt.

Prozedur SetEmptyheap
Die Variable „Emptyheap" wird vom Pascalsystem nach dem Initialisieren aller benötigten Variablen, die dynamischen Speicherplatz brauchen, auf den Wert des Heap-Pointers gesetzt. Dieser Wert bezeichnet also den niedrigsten Wert, den der Heap-Pointer erhalten kann, ohne daß Variablen des Pascalsystems gelöscht werden. Dieser Wert ist normalerweise 2418 = $0972. Könnte man diesen Wert hochsetzen, dann würde man mit dieser Aktion Speicherplatz reservieren, auf den weder das Pascalsystem noch ein Anwenderprogramm zugreifen kann – vorausgesetzt, es wendet keinen Trick an. Der Speicherplatz bleibt bis zum erneuten Booten oder bis zum Rücksetzen des Emptyheap reserviert.

Da auf dem Heap dynamische Variablen und beim Öffnen von Files auch das Directory liegen (Global Directory Pointer des Syscom = Gdirp), sollte man den „Emptyheap" nur am Ende eines Programms verändern, es sei denn, man benutzt weder Files noch dynamische Variablen. Dann müßte man den Heap-Pointer selbst hochsetzen, wie das im Beispiel unten gezeigt wird. Das Pascalsystem führt nach Ausführen eines Programms unter anderem ein
release (Emptyheap)
durch, welches den Heap-Pointer auf den neuen Wert setzt. Was man mit dem gewonnenen Speicherplatz anfängt, ist eine Frage, die der Programmierer zu beantworten hat. Eine Möglichkeit wäre der Anschluß von Gerätetreibern ohne Benutzung des von Apple an Software-Entwickler ausgegebenen speziellen SYSTEM.ATTACH-Programms, welches ebenfalls den „Emptyheap" hochsetzt. Im folgenden Beispiel werden 200 Bytes reserviert und auf Null gesetzt.

```
var Heap      : ↑integer;
    Emptyheap : integer;

begin
  mark (Heap); {Setze Gdirp auf nil}
  GetEmptyheap (Emtptyheap);
  moveleft (Emptyheap, Heap,
    size_of (Heap));
  fillchar (Heap↑, 200, 0);
  Emptyheap := Emptyheap + 200;
  SetEmptyheap (Emptyheap);
  moveleft (Emptyheap, Heap,
    size_of (Heap));
  release (Heap);
  {ab hier können wieder Files und
  dynamische Variablen benutzt werden}
end.
```

Soviel zur Utility-Unit, von der nach Bedarf auch einzelne Prozeduren ausgespart werden können.
Das nächste Mal soll ein Patch vorgestellt werden, mit dessen Hilfe man zu praktisch jedem Zeitpunkt eine Hardcopy des Apple-IIe-80-Zeichen-Bildschirms erstellen kann.

Kurzhinweise

1. Zweck:
Hilfsroutinen für UCSD-Pascal
2. Konfiguration:
II+/e/c; UCSD-Pascal 1.1, 1.2
3. Test:
E(xecute DEMO. Näheres siehe Aufsatz
4. Sammeldisk:
UTILITY.TEXT
UTIL.ASS.TEXT
DEMO.TEXT
5. Sonstiges:
Die Dateien müssen zunächst mit GETDOS von der DOS-3.3-Sammeldisk auf Ihre UCSD-Arbeitsdisk konvertiert und dann compiliert, assembliert und gelinkt werden.

UTILITY.TEXT

```
{$U-}
{$E+}
{Uebersetzungsrahmen fuer die Unit-Utility, 14. Januar 1986}

program Pascalsystem;

const Maxunit     =    12;
      Maxdir      =    77;
      Vidleng     =     7;
      Tidleng     =    15;
      Fblksize    =   512;
      Dirblk      =     2;
      Strleng     =    80;
      EOL         =    13;
{-------------------------------------------------------------}
type  Unitnum     = 0..Maxunit;
      Dirrange    = 0..Maxdir;
      Vid         = string [Vidleng];
      Tid         = string [Tidleng];

      Daterec     = packed record
                      Month : 0..12;
                      Day   : 0..31;
                      Year  : 0..100
                    end; {Daterec}

      Filekind    = (Untypedfile, Xdskfile, Codefile, Textfile,
                     Infofile, Datafile, Graffile, Fotofile,
                     Securedir, Subsvol);

      Direntry    = packed record
                      Dfirstblk : integer;
                      Dlastblk  : integer;
                      case Dfkind : Filekind of
                        Securedir,
                        Untypedfile : (Filler_1  : 0..4095;
                                       Dvid      : Vid;
                                       Deovblk   : integer;
                                       Dnumfiles : Dirrange;
                                       Dloadtime : integer;
                                       Dlastboot : Daterec);
                        Xdskfile,
                        Codefile,
                        Textfile,
                        Infofile,
                        Datafile,
                        Graffile,
                        Fotofile,
                        Subsvol     : (Filler_2  : 0..2047;
                                       Dstatus   : boolean;
                                       Dtid      : Tid;
                                       Dlastbyte : 1..Fblksize;
                                       Daccess   : Daterec)
                    end; {Direntry}
      Directory   = array [Dirrange] of Direntry;
      Windowp     = ↑Window;
      Window      = packed array [0..0] of char;
      Fibp        = ↑Fib;
      Fib         = record
                      Fwindow  : Windowp;
                      Feof,
                      Feoln    : boolean;
                      Fstate   : (Fjandw, Fneedchar, Fgotchar);
                      Frecsize : integer;
                      case Fisopen : boolean of
                        true : (Fisblkd   : boolean;
                                Funit     : Unitnum;
                                Fvid      : Vid;
                                Freptcnt,
                                Fnxtblk,
                                Fmaxblk   : integer;
                                Fmodified : boolean;
                                Fheader   : Direntry;
                                case Fsoftbuf : boolean of
                                  true : (Fnxtbyte,
                                          Fmaxbyte  : integer;
                                          Fbufchngd : boolean;
                                          Fbuffer   : packed
                                  array [0..Fblksize] of char))
                    end; {Fib}

      Inforec     = record
                      Filler1 : array [0..1] of integer;
                      Errsym,
                      Errblk,
                      Errnum  : integer;
                      Filler2 : array [0..4] of integer;
                      Workvid,
                      Symvid,
                      Codevid : Vid;
                      Worktid,
                      Symtid,
                      Codetid : Tid
                    end; {Inforec}
{-------------------------------------------------------------}
var   Filler1     : array [0..6] of integer;
      Userinfo    : Inforec;
      Emptyheap   : integer; {normlerweise ↑integer}
      Filler2     : array [0..3] of integer;
      Syvid,
      Dkvid       : Vid;
      TheDate     : Daterec;
{-------------------------------------------------------------}
unit Utility; intrinsic code 27;

interface

type Filekind = (Untypedfile, Xdskfile, Codefile, Textfile,
                 Infofile, Datafile, Graffile, Fotofile,
                 Securedir, Subsvol);

     Kindset  = set of Filekind;

     Daterec  = packed record
                  Month : 0..12;
                  Day   : 0..31;
                  Year  : 0..100
                end; {Daterec}

function  Addr (var Anyvariable) : integer;
procedure UpperChar (var C : char);
procedure UpperString (var S : string);
procedure EatSpaces (var S : string; Upper : boolean);
procedure AddSuffix (var S : string; Suffix : string);
procedure Append (Faddr : integer; Ftitle : string);
procedure GetFileinfo (    Faddr   : integer;
                       var Fisblkd : boolean;
                       var Funit   : integer;
                       var Fvid    : string;
                       var Ftid    : string);
procedure ListDir (Funit : integer; Kinds : Kindset);
procedure GetDate (var Date : Daterec);
procedure SetDate (Date : Daterec);
procedure GetWork (var Workvid, Symvid, Codevid : string;
                   var Worktid, Symtid, Codetid : string);
procedure GetErrorstuff (var Errsym, Errblk, Errnum : integer);
procedure GetPrefix (var Prefix : string);
procedure SetPrefix (Prefix : string);
procedure GetEmptyheap (var Heap : integer);
procedure SetEmptyheap (Heap : integer);

implementation
{-------------------------------------------------------------}
function Addr {(var Anyvariable) : integer};
external;
{-------------------------------------------------------------}
procedure UpperChar {var C : char};
external;
{-------------------------------------------------------------}
procedure UpperString {var S : string};
external;
{-------------------------------------------------------------}
procedure EatSpaces {var S : string; Upper : boolean};
external;
{-------------------------------------------------------------}
procedure AddSuffix {var S : string; Suffix : string};

var Leng       : integer;
    Lpos       : integer;
    Suffixleng : integer;
    Sizestr    : string;
    Tail       : string;

begin {AddSuffix}
  EatSpaces (S, true);
  EatSpaces (Suffix, true);
  Leng       := length (S);
  Suffixleng := length (Suffix);
  if Leng > 0 then
  begin
    if S [Leng] <> '.'
    then
```

```
      begin
        if Leng < Strleng then
        if (S [1] = '$') and (pos (':', S) = Ø)
        then insert (':', S, Leng + 1);
        Sizestr := '';
        Tail    := '';
        Lpos    := pos ('[', S);
        if Lpos > Ø then
        begin
          Leng    := length (S) - Lpos + 1;
          Sizestr := copy (S, Lpos, Leng);
          delete (S, Lpos, Leng)
        end; {if}
        Leng := length (S);
        if Leng > Ø then
        if S [Leng] <> ':' then
        begin
          if Leng > Suffixleng then Tail :=
            copy (S, Leng - Suffixleng + 1, Suffixleng);
          if (Tail <> Suffix) and (Leng + length (Sizestr)
            <= Strleng - Suffixleng) then
          begin
            moveleft (Suffix [1], S [Leng + 1], Suffixleng);
            S [Ø] := chr (Leng + Suffixleng)
          end {if}
        end; {if}
        S := concat (S, Sizestr)
      end {if}
    else S [Ø] := chr (Leng - 1)
  end {if}
end; {AddSuffix}
{-------------------------------------------------------------}



procedure Append {Faddr : integer; Ftitle : string};

var Fileinfo : Fibp;
    AnyFile  : interactive;

begin {Append}
  moveleft (Faddr, Fileinfo, size_of (Fileinfo));
  moveleft (Fileinfo↑, Anyfile, size_of (Fib));

  if Fileinfo↑.Fisopen
  then reset (Anyfile)
  else reset (AnyFile, Ftitle);

  moveleft (AnyFile, Fileinfo↑. size_of (Fib));
  with Fileinfo↑, Fheader do
  if IOresult = Ø then
  if Fisblkd then
  if Frecsize = Ø {ohne Fsoftbuf}
  then Fnxtblk := Fmaxblk
  else
    if (Dfirstblk = Dlastblk) or
       (ord (Dfkind) = ord (Textfile)) and (Fmaxblk = 2)
    then
      begin {Sonderfall fuer leere Files}
        Feof  := false;
        Feoln := false
      end {if}
    else
      if Frecsize = 1 {Textfile oder Interaktivfile}
      then
        begin
          unitread (Funit, Fbuffer, Fblksize,
            Dfirstblk + Fmaxblk - 2);
          Fnxtbyte := 512 +
            scan (-512, <> chr (Ø), Fbuffer [511]);
          Fnxtblk  := Fmaxblk - 1;
          if Fnxtbyte = 512 then
          begin
            unitread (Funit, Fbuffer, Fblksize,
              Dfirstblk + Fmaxblk - 1);
            Fnxtbyte := 512 +
              scan (-512, <> chr (Ø), Fbuffer [511]);
            Fnxtblk  := Fnxtblk + 1
          end {if}
        end {if}
      else {normaler File}
        begin
          unitread (Funit, Fbuffer, Fblksize,
            Dfirstblk + Fmaxblk - 1);
          Fnxtbyte := Fmaxbyte;
          Fnxtblk  := Fmaxblk
        end {else, else, else, if, if, with}
end; {Append}
{-------------------------------------------------------------}
```

```
procedure GetFileinfo {    Faddr   : integer;
                       var Fisblkd : boolean;
                       var Funit   : integer;
                       var Fvid    : string;
                       var Ftid    : string};

var Fileinfo : Fibp;

begin {GetFileinfo}
  moveleft (Faddr, Fileinfo, size_of (Fileinfo));
  Fisblkd := Fileinfo↑.Fisblkd;
  Funit   := Fileinfo↑.Funit;
  Fvid    := Fileinfo↑.Fvid;
  Ftid    := Fileinfo↑.Fheader.Dtid
end; {GetFileinfo}
{-------------------------------------------------------------}
procedure ListDir {Funit : integer; Kinds : Kindset};

var Num  : integer;
    I    : integer;
    N    : integer;
    CR   : char;
    Knds : set of Ø..15;
    S    : string;
    Ldir : Directory;

begin {ListDir}
  if Funit in [4, 5, 9..12] then
  begin
    moveleft (Kinds, Knds, size_of (Knds));
    unitread (Funit, Ldir, size_of (Ldir), Dirblk);
    if IOresult = Ø then
    begin
      fillchar (CR, size_of (CR), chr (EOL));
      Num := Ldir [Ø].Dnumfiles;
      N   := Ø;
      for I := Ø to Num do
      with Ldir [I] do
      if ord (Dfkind) in Knds then
      begin
        if ord (Dfkind) in [ord (Untypedfile), ord (Securedir)]
        then
          begin
            unitwrite (1, CR, 1);
            S := concat ('Volume: ', Dvid);
            unitwrite (1, S [1], length (S));
            unitwrite (1, CR, 1)
          end {if}
        else
          begin
            N := N + 1;
            S := '  ';
            if N > 9 then S [1] := chr (N div 1Ø + ord ('Ø'));
            S [2] := chr (N mod 1Ø + ord ('Ø'));
            S     := concat (S, '. ', Dtid);
            unitwrite (1, S [1], length (S))
          end; {else}
        unitwrite (1, CR, 1)
      end {if, with, for}
    end {if}
  end {if}
end; {ListDir}
{-------------------------------------------------------------}
procedure GetDate {var Date : Daterec};

begin {GetDate}
  Date := TheDate
end; {GetDate}
{-------------------------------------------------------------}
procedure SetDate {Date : Daterec};

begin {PutDate}
  TheDate := Date
end; {PutDate}
{-------------------------------------------------------------}
procedure GetWork {var Workvid, Symvid, Codevid : string;
                   var Worktid, Symtid, Codetid : string};

begin {GetWork}
  Workvid := Userinfo.Workvid;
  Symvid  := Userinfo.Symvid;
  Codevid := Userinfo.Codevid;
  Worktid := Userinfo.Worktid;
  Symtid  := Userinfo.Symtid;
  Codetid := Userinfo.Codetid
end; {GetWork}
{-------------------------------------------------------------}
```

```
procedure GetErrorstuff {var Errsym, Errblk, Errnum : integer};

begin {GetErrorstuff}
  Errsym := Userinfo.Errsym;
  Errblk := Userinfo.Errblk;
  Errnum := Userinfo.Errnum
end; {GetErrorstuff}
{-------------------------------------------------------------}
procedure GetPrefix {var Prefix : string};

begin {GetPrefix}
  Prefix := Dkvid
end; {GetPrefix}
{-------------------------------------------------------------}
procedure SetPrefix {Prefix : string};

begin {SetPrefix}
  if length (Prefix) > Vidleng then Prefix [Ø] := chr (Vidleng);
  Dkvid := Prefix
end; {SetPrefix}
{-------------------------------------------------------------}
procedure GetEmptyheap {var Heap : integer};

begin {GetEmptyheap}
  Heap := Emptyheap
end; {GetEmptyheap}
{-------------------------------------------------------------}
procedure SetEmptyheap {Heap : integer};

begin {SetEmptyheap}
  Emptyheap := Heap
end; {SetEmptyheap}
{-------------------------------------------------------------}
begin {Utility}
end; {Utility}
{-------------------------------------------------------------}
begin {Pascalsystem}
end {Pascalsystem}.
```

UTIL.ASS.TEXT

```
;*************************************************************
;*                                                           *
;* Maschinenprozeduren fuer die Unit-Utility.                *
;* Von Dieter Geiss, 7. Juli 1984                            *
;*                                                           *
;* function  Addr (var Anyvariable) : integer;               *
;* procedure UpperChar (var C : char);                       *
;* procedure UpperString (var S : string);                   *
;* procedure EatSpaces (var S : string; Upper : boolean);    *
;*                                                           *
;*************************************************************

Return          .EQU ØØ
;*************************************************************
                .MACRO PullPascal       ;holt PASCAL
                                        ;Rueckkehradresse
                 PLA                    ;vom Stack
                 STA Return
                 PLA
                 STA Return+1

                .ENDM
;*************************************************************
                .MACRO PushPascal       ;drueckt PASCAL
                                        ;Rueckkehradresse
                 LDA Return+1           ;auf den Stack
                 PHA
                 LDA Return
                 PHA

                .ENDM
;*************************************************************
                .MACRO PullAddress      ;holt Adresse eines
                                        ;Parameters vom Stack
                 PLA
                 STA %1
                 PLA
                 STA %1+1

                .ENDM
;*************************************************************
                .FUNC Addr, 1

;function Addr (var Anyvariable) : integer;

                 PullPascal

                 PLA                    ;hole Dummy-
                 PLA                    ;Funktionsergebnis
                 PLA                    ;Adresse des Parameters
                 PLA                    ;ist jetzt oben
                                        ;auf dem Stack
                 PushPascal

                 RTS
;*************************************************************
                .PROC UpperChar, 1

;procedure UpperChar (var C : char);

Addr            .EQU Ø2                 ;Adresse von C

                 PullPascal
                 PullAddress Addr       ;hole Adresse von C

                 LDY #ØØ
                 LDA (Addr),Y           ;Inhalt von C
                 CMP #"a"               ;konvertiere
                 BCC $1                 ;nur
                 CMP #"ä"               ;Buchstaben
                 BCS $1                 ;von 'a' bis 'z'
                 AND #5F                ;in Grossbuchstaben
                 STA (Addr),Y           ;speichere C

$1               PushPascal

                 RTS
;*************************************************************
                .PROC UpperString, 1

;procedure UpperString (var S : string);

Addr            .EQU Ø2                 ;Adresse von S

                 PullPascal
                 PullAddress Addr       ;hole String-Adresse

                 LDY #ØØ
                 LDA (Addr),Y           ;Laenge des Strings
                 INY                    ;erstes Zeichen
                 TAX                    ;X := length (S)

$1               BEQ $4                 ;if X = Ø then goto 4
$2               LDA (Addr),Y           ;A := S [Y]
                 CMP #"a"               ;konvertiere
                 BCC $3                 ;nur
                 CMP #"ä"               ;Buchstaben
                 BCS $3                 ;von 'a' bis 'z'
                 AND #5F                ;in Grossbuchstaben
                 STA (Addr),Y           ;S [Y] := A
$3               INY                    ;naechstes Zeichen
                 DEX                    ;X := X - 1
                 BNE $2

$4               PushPascal

                 RTS
;*************************************************************
                .PROC EatSpaces, 2

;procedure EatSpaces (var S : string; Upper : boolean);

Addr            .EQU Ø2                 ;Adresse von S
Upper           .EQU Ø4                 ;Upper-Wert
Length          .EQU Ø5                 ;Laenge von S
Dest            .EQU Ø6                 ;Zeiger

                 PullPascal

                 PLA                    ;hole Upper-Wert
                 AND #Ø1                ;boolean nur false,,true
                 STA Upper
                 PLA                    ;nur niederwertiges Byte

                 PullAddress Addr       ;hole String-Adresse

                 LDY #ØØ
                 LDA (Addr),Y           ;Laenge des Strings
                 STA Length             ;abspeichern
```

```
                STY Dest                ;Dest := Ø
                LDX #Ø1                 ;X    := 1

$1              TXA
                CMP Length              ;if X > Length
                BEQ $2                  ;then goto 5
                BCS $5
$2              TAY
                LDA (Addr),Y            ;A := S [X]
                LDY Upper               ;if not Upper then goto 3
                BEQ $3
                CMP #"a"                ;Konvertiere
                BCC $3                  ;nur
                CMP #"ä"                ;Buchstaben
                BCS $3                  ;von 'a' bis 'z'
                AND #5F                 ;in Grossbuchstaben
$3              CMP #" "                ;if A <= ' ' then goto 4
                BEQ $4                  ;d.h. Buchstaben
                BCC $4                  ;uebergehen
                INC Dest                ;Dest := Dest + 1
                LDY Dest                ;Y    := Dest
                STA (Addr),Y            ;S [Dest] := A
$4              INX                     ;X := X + 1
                BNE $1                  ;goto 1
$5              LDY #ØØ                 ;Laenge eines Strings
                LDA Dest                ;in Byte Ø
                STA (Addr),Y            ;S [Ø] := Dest

                PushPascal

                RTS
;***************************************************************
                .END
```

DEMO.TEXT

```
program Demo;

uses Utility;

var S : string;
    U : integer;
    D : Daterec;

begin
  write ('Test einiger Utility-Prozeduren');
  writeln;
  write ('Bitte einen String in Kleinbuchstaben eingeben: ');
  readln (S);
  UpperString (S);
  writeln ('Nach Konvertierung: ', S);
  writeln;
  write ('Bitte Disketten-Unitnummer eingeben: ');
  readln (U);
  ListDir (U, [Untypedfile..Subsvol]);
  writeln;
  GetDate (D);
  with D do writeln
   ('Das aktuelle Datum ist: ', Day, '.', Month, '.', Year);
  writeln;
  write ('Neuer Prefix (<RETURN> = keine Aenderung): ');
  readln (S);
  EatSpaces (S, true);
  if S <> '' then SetPrefix (S);
  GetPrefix (S);
  writeln ('Der Prefix ist: ', S);
end.
```


Börse

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

Ihre Erphi-Vertretung für die Schweiz: Beltronic, Im Chapf, 8455 Rüdlingen, Tel. 01867 31 41, Telex 825 981.

*** Ausschneiden! Aufbewahren! ***
Applesoft – MSDOS-Transfer 1. Prg 80, ab DM 2,50 macht:
K. Freimann, Andelsbachstr. 2a, 7887 Laufenburg

!!!! Da gibt's was umsonst !!!!
4 × im Jahr den neuen Katalog. Bühler Elektronik, Postfach 32, 7570 Baden-Baden

Wer hilft? Welche Hardware-Änderungen muß ich machen an meinem Diskdriver Remex RFDG60 damit er einwandfrei mit Apple II + Ehring Controller arbeitet (Holland) P. Visser Marykestr. 23 Noordwyk.

****** Apple kompatibles ******
Gelsen-Electr.
K-Schumacher Str. 124, 4650 Gelsenkirchen, Tel. 0209/83033

Endlich–Wordstar an Star SG10 ob NLQ od. Engschrift. Wir patchen Ihren WS für den Drucker SG10. für DM 40,– Bitte rufen Sie uns an. n. 19.00h Tel. 09661/1300. Biesler 8458 Sulzbach Max-Planck-Str. 12.

Medizin. Doktorarb. zu vergeben ges. med. Stud. mit guten Kenntniss. in Apple-Grafik + Assembler für Rekonstr.-Progr. in histologie. Zuschr. an Dr. Lechleuthner Anat. Anst. Pettenkoferstr. 11, 8000 München 2

Apple II + System VB 550, M. Horak, Baumeisterweg 12, 7000 Stuttgart 1.

Ramworks IIe (Batterie gepuff.) Accelerator, Uhr Z80. 80Z zu verk. Tel. Mo-Fr. 20-21 Uhr 02 12/5 19 76.

Apple IIc, 2 Laufwerke, Z80-Karte, Joy (Orig) + Software (100 Disk) + Bücher + Zeitsch. nur DM 2900,– Tel.: 069/7 89 42 67.

Apple IIe-64KB, Monitor II, DuoDisk-Laufwerk, Superserielle Karte, 80Z Karte. Incl. Software z.B. Apple Writer II, Quick File und vieles mehr sowie Handbücher.
Neupreis ohne Software DM 4300,– Verkaufspreis komplett DM 2700,– Tel. 08389/256

Hochauflös. Videoscanner + CAD (Elektro/Elektronik) für Apple IIe gesucht. Eilt! St. Richter 8650 Kulmbach, Kalte Marter 3.

Print-Files

Ein komfortables Druckprogramm für Pascal-Listings

von Frank Becker

Geht es Ihnen auch so? Beim Editieren eines Pascal-Programms auf einem Apple mit deutscher Tastatur hat man sich daran gewöhnt, daß man statt eckiger Klammern immer nur die entsprechenden Umlaute eintippt. Was mich aber immer wieder ärgert, ist die Tatsache, daß ich beim Ausdruck des Quelltextes auf dem Drucker entscheiden muß, ob alles mit deutschem oder amerikanischem Zeichensatz ausgedruckt wird, obwohl ich sowohl die eckigen Klammern als auch für die deutschen Texte die Umlaute verwende. Für diesen Fall ist das folgende Programm gedacht. Es ist für den Apple IIe bzw. IIc mit Imagewriter geschrieben worden, läßt sich aber durch Änderung der Steuercodes leicht an andere Drucker anpassen.
Das Programm druckt Pascal-Textdateien in drei wählbaren Versionen aus, nämlich mit
1) amerikanischem,
2) deutschem oder
3) gemischtem Zeichensatz.
Die 3. Version löst das angesprochene Problem. Hier werden alle Programmanweisungen mit amerikanischem Zeichensatz ausgedruckt, nur die in Anführungszeichen stehenden Texte und die Kommentare, gekennzeichnet durch „(*“ und „*)“ bzw. „ä“ und „ü“ werden mit deutschem Zeichensatz gedruckt. Außerdem werden Includefiles ordnungsgemäß verarbeitet, um auf diese Weise auch lange Quelltexte ohne großen Aufwand „in einem Stück“ ausdrucken zu können.
Weiterhin erfolgt spätestens nach 67 Zeilen ein Seitenvorschub. Um hier unschöne Trennungen zu vermeiden, kann man selbst im Text jederzeit vorher einen Seitenvorschub bewirken, indem eine zusätzliche Zeile eingegeben wird, die nur den String „(**)“ enthält. Diese Zeile wird nicht mit ausgedruckt, und der Compiler ignoriert sie bekanntlich auch, so daß sie im Text nicht stört. Natürlich werden die dadurch möglichen doppelten Seitenvorschübe verhindert, die ungewollte Leerseiten bewirken würden. Außerdem werden Leerzeilen am Anfang jeder Seite unterdrückt, da im allgemeinen ein Seitensprung zur Trennung von Textabschnitten ausreicht.
Zusätzlich ist zur Erhöhung des Eingabekomforts eine kleine String-Eingabe-Prozedur eingefügt worden. Diese Prozedur erlaubt das Abbrechen der Eingabe durch <Esc> zu jeder Zeit unter Beibehaltung des ursprünglichen Variableninhalts. Zusätzlich wird ein String-Overflow verhindert, und ein <Return> am Ende der Eingabe wird nicht mit einem Zeilensprung auf dem Bildschirm quittiert, so daß auch Stringeingaben (mit weniger als 80 Zeichen) in der letzten Bildschirmzeile ohne Scrolling funktionieren.

Pascal-Listings für Druckerei

In der Peeker-Redaktion werden Pascal-Quelltexte mit §-Zeichen wie folgt codiert:
§(→ {
§) → }
§< → [
§> →]
(* → {
*) → }
Daher können Umlaute mit eckigen und geschweiften Klammern beliebig gemischt werden. In der Regel vermeiden wir jedoch die Umlaute „ä“ und „ü“ innerhalb von Kommentarzeilen mit geschweiften Klammern, weil sonst die Listings nicht in dieser Form abgetippt werden können. us

Kurzhinweise

1. Zweck:
Druckprogramm für UCSD-Pascal-Quelltexte
2. Konfiguration:
II+/e/c; Apple-Pascal 1.1 oder 1.2;
Imagewriter; für andere Drucker Steuerzeichen im Quelltext ändern.
3. Test:
E(xecute PRINT.FILE
4. Sammeldisk:
PRINT.FILE.TEXT
5. Sonstiges:
PRINT.FILE.TEXT muß zunächst mit GET-DOS von der DOS-3.3-Sammeldisk auf Ihre Pascal-Arbeitsdiskette konvertiert und dann compiliert werden.

PRINT.FILE.TEXT

```
PROGRAM print_file;
{von Frank Becker 1985}
USES applestuff;

CONST zps = 67; {Zeilen pro Seite}

VAR drucker, datei, includedatei       : TEXT;
    d_name, i_name                     : STRING;
    antwort                            : CHAR;
    ok, cat, incl, mit_titel           : BOOLEAN;
    pause, laufwerk                    : INTEGER;
    bs,lf,aus,an,clz,inv,norm,esc,bell : CHAR;
{*****************************************************}
PROCEDURE loesche;

BEGIN
  GOTOXY (0,21);
  WRITE (clz,lf,clz,lf,clz);
  GOTOXY (0,21);
END;
{*****************************************************}
PROCEDURE linien;

BEGIN
  GOTOXY (0,19);
  WRITE ('----------------------------------------');
  WRITE ('----------------------------------------');
END;
{*****************************************************}
PROCEDURE titel_text;

BEGIN
  mit_titel := FALSE;
  PAGE (OUTPUT);
  GOTOXY (15,5);
  WRITE (inv,'                                      ');
  GOTOXY (15,6);
  WRITE (aus,'     Ausdrucken einer Textdatei     ');
  GOTOXY (15,7);
  WRITE ('                                      ',norm);
  linien;
END;
{*****************************************************}
PROCEDURE ende;

BEGIN
  loesche;
  UNITCLEAR (1);
  WRITE ('Programmdiskette einlegen und eine Taste');
  WRITE (' drücken');
  READ (KEYBOARD, antwort);
  EXIT (PROGRAM);
END;
{*****************************************************}
PROCEDURE str_eingabe (VAR zeile : STRING;
                       VAR ok    : BOOLEAN);

VAR zw     : STRING;
    taste  : CHAR;
    index  : INTEGER;
{-----------------------------------------------------}
   PROCEDURE abbruch;

   VAR j : INTEGER;

   BEGIN
     ok := FALSE;
     IF index > 1 THEN
       FOR j := 1 TO index-1 DO
         WRITE (bs,' ',bs);
   END;
{-----------------------------------------------------}
   PROCEDURE zurueck;

   BEGIN
     IF index > 1 THEN
       BEGIN
         index := PRED (index);
         WRITE (bs,' ',bs);
         DELETE (zw,index,1);
       END;
   END;
{-----------------------------------------------------}
   PROCEDURE addiere;

   BEGIN
     IF (NOT EOLN (KEYBOARD)) AND (index <= 80) THEN
       BEGIN
         WRITE (taste);
         zw := CONCAT (zw,'?');
         zw [index] := taste;
         index := SUCC (index);
       END;
   END;
{-----------------------------------------------------}
   BEGIN            {PROCEDURE str_eingabe}
     ok := TRUE;
     zw := '';
     index := 1;

     REPEAT
       READ (KEYBOARD, taste);

       IF taste = esc THEN
         abbruch
       ELSE
         IF taste IN [' '..'ß'] THEN
           addiere
         ELSE
           IF taste = bs THEN
             zurueck;
     UNTIL (NOT ok) OR EOLN (KEYBOARD);

     IF ok THEN
       zeile := zw;
   END;
{*****************************************************}
PROCEDURE catalog (disk : INTEGER);

TYPE Datum     = PACKED RECORD
                   Monat : 0..12;
                   Tag   : 0..31;
                   Jahr  : 0..100;
                 END;

     Dateityp = (untyped, defekt, Code, Text,Info,
                 Data, Graf, Foto, sec_Dir, Sub_vol);

     Eintrag  = PACKED RECORD
                  erster_Block  : INTEGER;
                  letzter_Block : INTEGER;
                  CASE Typ : Dateityp OF
                    sec_Dir,
                    untyped : (Nichts1   : 0..4095;
                               Vol_Name  : STRING [7];
                               Blockzahl : INTEGER;
                               Anzahl    : 0..77;
                               Zeit      : INTEGER;
                               Boot_Datum: Datum);
                    defekt,
                    Code,
                    Text,
                    Info,
                    Data,
                    Graf,
                    Foto,
                    Sub_vol : (Nichts2   : 0..2047;
                               Status    : BOOLEAN;
                               Datei_Name: STRING [15];
                               last_Byte : 1..512;
                               Zugriff   : Datum)
                END;   {Eintrag}

VAR Directory : ARRAY [0..77] OF Eintrag;
    i, j      : INTEGER;
{-----------------------------------------------------}
{$I-}
FUNCTION Pascal_Diskette : BOOLEAN;

BEGIN
  UNITREAD (disk,Directory,sizeof (Directory),2);

  IF IORESULT = 0 THEN
    WITH Directory [0] DO
      Pascal_Diskette := (erster_Block = 0) AND
                         (Typ IN [untyped, sec_Dir])
                         AND (anzahl IN [0..77]) AND
                         (LENGTH (vol_Name) IN [0..7])
  ELSE
    Pascal_Diskette := FALSE;
END;
{$I+}
{-----------------------------------------------------}
```

PEEKER Börse

Biete Hardware

RAM-KARTEN
Rüsten Sie Ihren Apple II+ auf:
512 KB ..... DM 383,–
768 KB ..... DM 463,–
Die Karten rüsten direkt den Hauptspeicher auf und sind auch unter Basic voll nutzbar, Erhöhung der Anzahl der HGR-Seiten bis auf 32 Seiten.
Die Karten werden komplett aufgebaut und mit Software geliefert. Versand nur gegen VR-Scheck.
Andreas Müller, Lessingstr. 17, 4000 Düsseldorf, T. 0211/776503

CP/M Karte für IIc
Einbauplatine (ohne Löben) mit Betriebssystem 2.23 und RAM-Disk verkauft für DM 300,–
Tel. 0211/347411 o. 348387

Apple IIe + Monitor + 2 LW + CP/M + 80Z + 64K + Epson FX-80+ + Lit. + ca. 150 Disks, NP ca. DM 6800,– für DM 4000,–.
Tel. 02365/42206

Maus-orig. Apple mit MOUSE-PAINT u. Interface DM 300,–. Tel. 0431/81598. **AP17** 256K-Karte u. **IBS** m. org. Software DM 250,–.
Tel. 0431/81598

Verkaufe
Apple Grafikkarte mit Farbe 1024* 1024 Bildpunkte + Software
nach 19 Uhr Tel. 0621/711604

EPSON FX-80 Drucker VB 900,–
Original Apple Disk II VB 600,–
Orig. Maus mit Karte VB 350,–
CP/M-, Drucker-, Farb- und orig. Apple Controller je VB 70,–
Tel. ab 18 Uhr 07022/45485

Typenraddrucker Silver Reed, RS2 32C-Interface, Bidirektional, 1, 8 KB-Puffer Apple II-Kompatibel neu DM 640,–. Apple Scribe-Printer (NP 1300,–) DM 400,–. Tel. 07222/6338.

Apple IIe Speichererweiterung der 1 MB Karte von 256 K auf 1 MB DM 369,– Apple IIc 10ér Block Adapter DM 99,–, Macintosh Speichererweiterung von 128 auf 512 KB DM 299,–
Macintosh Power Sound DM 149,–
XL-Enhancement-Kit DM 499,–
Fa. Reinhard Schlösser, Ismaninger Str. 108, 8000 München,
Tel. ab 18 Uhr 089/985889

IIc-Uhr, Z80c, Z-RAM Karten + CP/M!! Tel. 0212/315972 nach 20 Uhr.

II+ Nachbau, Hardware (Monitor, Maus, Floppys uvm.), Software (Apple Pascal 1.2 für DM 650,–) uvm. orig. verpackt, Zeitschriften, Bücher zu verk. Liste gegen DM 3,– in Briefmarken.
Tel. 08241/2882.

Zubehör für Apple IIe:
RAM-Works-Karte mit 512 K, Accelerator IIe, Grappler+–IF, Z-80 Softcard, Erphi-Controller, BASF-Doppelfloppy 2 × 640 K, FDC4-Controller, Exbasic Level II, Eprom Burner, 80-Zeichen-Karte mit 64 K, 2 Paddles.
Preise nach Vereinbarung.
Tel. 07138/8364

Superpreis für Drucker
C.ITOH Riteman F+ Matrixdrucker IBM PC/FX80 komp./ 105 Z/Sek./NLQ DM 750,–
C.ITOH Typenraddrucker A10-30 P/parallel/30 Z/Sek. DM 1190,–
A10–30 RM/seriell/30 Z/Sek. DM 1090,–
Alle Preise incl. Mehrwertsteuer zuzüglich Versandkosten.
TEACH-Computer, Siemensstr. 22, 7257 Ditzingen, T. 07156/30001-34

IIe komp., 64K, Z80, 80Z, 2 TEAC-LW 55F (je 1MB, 160+35 Spur), ext. Tastatur, Monitor, BS: DOS+CP/M Pr.: VB DM 3500,–, SW umsonst dazu: Compiler, Datenbank uvm. Tel. pr. 089/951429, o. 089/63645667

Macintosh f. Studenten 25 % unter Liste; Apple-Produkte (orig. + kompat.) EDV-Zubehör; Drucker (z. B. Juki, etc.) Datent. Schoeben, Tel. 0821/152377

Xebec IBM-Hostadapter für Xero D Subsystem mit Install-Software neu DM 250,–.
Tel. 0831/95558

Megaboard-HDC VB DM 800,– 8" FDC VB DM 450,– Org. UCSD-PASCAL 1.1 + HB VB DM 100,–. Tel. 07231/61816

Frei prog. Tastatur für II plus (ähnl. Brose siehe UM 2/85) für DM 350,– (DM 695,–).
Tel. (06221) 20250 Kehrel.

IIe, 80Z + 64K, Matrixdr. 80Z/Sek. grafikfäh., Juki 6100 Typenrad 20 Z/Sek. + 2. Disclaufw./Monitor Textverarb. progr. VB DM 2500,–. Tel. 07021/59462.

RAMFACTOR (bat. gepuff. RAM-Karte)! Accelerator, Uhr, Z80, 80Z+ 64K, usw.
Tel. 0212/315972 nach 20 Uhr.

Apple-Graphic-Tablet DM 700,– TEAC FD55F+ERPHI-Contr. DM 900,– mit 3 Mon. Garantie. Tel. 06081/15519

Karten für IIe, 80Z+64K, TIMER von AE je DM 50,–.
Tel. 06103/82334

DISK II-Laufwerk, orig. Apple, DM 300,–. Tel. 02666/704

Apple II. DFÜ-Kermit Pascal satt, Public Domain in DOS u. CP/M. Je Volume DM 15,–. Bahnhofssimulation, Sprachen (S.A.L.), Schulprogr.
Gratisinfo: Fa. Waltraud Muhle, Waldwinkel 3, 2105 Seevetal 3

Biete Software

DISKETTEN
* 5¼", 48 tpi, DM 1,20 *
* 3½", 135 tpi, DM 4,10 *
* auch andere, bes. Garantie *
* Allg. Austro-Ag, Ringstraße 10 *
* D-8057 Eching, T. 08133/6116 *

68000-SYSTEM für Apple II mit Prozessorkarte und Software: Assembler, Editor, Monitor, Tracer (in 6800-Code) und komfortable I/O-Treiber (DOS, 80Z, Printer). KOSTENLOSE INFO anfordern bei Krey, Christianistr. 18, 2300 Kiel

Vereins-, Immobilien-, Kunden-, Adreß-, Kassen- und Dateiverwaltung. Info 0,80 DM von E. Heinz, Püttkampsweg 13, 2000 Hamburg 52

UNI Software
Print, PLOT... & Show it! alle Business Grafiken 225
Regression XVII 225
Cluster Analyse 125
Books & Software, T. 09571/3182

Verkaufe Original
Appleworks Utilities:
Macroworks 80,– Mailworks, Sideway DMP-Charger je 150,– und Print Shop Library Disk #3 50,–.
Tel. 0211/347411 oder 348387

OHO 256/512-Byte Ramkarte als Ramdisk unter ProDOS 1.0-1.1.1 Disk + Anl. DM 20,– (NNV). Info von Postfach 1323, 2400 Lübeck 1 od.
Tel. 0451/895697 ab 16 Uhr.

Topsoftware für Apple II:
Fußballtabellenverwaltung: 30,–
Basic-Pascal Konvertierer: 150,–
MS Fortran: 550,–, Gato: 120,–
Info (0,80 DM) bei: W. Rittmeyer, Wehrbruchweg 30, 4060 Viersen 1

Original Software: (m. Handbuch) DMP-Charger, Apple Writer IIe Input 2.0, ProDOS Editor 1.0, Music Construction Set, Print Shop, Robo CAD-System, Sideways, Preise nach Vereinbarung.
Tel. 07138/8364

> 30 Zeichensätze f. EPSON FX-80. DM 130,–. Info DM 3,–. Achim Stößer, Hauptstr. 83, 7552 Durmersheim.

ÖKOLOPOLY für Apple II
Disk DM 30,–. Info gratis.
Tel. 0921/56611 ab 17 Uhr.

Software Uhr für Apple II+, e, c, Zeitschaltmöglichkeit Diskette + Anleitung DM 25,– Oecking
Tel: Do. 0231/391920

Suche Hardware

Suche Flachbildschirm für IIc v.
Priv. oder Hdl. Tel. 06103/82334

Suche Software

Suche für IIc Finanzmanager, private Haushaltsbuchführung od. Vermögensverwaltung alles mit Simulationsmöglichkeit
Tel. 06047/1822

TUTSIM für IIc günstig zu kaufen ges., Tel. 0214/68232 ab 18 Uhr

Tausch

Habe gute Software zum TAUSCH!!
Tel. 0043/4875/6665 (von BRD)
04875/6665 (von AUT)

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 72,– (Inland) inkl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt z. Zt. DM 72,– plus DM 18,– 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, ca. DM 40,–
- ☐ 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,–
- ☐ Softbreaker 1.0, Programm, DM 20,–
- ☐ 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.

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

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

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

```
BEGIN           {PROCEDURE catalog}
  PAGE (OUTPUT);
  GOTOXY (1Ø,Ø);

  IF NOT Pascal_Diskette THEN
    BEGIN
      WRITE (inv,' In Laufwerk ',disk);
      WRITE (' befindet sich keine ');
      WRITELN ('PASCAL - Diskette !!! ',norm);
      EXIT (catalog);
    END;

  WRITE ('          ',inv,' Katalog von  ');
  WRITELN (Directory [Ø].vol_Name,': ',norm);
  WRITELN;
  j := Ø;

{Es werden nur Textdateien angezeigt}

  FOR i := 1 TO Directory [Ø].anzahl DO
    WITH Directory [i] DO
      IF Typ = Text THEN
        BEGIN
          GOTOXY (3Ø * (j MOD 3) + 3,j DIV 3 + 2);
          WRITE (Datei_Name);
          j := SUCC (j);
        END;

  IF j = Ø THEN
    WRITELN ('          Keine Textdatei vorhanden !!!');

  linien;
END;
{*****************************************************}
{$I-}
FUNCTION existiert (name : STRING) : BOOLEAN;

BEGIN
    IF incl THEN
      RESET (includedatei, name)
    ELSE
      RESET (datei, name);

  {ggfls. ".TEXT" an den Dateinamen anfuegen}
  IF IORESULT <> Ø THEN
    IF incl THEN
      RESET (includedatei, CONCAT (name,'.TEXT'))
    ELSE
      RESET (datei, CONCAT (name,'.TEXT'));

    existiert := (IORESULT = Ø);
END;
{$I+}
{*****************************************************}
PROCEDURE ausdrucken;

VAR zeile, zw              : STRING [255];
    zeilen_zaehler         : INTEGER;
    usa, brd               : STRING;
    opt, neue_seite, break : BOOLEAN;
    kommentar              : INTEGER;
    k_ein, k_aus           : ARRAY [1..2] OF INTEGER;
{-------------------------------------------------}
  PROCEDURE ergaenze (wieviel : INTEGER; land : STRING);

  BEGIN
    IF land = '' THEN
      BEGIN
        zw := CONCAT (zw,zeile);
        zeile := '';
      END
    ELSE
      BEGIN
        zw := CONCAT (zw, COPY (zeile,1,wieviel),land);
        DELETE (zeile, 1, wieviel);
      END;
  END;
{-------------------------------------------------}
  PROCEDURE keine_leerzeilen_drucken;

  VAR leer : STRING [255];

  BEGIN
    leer := '                                                  ';
    leer := CONCAT (leer,leer,leer,leer,leer,leer,'    ');
    DELETE (leer,1,255 - LENGTH (zeile));
    IF zeile <> leer THEN
      neue_seite := FALSE;
  END;
{-------------------------------------------------}
  PROCEDURE transform;
```

```
  VAR i           : INTEGER;
      deutsch     : BOOLEAN;

  BEGIN
    IF kommentar <> Ø THEN
      zw := brd
    ELSE
      zw := usa;
    deutsch := FALSE;

    REPEAT
      i := (POS ('''', zeile) + 256) MOD 257;
      k_ein [1] := (POS ('(*', zeile) + 256) MOD 257;
      k_aus [1] := POS ('*)', zeile);
      k_ein [2] := (POS ('ä',zeile) + 256) MOD 257;
      k_aus [2] := POS ('ü',zeile);

      IF kommentar <> Ø THEN
        IF k_aus [kommentar] = Ø THEN
          ergaenze (Ø,'')
        ELSE
          BEGIN
            ergaenze (k_aus [kommentar] - 1, usa);
            kommentar := Ø;
          END
      ELSE
        IF deutsch THEN
          BEGIN
            ergaenze ((i+1) MOD 257, usa);
            deutsch := FALSE;
          END
        ELSE
          IF (i < k_ein [1]) AND (i < k_ein [2]) THEN
            BEGIN
              deutsch := TRUE;
              ergaenze (i+1,brd);
            END
          ELSE
            IF (k_ein [1] < k_ein [2]) THEN
              BEGIN
                kommentar := 1;
                ergaenze (k_ein [1] + 1, brd);
              END
            ELSE
              IF k_ein [2] < 256 THEN
                BEGIN
                  kommentar := 2;
                  ergaenze (k_ein [2] + 1, brd);
                END
              ELSE
                ergaenze (Ø,'');
    UNTIL zeile = '';

    zeile := zw;
  END;
{-------------------------------------------------}
  PROCEDURE vorbereiten;

  BEGIN
    zeilen_zaehler := Ø;
    kommentar := Ø;

    loesche;
    WRITE ('Drucker einschalten und <RETURN> drücken ');
    READ (KEYBOARD, antwort);
    IF NOT EOLN (KEYBOARD) THEN
      EXIT (ausdrucken); {ohne RETURN kein Ausdruck}

    loesche;
    WRITE ('Titelzeile eingeben    (nur <RETURN>');
    WRITELN (' = keine Titelzeile)',an);
    zeile := '';
    str_eingabe (zeile,ok);
    WRITE (aus);

    IF NOT ok THEN
      EXIT (ausdrucken);

    loesche;
    WRITE ('D(eutscher), A(merikanischer) oder ');
    WRITE ('O(ptimaler) Zeichensatz ?');

    REPEAT
      READ (KEYBOARD, antwort);
    UNTIL antwort IN ['d','D','a','A','o','O',esc];

    IF antwort = esc THEN
      EXIT (ausdrucken);

    UNITCLEAR (6);
    REWRITE (drucker, 'printer:');
```

```
    WRITE (drucker,esc,'c',esc,'e',esc,'LØ1Ø');
        {Drucker in Grundstellung bringen,
         Schriftart einstellen und linken Rand setzen.
         Die Schriftart muß so gewaehlt werden,
         daß nach Abzug des Randes noch mindestens
         8Ø Zeichen pro Zeile gedruckt werden koennen}

    usa := '.Z..';
    brd := '.Z...D..';
    usa [1] := esc;
    usa [3] := CHR (7);
    usa [4] := CHR (Ø);
    brd [1] := esc;
    brd [3] := CHR (3);
    brd [4] := CHR (Ø);
    brd [5] := esc;
    brd [7] := CHR (4);
    brd [8] := CHR (Ø);

    opt := FALSE;
    CASE antwort OF
      'a', 'A' : WRITE (drucker, usa);
      'd', 'D' : WRITE (drucker, brd);
      'o', 'O' : opt := TRUE;
    END;

    gotoxy (Ø,21);
    WRITE (clz,'F(ettdruck oder N(ormaldruck ?');

    REPEAT
      READ (KEYBOARD, antwort);
    UNTIL antwort IN ['n','N','f','F',esc];

    IF antwort IN ['f','F'] THEN
      WRITE (drucker,esc,'!');
      {Steuerzeichen fuer Fettdruck}

    JF antwort = esc THEN
      EXIT (ausdrucken);

    PAGE (OUTPUT);
    WRITELN (inv,' Jede Taste = Unterbrechung ',norm);


    IF zeile <> '' THEN
      BEGIN
        WRITELN (drucker,zeile);
        WRITE (drucker,'_____________________________');
        WRITE (drucker,'_____________________________');
        WRITELN (drucker,'_____________________________',lf);
        zeilen_zaehler := 3;
      END;
  END;
{-------------------------------------------------------}
  PROCEDURE wechseln;

  VAR start, laenge : INTEGER;
      fehler        : STRING;
{.......................................................}
    PROCEDURE error (nr : INTEGER);

    BEGIN
      WRITELN (lf,lf,bell);
      WRITE (inv);

      CASE nr of
        1 : BEGIN
              WRITE (' Geschachtelte Includedateien ');
              WRITELN ('sind nicht erlaubt ! ',norm);
            END;
        2 : BEGIN
              incl := FALSE;
              WRITELN (i_name,norm,' existiert nicht !');
            END;
      END;

      WRITE (lf,'<RETURN> = ignorieren   ...   ');
      WRITELN ('<ESC> = Abbruch',lf,lf);

      REPEAT
        READ (KEYBOARD,antwort);
      UNTIL antwort IN [' ',esc];

      IF antwort = esc THEN
        EXIT (ausdrucken);
    END;
{.......................................................}
  BEGIN     {PROCEDURE wechseln}
    IF (POS ('(*$I+',zeile) = Ø)
       AND (POS ('(*$I-',zeile) = Ø)
       AND (POS ('''',zeile) = Ø)
    THEN
      IF incl THEN
        error (1)
      ELSE
        BEGIN
          incl := TRUE;
          start := POS ('(*$I',zeile) + 4;
          laenge := POS ('*)',zeile) - start;
          i_name := COPY (zeile, start, laenge);

          WHILE POS (' ',i_name) <> Ø DO
            DELETE (i_name, POS (' ',i_name), 1);

          IF NOT existiert (i_name) THEN
            error (2)
          ELSE
            BEGIN
              READLN (includedatei, zeile);
              WRITELN (zeile);
            END;
        END;
  END;
{-------------------------------------------------------}
  FUNCTION unterbrechung : BOOLEAN;

  BEGIN
    READ (KEYBOARD, antwort);
    WRITELN;
    WRITE (inv,' <ESC> = Abbruch   ...   jede andere ');
    WRITELN ('Taste = Ausdruck fortsetzen ',norm,lf);
    UNITCLEAR (1);
    READ (KEYBOARD, antwort);

    IF antwort = esc THEN
      unterbrechung := TRUE
    ELSE
      unterbrechung := FALSE;
  END;
{-------------------------------------------------------}
BEGIN       {PROCEDURE ausdrucken}
  vorbereiten;

  REPEAT
    IF incl THEN
      READLN (includedatei, zeile)
    ELSE
      READLN (datei, zeile);
    WRITELN (zeile);

    IF POS ('(*$I',zeile) <> Ø THEN
      wechseln;

    IF POS ('(**)',zeile) <> Ø THEN
      BEGIN
        zw := zeile;
        WHILE POS (' ',zw) <> Ø DO
          DELETE (zw, POS (' ',zw), 1);

        IF zw = '(**)' THEN
          BEGIN           {Seitenvorschub-Befehl}
            zeilen_zaehler := Ø;
            IF NOT neue_seite THEN
              PAGE (drucker);
            neue_seite := TRUE;
            zeile := '';
          END;
      END;

    IF neue_seite THEN
      keine_leerzeilen_drucken;

    IF NOT neue_seite THEN
      BEGIN
        IF opt THEN
          transform;

        WRITELN (drucker, zeile);
        zeilen_zaehler := SUCC (zeilen_zaehler);
        {Seitenvorschub}
        IF zeilen_zaehler MOD zps = Ø THEN
          BEGIN
            neue_seite := TRUE;
            PAGE (drucker);
          END;
      END;

    IF incl THEN
      IF EOF (includedatei) THEN
        BEGIN
          CLOSE (includedatei);
```

```
          incl := FALSE;
        END;

    break := FALSE;
    IF keypress THEN
      break := unterbrechung;

  UNTIL EOF (datei) OR break;

  WRITE (drucker,brd);      {deutscher Zeichensatz}

  IF NOT (break OR neue_seite) THEN
    PAGE (drucker);
END;
{*****************************************************}
BEGIN       {Programm}
  esc := CHR (27);        {Es folgen Steuerzeichen}
  clz := CHR (26);        {Zeile loeschen}
  inv := CHR (15);        {Inv-Modus einschalten}
  norm := CHR (14);       {Inv-Modus ausschalten}
  aus := CHR (6);         {Cursor unsichtbar}
  an := CHR (5);          {Cursor sichtbar}
  bell := CHR (7);        {Signalton}
  lf := CHR (10);         {Linefeed}
  bs := CHR (8);          {Backspace}

  titel_text;

  WHILE TRUE DO
    BEGIN
      IF mit_titel THEN
        titel_text;

      REPEAT
        loesche;
        cat := FALSE;
        WRITE ('Wie heißt die Textdatei ?     <ESC> =');
        WRITELN (' Abbruch   <RETURN> = Catalog ', an);
        UNITCLEAR (1);
        str_eingabe (d_name,ok);
        WRITE (aus);
        IF NOT ok THEN          {ESC = Abbruch}
          ende;

        IF d_name = '' THEN
          BEGIN    {RETURN = Katalog zeigen}
            cat := TRUE;
            loesche;
            WRITE ('Laufwerk ? #',an);
            {$I-}
            READLN (laufwerk);
            {$I+}
            WRITE (aus);
            IF laufwerk IN [4,5,9..12] THEN
              catalog (laufwerk);
            linien;
          END;
      UNTIL NOT cat;

      incl := FALSE;
      ok := TRUE;
      IF NOT existiert (d_name) THEN
        BEGIN
          loesche;
          ok := FALSE;
          WRITE (inv,' ',d_name,' ist nicht ');
          WRITE ('vorhanden !',norm,bell);
          FOR pause := 1 TO 4000 DO;
        END;

      IF ok THEN
        BEGIN
          mit_titel := TRUE;
          ausdrucken;
          CLOSE (datei);
          CLOSE (drucker);
          IF incl THEN
            CLOSE (includedatei);
        END;
    END;
END.
```

Gerüchteküche

Zum IIgs

In den USA haben externe Soft- und Hardware-Entwickler nach dem 15. September 1986 ein ganzes Bündel von Apple-IIgs-Produkten angekündigt, die im Laufe der nächsten Wochen und Monate erscheinen werden. Eine kleine Auswahl:

- Kyan Software wird im Dezember ein neues „Kyan-Pascal" (65818-Pascal-Compiler) vorstellen.
- Von TML Systems soll ebenfalls ein „Pascal-Compiler" erscheinen.
- Roger Wagner Publishing wird den Assembler „Merlin 16" (65816-Merlin/Big Mag), ein erweitertes „Mousewrite" (Textverarbeitungsprogramm; vgl. Peeker 7/86) sowie ein Pseudo-Coprocessor-Programm „Softswitch" zum Umschalten zwischen mehreren RAM-residenten 8-Bit-Programmen (ähnlich wie unser „Softbreaker") herausbringen.
- Von Applied Engineering werden mehrere RAM-Erweiterungskarten sowie
  - man höre und staune – eine Accelerator mit 6 MHz erscheinen.
- Broderbund Software wird sein beliebtes „Printshop" in einer Neufassung herausbringen.
- Von VIP Technology wird das bereits für den Atari ST vorliegende integrierte Software-Paket „VIP Professional" (Tabellenkalkulation, Dateiverwaltung, Business-Grafik) veröffentlicht.

Es ist geplant, die interessanteren IIgs-Produkte zu sehr günstigen Preisen den Peeker-Lesern über den Hüthig Software Service anzubieten.

Zum Macintosh

Der neue „Macintosh" soll möglicherweise schon Ende Januar vorgestellt werden, und zwar mit 68020-Prozessor mit 16 MHz, größerem RAM-Speicher, größerem Bildschirm und Slots. Von Atari wird der analoge „Atari TT" (32-Bit-Prozessor 68020: „Thirtytwo-Thirtytwo" gegenüber „ST": 16/32-Bit-Prozessor 68000: „Sixteen-Thirtytwo") vermutlich erst im Herbst 1987 erscheinen. Als Kuriosum am Rande: Für den Atari 1040 ST gibt es seit kurzem einen Macintosh-Emulator für weniger als DM 400,–. Allerdings muß man die Mac-ROMs besitzen.

Zum Atari

Von Atari gibt es seit Mitte Oktober für die ST-Reihe einen „Bit-Blitter": Dies ist ein Grafikspeicher-„Blockverschiebungsbaustein", der in Verbindung mit dem geänderten Betriebssystem-ROM die gesamte Grafik-Bildschirmausgabe um den Faktor 5 beschleunigt. Ferner wird ein „MS-DOS-Emulator" erscheinen: Dies ist eine 8088-Platine mit 512K-RAM, die an den DMA-Port (im Daisy-Chain-Verfahren) angeschlossen wird. MS-DOS-Programme können dann auf den Atari-3,5-Zoll-Drives gefahren werden.

INPUT für Formeln

Eine Utility für Applesoft-BASIC

von Hans-Martin Eng

Ausgangslage

Geben Sie zu, am liebsten hätten Sie schon gerne einmal auf die Frage des Apple nach einer Eingabe einfach mit „1/7“ statt mit „0.142857143“ geantwortet. Oder vielleicht sind Sie es leid, statt einfach die dem Applesoftprogramm längst bekannte Variable PI mit ihrem Namen aufzurufen, brav „3.1415926...“ in die Tastatur zu hacken. Nun, für dieses Problem gibt es eine überraschend einfache und elegante Lösung. Ersetzen Sie alle INPUT-Statements durch Befehle der Form XX = USR(0). Dann starten Sie vom Programm aus das unten aufgelistete Maschinenprogramm „INPUT.FORM“ (Adresse ist unwichtig, das Programm ist relokativ geschrieben und setzt den USR-Vektor automatisch richtig) und Sie können Ihrem Programm ganz nach Belieben Formeln zu futtern geben. Wie funktioniert es?

Der USR-Befehl

Wie Sie vielleicht wissen, funktioniert der USR-Befehl ähnlich wie der &-Befehl. Im Unterschied zu diesem wird jedoch der FAC1 (Floating Point Accumulator 1) mit dem Wert des Arguments geladen, erst dann erfolgt der Sprung an die vom USR-Vektor ($0A..$0C) angegebene Adresse. Ist die aufgerufene Routine abgearbeitet, wird der jetzige Inhalt des FAC1 als Wert der USR-Funktion übernommen.

Das Programm

Die unten aufgelistete Routine rettet zuerst die Registerinhalte und den CHRGET-Pointer auf den Stack. Dann wird die ROM-Routine INLINE2 aufgerufen, die normalerweise BASIC-Programmzeilen einliest. Nun wird zuerst überprüft, ob nicht eine leere Eingabe erfolgte. Ist dies nicht der Fall, wird der String, der sich immer noch im INPUT-Puffer befindet, durch die PARSE-Routine geschickt. Diese Routine konvertiert die BASIC-Keywords in die zugehörigen Tokens. Nun wird der CHRGET-Pointer auf den Anfang der nunmehr codierten Eingabezeile gesetzt und die ROM-Routine FRMEVL (= Formula Evaluator) aufgerufen. Diese Routine klassifiziert einen Ausdruck (String- oder numerischer Ausdruck) und wertet ihn aus. Das Ergebnis befindet sich dann im FAC1 (wenn es ein numerischer Ausdruck war).

Haben Sie es gemerkt? Wir sind schon beinahe fertig. Nun muß nur noch der CHRGET-Pointer auf seinen alten Wert gebracht werden, die geretteten Register müssen zurückgeholt werden, und der Akkumulator sollte noch das nächste Zeichen des Programmtextes enthalten. Letzteres erledigt die CHRGOT-Routine, so daß man sich auch noch ein abschließendes RTS sparen kann. Der restliche Code besteht zum einen aus der INIT-Routine, die nach einem Start mit „BRUN“ den USR-Vektor setzt, und zum anderen aus den Abschnitten zur Fehlerbehandlung. Das alles ist nur 95 Bytes lang, also wirklich nicht allzu umfangreich.

DEMO.INPUT.FORM

(Applesoft-BASIC-Listing)

```
10 REM Demoprogramm zu INPUT.FORM
20 REM z.B. 'COS(4*ATN(1))' eingeben (= -1)
30 PRINT "Bitte numerischen Ausdruck eingeben";
40 X = USR (0)
50 PRINT "Wert: ";X
60 END
70 REM Niemand verbietet Ihnen, den
80 REM USR-Ausdruck zu verschachteln
90 X = USR (0) * SIN (2 * USR (0) - 1)
95 REM Auch das ist erlaubt!
```

T.INPUT.FORM

BSAVE INPUT.FORM, A768, L95

```
1    ********************************
2    *                              *
3    * INPUT-Routine für numerische *
4    * Ausdrücke:                   *
5    *                              *
6    * Syntax:  X = USR (0)         *
7    *                              *
8    * Von Hans-Martin Eng,         *
9    * Hirschberg, 10. 8. 1986      *
10   *                              *
11   ********************************
12   *
13            ORG  $300
14   *
15   USRVEKT  EQU  $0B
16   CHRGOT   EQU  $B7
17   CHRPNT   EQU  $B8
18   ERRFLAG  EQU  $D8
19   INBUF    EQU  $200
```

```
                    20   INLIN2   EQU  $D52E
                    21   PARSE    EQU  $D56C
                    22   STROUT   EQU  $DB3A
                    23   FRMEVL   EQU  $DD7B
                    24   HANDLERR EQU  $F2E9
                    25   FIXRTS   EQU  $FF58
                    26   *
                    27   * Sieht kompliziert aus, nicht?
                    28   * So ist das Programm aber voll
                    29   * relokativ!
                    30   *
0300: 48            31   INIT     PHA
0301: 8A            32            TXA
0302: 48            33            PHA
0303: 20 58 FF      34            JSR  FIXRTS
0306: BA            35            TSX
0307: 18            36            CLC
0308: BD FF 00      37            LDA: $FF,X      ; Absicht!
                    38   *           ↑ erzwingt 2-Byte-Adresse
030B: 69 15         39            ADC  #ENTRY-INIT-5
030D: 85 0B         40            STA  USRVEKT
030F: BD 00 01      41            LDA  $100,X
0312: 69 00         42            ADC  #0
0314: 85 0C         43            STA  USRVEKT+1
0316: 68            44            PLA
0317: AA            45            TAX
0318: 68            46            PLA
0319: 60            47            RTS
                    48   * Rette Register und CHRGET-Pointer
031A: 8A            49   ENTRY    TXA
031B: 48            50            PHA
031C: 98            51            TYA
031D: 48            52            PHA
031E: A5 B8         53            LDA  CHRPNT
0320: 48            54            PHA
0321: A5 B9         55            LDA  CHRPNT+1
0323: 48            56            PHA
                    57   * und los geht's!
                    58   * '#' als Prompt-Character
0324: A2 A3         59   FORMGET  LDX  #"#"
0326: 20 2E D5      60            JSR  INLIN2
0329: AD 00 02      61            LDA  INBUF
                    62   * keine Eingabe => Fehler
032C: F0 12         63            BEQ  ERROR
                    64   * Keywords codieren
032E: 20 6C D5      65            JSR  PARSE
                    66   * CHRGET-Pointer umlenken
0331: A9 FD         67            LDA  #<INBUF-3
0333: 85 B8         68            STA  CHRPNT
0335: A9 01         69            LDA  #>INBUF-3
0337: 85 B9         70            STA  CHRPNT+1
                    71   * numerischen Ausdruck auswerten
0339: 20 7B DD      72            JSR  FRMEVL
                    73   * Stringausdruck?
033C: 24 11         74            BIT  $11
                    75   * nein, o.k.
033E: 10 12         76            BPL  OKAY
                    77   * ONERR-Flag gesetzt?
0340: 24 D8         78   ERROR    BIT  ERRFLAG
0342: 10 05         79            BPL  ERROR1
                    80   * wenn ja, übergib an ONERR-Routine
0344: A2 FE         81            LDX  #$FE
0346: 4C E9 F2      82            JMP  HANDLERR
                    83   * sonst gib '?REENTER' aus
0349: A9 EF         84   ERROR1   LDA  #$EF
034B: A0 DC         85            LDY  #$DC
034D: 20 3A DB      86            JSR  STROUT
                    87   * und mach's noch einmal.
0350: F0 D2         88            BEQ  FORMGET
                    89   * Pointer und Register zurückholen
0352: 68            90   OKAY     PLA
0353: 85 B9         91            STA  CHRPNT+1
0355: 68            92            PLA
0356: 85 B8         93            STA  CHRPNT
0358: 68            94            PLA
0359: A8            95            TAY
035A: 68            96            PLA
035B: AA            97            TAX
                    98   * Fertig, weiter im Programmtext
035C: 4C B7 00      99            JMP  CHRGOT
```

CP/M 3.0 beim Apple II

Was hat sich geändert?

von Dr. H. Kersten und Dr. H.-J. Ganser

Seit einiger Zeit sind für den Apple II Zusatzkarten erhältlich, die einerseits einen schnellen Z80H-Prozessor (6/8 MHz) und andererseits 64K RAM on board haben. Insbesondere der letzten Eigenschaft wegen ist es möglich, eine stark erweiterte (und verbesserte) CP/M-Version zu benutzen – eben CP/M 3.0. Hier soll untersucht werden, worin sich das neue CP/M von dem gängigen CP/M 2.2x unterscheidet.

Um das CP/M 3.0 lauffähig zu machen, ist – wie bereits gesagt – eine geeignete Zusatzkarte erforderlich. Es gibt sie als reine CP/M-3.0-Karte (z.B. vom ALS-Typ) – hiermit kann kein „normales" CP/M 2.2x gefahren werden – oder als Z80+-Karte (vgl. Testbericht im Peeker 6/86), auf der beide CP/M-Versionen lauffähig sind. Natürlich braucht man auch die entsprechenden Disketten mit dem CP/M-3.0-System.
Der Anwender wird zunächst nach dem Boot-Vorgang seine „alten" CP/M-Kenntnisse ausnutzen und ein gehaltvolles „DIR"-Kommando eingeben, danach noch eine „TYPE"-Anweisung ausprobieren und vielleicht schon das Gefühl haben, das neue CP/M 3.0 im Griff zu haben: Weit gefehlt.
Schlägt man nämlich im CP/M+ User's Guide unter dem Stichwort „DIR" nach, so findet man überraschenderweise 9 Seiten Erläuterungen zu diesem Kommando und all seinen Optionen. Selbst bei dem scheinbar wenig ausbaufähigen „TYPE" und anderen Standard-Kommandos sind viele neue Details vorhanden.
Nach diesem Stimmungsbericht erübrigt es sich schon fast zu sagen, daß eine Reihe neuer (transienter) Kommandos hinzugefügt worden ist. Es führt kein Weg daran vorbei: Zumindest den User's Guide muß man sich genauer ansehen.

Es gibt weiterhin noch den Programmer's Guide (u.a. Beschreibung der BDOS-Schnittstellen) und einen System Guide (BIOS-Anpassungen und anderes). Die Anzahl der BDOS- und BIOS-Funktionen hat sich gegenüber der Betriebssystem-Version 2.2 erheblich vergrößert – einige Informationen hierzu in einem späteren Abschnitt.

Ein kurzer Blick über den Zaun: Wer – wie die Autoren – auch mit PC-DOS und MS-DOS beim blauen 16-Bitter arbeitet, stellt fest, daß beim CP/M 3.0 in vielen Details Vergleichbares vorliegt: Umleiten von Standard-Ein- und -Ausgabe, zusätzlich ladbare System Extensions, wahlweise Start von Batch- oder Command-Files, Grundformen eines Suchpfades, Zeit- und Datum-Stempel bei Files etc.
In anderer Hinsicht bleibt natürlich alles beim alten: Max. 64K adressierbarer Speicher (der aber weitestgehend dem Benutzer zur Verfügung steht, da das Betriebssystem fast vollständig in der 2. Bank untergebracht ist), die typischen CP/M-User-bereiche statt der schönen Subdirectories (wie etwa auch bei PRODOS).
Wie schon im Peeker-Testbericht 6/86 erwähnt, ist die Verarbeitungsgeschwindigkeit beim Z80H-Prozessor mit der hohen Taktfrequenz beachtlich und braucht keinen Vergleich zu scheuen.

Erwähnenswert ist noch, daß manche Händler schon Patches für den Betrieb von doppelseitigen Laufwerken bis 160 Tracks mitliefern. Die Verwendung von Laufwerken mit größerer Kapazität muß im Zusammenhang mit CP/M 3.0 dringend empfohlen werden, da die System-Software schon ca. 330K umfaßt. Selbst wenn das Assembler-Paket abgerechnet wird, bleiben noch 250K für die transienten System-Kommandos. Das erweiterte DIR-Kommando benötigt alleine schon 16K.
Sind Sie noch interessiert? Dann gehen wir zu Einzelheiten über.

1. Editieren von Kommandozeilen, Mehrfach-Kommandos, On-Line-Hilfe

Keine Ausflüchte: Mit dem Editieren von Eingaben ist es beim alten CP/M 2.2 schlecht bestellt. Das Einfügen von Zeichen in eine fehlerhafte Kommando- oder Datenzeile ist praktisch nicht möglich. Im Normalfall muß die Zeile komplett oder zumindest teilweise neu eingetippt werden.
Beim CP/M 3.0 hat sich die Lage erheblich verbessert, wie man an folgender Tabelle erkennt. Von CP/M 2.2x bekannte Codes sind nicht aufgeführt:

```
CTRL-A: Cursor <- (ohne Löschen)
CTRL-F: Cursor -> (ohne Löschen)
CTRL-B: Cursor Zeilenanfang/-ende
CTRL-G: Zeichen löschen (Wordstar)
CTRL-X/CTRL-K: alle Zeichen links/rechts
        vom Cursor löschen
CTRL-U: Zeile komplett löschen, in
        Editierpuffer übernehmen
CTRL-W: Zeile aus Editierpuffer auf
        Bildschirm bringen
```

Wichtig: Bei allen Editierfunktionen ist der automatische Einfügemodus eingeschaltet. Diese Option läßt sich leider nicht abschalten. Dennoch kann man hiermit und mit den teilweise unüblichen Editiercodes leben.
Eine schöne Neuerung: Mehrere Kommandos auf einer Zeile sind möglich. Beispiel:

TYPE TEST.DAT ! TYPE TEST1.DAT ! DIR

Das Zeichen „!“ trennt die einzelnen Befehle voneinander.
In Anbetracht der schon kurz erwähnten Neuerungen bei vielen Kommandos könnte der Eindruck entstehen, daß nun ermüdendes Blättern in den Handbüchern angesagt ist: Auch hier gibt es Positives zu berichten. Mit dem HELP-Kommando kann man sich eine Kommandoliste ausgeben lassen oder sofort Syntax und Erläuterungen einzelner Kommandos abrufen.

2. Umleiten der Standard-Ein- und -Ausgabe

Mit den neuen Anweisungen PUT und GET kann man die Datenströme umleiten – unter CP/M 2.2 war dies nur mit Zusatzprogrammen (z.B. MicroShell oder Unix-Tools) möglich. Beispiel:

```
PUT CONSOLE OUTPUT TO FILE TEST.TXT
```

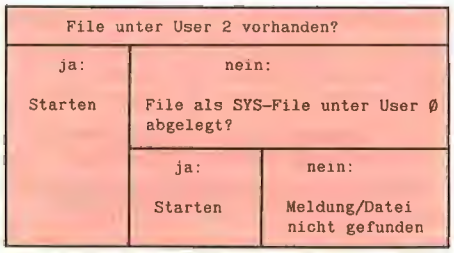
Hiermit wird die Console-Ausgabe des nächsten Kommandos oder des nächsten Programms in die Datei TEST.TXT umgeleitet. Fügt man bei der Zeile noch die Option [SYSTEM] an, so bleibt die Umleitung bis zur nächsten PUT-Anweisung bestehen.
Besonders schön ist die Umleitungsmöglichkeit im Zusammenhang mit dem DIR-Kommando (Diskettenverzeichnis als Datenfile) oder für das Verketten von Programmen: Die Ausgabe des 1. Programms ist die Eingabe des 2. Programms. Für die letzte Anwendung muß dann noch vor dem Start des zweiten Programms die Eingabe umgeschaltet werden. Diese generelle Möglichkeit wird wie folgt angewählt:

```
GET CONSOLE INPUT FROM FILE TEST.TXT
```

Danach wird das 2. Programm gestartet und entnimmt seine Eingabedaten, die sonst von der Console kommen, dem angegebenen File. Auf diese Weise kann man eine (über einen Zwischenfile simulierte) Pipeline aufbauen. Auch bei GET ist die [SYSTEM]-Option erlaubt. Statt „CONSOLE OUTPUT“ ist auch „PRINTER OUTPUT“ möglich, statt Umleitung über Files ist bspw. „PUT CONSOLE OUTPUT TO PRINTER“ statthaft.
Am Ende dieser GET- und PUT-Anweisungen sind noch weitere Optionen anfügbar: z.B. mit oder ohne Echo auf dem Bildschirm, Filtern von nicht druckbaren Zeichen.
Mit der erwähnten SYSTEM-Option kann man offensichtlich Ähnliches erreichen wie früher mit SUBMIT und XSUB – abgesehen von veränderlichen Parametern. Das SUBMIT-Kommando existiert beim CP/M 3.0 natürlich auch, muß aber nur dann angewendet werden, wenn wirklich Parameter auszutauschen sind. Alle anderen Fälle erledigt man durch Umleitung.

3. Userbereiche, File-Search und Diskettenwechsel

Eine wichtige Verbesserung: Man sieht stets am Prompt, in welchem Userbereich man sich befindet. Z.B. wird (bei Laufwerk A:) nach USER 2 mit 2A> geantwortet. Statt USER 2<Return> kann der Benutzer auch die Kurzform 2A:<Return> eingeben.
Erfahrungsgemäß wird beim CP/M 2.2 selten mit mehreren Userbereichen gearbeitet – insbesondere deshalb, weil dann dauernd benötigte Utilities in mehreren Userbereichen vorhanden sein müssen oder ihr Start ein Umschalten der Usernummer voraussetzt.
Dies ist bei CP/M 3.0 ganz unproblematisch, da hier eine Suchautomatik eingebaut ist: Wird z.B. von Usernummer 2 in Laufwerk B: ein Programm – ein transientes Kommando – gestartet, sucht CP/M 3.0 nach folgendem Schema:



Das gleiche Schema gilt, wenn von irgendeinem Programm aus ein Datenfile verlangt wird.
Das alles ist zwar nützlich, aber noch nicht aufregend. Spannend wird es jetzt: Mit dem SETDEF-Kommando kann für den Aufruf von transienten Kommandos ein Suchpfad definiert werden, auf dem die gleichen Userbereiche bei mehreren Laufwerken abgesucht werden. Ist z.B. irgendwann vorher

```
SETDEF *,A:,C:
```

eingegeben worden, so wird zunächst das aktuelle Laufwerk (angedeutet durch *, im Beispiel oben B:) in der Reihenfolge User 2 / User 0 durchsucht, danach Laufwerk A: User 2 / User 0, schließlich Laufwerk C: User 2 / User 0. Bis zu 16 Laufwerke können in den Suchpfad eingebaut werden.
Alles in allem ist SETDEF ein Feature, an das man sich sehr schnell gewöhnen kann.
Genauso steht es mit dem folgenden: CP/M 3.0 erkennt wie sein Vorläufer, ob in einem Laufwerk die Diskette gewechselt wurde. Statt aber dieses Laufwerk als „Read Only“ zu klassifizieren (wie bei CP/M 2.2), wird die Diskette neu eingelogt und ist ohne weiteres beschreibbar. Die Eingabe von CTRL-C ist überflüssig. Nachteil dieser Methode: Auch beim Nur-Lesen einer neuen Diskette wird das Log-In ausgeführt, was z.B. bei einem simplen DIR-Kommando und einer vollen Diskette die Geduld strapaziert.
Die Eingabe von CTRL-C hat hier eine etwas andere Funktion: Hätte man eine Diskette mit dem SET-Kommando als „Read Only“ deklariert, könnte man dies durch CTRL-C wieder aufheben. Wird der Read-Only-Status explizit gesetzt, geht CP/M 3.0 natürlich nicht einfach durch ein automatisches Log-In darüber hinweg.

4. Volumes, Dateien, Attribute, Zeitstempel und Passwords

Jetzt geht es in die vollen: CP/M 3.0 ist u.a. in der Lage, Dateien mit Datum- und Uhrzeit-Eintragungen zu versehen, Disketten einen Volumenamen zuzuordnen und Disketten oder einzelne Dateien mit Passwords zu schützen.
Zunächst zu den Volume/Disk-Labels. Mit SET B: [NAME=BACKUP.A17] erhält die Diskette in Laufwerk B: den Namen BACKUP.A17. Dieser Name wird im Directory gespeichert. Für Volumenamen gilt die gleiche Bildungsregel wie für Dateinamen.
Nun zu Passwords bei Disketten. Mit SET [PASSWORD=XYZ] erhält die Diskette im aktuellen Laufwerk ein Password (hier XYZ, max. 8 Zeichen), d.h. alle weiteren Kommandos, die das Volume betreffen (z.B. Ändern des Labels, Ändern der Struktur der Zeitstempel u.a.), können nur nach Eingabe des korrekten Passwords ausgeführt werden.
Um einzelne Files mit Passwords zu versehen, muß erstens die Diskette ein Password besitzen, zweitens die Anweisung SET [PROTECT=ON] gegeben werden. Danach kann man z.B. mit SET *.COM [PASSWORD=XYZDF] alle COM-Files mit dem Password XYZDF belegen. Die Password-Sperre kann man auf bestimmte Zugangsformen einschränken. Mit SET *.COM [PASSWORD=XYZDF,PROTECT=WRITE] wird die Sperre auf den Zugangstyp „WRITE“ beschränkt. Der folgenden Tabelle entsprechend bedeutet dies, daß das Password vor einem Schreibvorgang, dem Löschen oder Umbenennen der Datei angegeben werden muß.

Zugangsformen bei Password-Sperre:

```
READ    Lesen,Schreiben,Löschen,Umbenennen
WRITE          Schreiben,Löschen,Umbenennen
DELETE                   Löschen,Umbenennen
```

Um Password-geschützte Dateien benutzen zu können, kann entweder mit SET ein Default-Password für eine ganze Gruppe von Files (z.B. die COM-Files im Beispiel) oder bei jedem Fileaufruf das betreffende Password explizit angegeben werden. Beispiel:

```
SET [DEFAULT=XYZDF]     Default Password
PIP B:= A:*.COM
bzw.
PIP B:= A:*.COM;XYZDF   explizites Password
```

CP/M 3.0 betrachtet das Password als Bestandteil des Filenamens. Aus „Drive:Filename.Extension" wird jetzt „Drive:Filename.Extension;Password". Schließen Sie aber daraus nicht, daß das Password im Inhaltsverzeichnis in „Klarschrift" abgelegt wird. Das soll zum Thema Passwords genügen.
Neben den bekannten Fileattributen DIR, SYS und RO, RW kennt CP/M 3.0 noch ARCHIVE=ON und ARCHIVE=OFF, sowie 4 vom Benutzer selbst definierbare Attribute F1 – F4. Das ARCHIVE-Attribut ist im Zusammenhang mit PIP wichtig. Ist ARCHIVE=ON, so setzt PIP nach jedem Kopiervorgang ein entsprechendes Flag im Directory und zeigt damit an, daß die betreffende Datei gesichert wurde. Bei der ersten Änderung dieser Datei wird vom System das betreffende Flag zurückgesetzt: PIP kopiert beim nächsten BACKUP-Vorgang diese Datei erneut, verzichtet jedoch im anderen Fall richtigerweise darauf, da ja die alte (bereits gesicherte) Fileversion immer noch aktuell ist. Vorteil: Eine Verkürzung der Backup-Zeiten. Die Attribute F1 – F4 kann der Benutzer für eigene Markierungszwecke verwenden.

Nun noch zu Zeit- und Datum-Stempel. Mit der DATE-Anweisung (transientes Kommando) kann die Systemzeit gesetzt oder abgefragt werden (setzt keine Hardware-Uhr voraus). CP/M 3.0 protokolliert bei allen Dateien wahlweise entweder Datum und Uhrzeit der Dateierstellung (CREATE) oder des letzten Zugriffs (ACCESS). Zusätzlich kann die letzte Dateiänderung eingetragen werden (UPDATE). Mit dem viel strapazierten SET-Kommando kann die gewünschte Wahl getroffen werden. Mit SET [CREATE=ON,UPDATE=ON] entscheidet man sich für die Kombination CREATE/UPDATE.
Das alles setzt aber voraus, daß das Directory der Diskette zur Aufnahme von Zeitstempeln vorbereitet wurde. Es muß ja zusätzlich Platz reserviert werden. Hierfür existiert das INITDIR-Kommando. Statt wie bisher 4 Dateieinträge pro 128-Byte-Record werden nun nur noch 3 angelegt, die 4. Position bleibt frei für die Zeitstempel der drei vorausgehenden Einträge. Mit INITDIR kann ein noch leeres Directory oder ein bereits gefülltes ohne Fileverluste angepaßt werden – vorausgesetzt, die DIR-Sektoren auf der Diskette bieten noch genügend Platz. Keine Angst: Bei Platznot bleibt das alte Verzeichnis unangetastet!

5. SUBMIT, Batch-Processing und „Hello"-File

Das von CP/M 2.2 bekannte SUBMIT-Kommando hat hier eine schöne Neuerung erfahren: In einem SUB-File können sowohl Kommandos wie auch Datenzeilen enthalten sein. (Früher war hierzu XSUB erforderlich). Mehr noch: Es erfolgt eine automatische Umschaltung von SUB-File auf CONSOLE. Ein Beispiel ist sicher instruktiv: Nehmen wir an, in TEST.SUB stehen folgende Zeilen:

```
PROG77
<12
<23,ABCD,-9
DIR B:
DIR A:
```

Nach dem Kommando SUBMIT TEST wird zunächst PROG77.COM geladen. Vorausgesetzt, dieses Programm möchte Daten von der Tastatur lesen, so werden ihm die mit „<" beginnenden Zeilen übergeben, als wären sie auf der Tastatur eingegeben worden – das Zeichen „<" wird herausgefiltert, macht also die Zeilen nur als Datenzeilen kenntlich. Verlangt PROG77 nur 2 Eingabezeilen – wie im Beispiel – und endet anschließend, so werden von CP/M 3.0 die nachfolgenden DIR-Kommandos ausgeführt. Möchte PROG77 dagegen noch mehr Eingaben haben, so schaltet CP/M 3.0 automatisch auf die Tastatur um, da in der SUB-Datei nun Kommandos (DIR) folgen.
Würde andererseits PROG77 nur eine einzige Eingabezeile benötigen, würde das Betriebssystem die 2. Datenzeile übergehen. Eine feine Sache!
Im Beispiel sind keine variablen Parameter im SUB-File vorhanden: Sie werden wie beim alten CP/M angewendet; insofern also nichts Neues.
Jede SUB-Datei kann intern weitere SUBMIT-Kommandos enthalten: eine Schachtelung (mit Rückkehr) ist erlaubt. Beim CP/M 2.2 bedeutet ein neues SUBMIT stets das Ende des vorhergehenden.

Nach dem Boot-Prozeß versucht das Betriebssystem, einen bestimmten SUB-File zu finden: PROFILE.SUB. Diese Datei enthält die ersten auszuführenden Kommandos – wie STARTUP bei PRODOS oder HELLO bei DOS. Dies scheint eine etwas einfachere Prozedur zu sein, verglichen mit AUTORUN beim CP/M 2.2, das ja ebenfalls dem Zweck dient, ein Turnkey-System aufzubauen.

Ein letztes Feature, das wie einige andere sehr an das Konkurrenzprodukt von „Big Blue" erinnert: Beim alten CP/M können nur transiente Kommandos – also COM-Files – durch Angabe ihres Namens gestartet werden. Beim CP/M 3.0 ist das zunächst auch so. Mit dem SETDEF-Kommando kann man aber angeben, daß auch SUB-Dateien durch Eintippen des Namens „gestartet" werden. Zusätzlich legt SETDEF in einem Befehl fest, in welcher Reihenfolge bei Namensgleichheit vorgegangen wird. Beispiel: SETDEF [ORDER=(SUB,COM)]. Hierdurch können von jetzt ab auch SUB-Dateien wie erläutert gestartet werden. Wäre im obigen Beispiel neben der Datei TEST.SUB auch noch TEST.COM vorhanden, so würde das Betriebssystem entsprechend dem SETDEF-Kommando zuerst TEST.SUB bearbeiten. Erst wenn kein TEST.SUB zu finden ist, wäre TEST.COM an der Reihe. Die Vorgehensweise ist also die gleiche wie bei MS-DOS.

6. Sonstige transiente Kommandos und Utilities

Beim DIR-Kommando und einigen anderen Kommandos unterscheidet CP/M 3.0 zwischen „built-in" und „transient" Kommandos. Das vom alten CP/M bekannte DIR (mit Wildcards) ist Bestandteil des CCP (= Console Command Processor), wird also direkt ausgeführt, ohne etwas von der Diskette zu laden, und ist vom Typ „built-in". Mit den vielen Zusatzoptionen beim CP/M 3.0 ist das aber platzmäßig nicht mehr zu machen. Konsequenz: Wird eine Sonderoption vom Benutzer eingetippt, wird das „built-in DIR" durch ein „transient DIR" verstärkt (im DIR-Listing: DIR.COM). DIR kann beim CP/M 3.0 u.a.
– die Userattribute F1...F4 anzeigen
– den Datum-/Zeitstempel abbilden
– Verzeichnisse mehrerer Disks auf ein Mal ausgeben
– Filelängen, Password/Protection Modes anzeigen
– alle oder einzelne Userbereiche durchlaufen
– ausschließende Dateiangaben verwenden, z.B. bedeutet ‚DIR [EXCLUDE] *.COM' alle Files außer *.COM
– vorab Form-Feeds senden, Paging nach n Zeilen ausführen.
In der folgenden Abbildung **(Abb. 1)** ist ein Auszug aus dem Inhaltsverzeichnis einer typischen CP/M-3.0-Diskette abgedruckt. Das built-in DIR zeigt normalerweise nur Files mit dem Attribut „DIR" an, also keine „SYS"-Files. Genau umgekehrt liegt der Fall beim Kommando DIRS bzw. DIRSYS. (Anmerkung: Die Datei LOG.DAT wurde mit PUT CONSOLE OUTPUT TO FILE LOG.DAT und dem DIR-Kommando er-

... in eigener Sache:

die Nachfrage nach den vergriffenen Heften 1/84, 4/85, 5/85, 6/85, 8/85, 10/85 und 11/85 ist groß. Ab sofort können Sie Heftkopien direkt beim Verlag bestellen.

Preis je Heftkopie: DM 10,– plus Versandkosten.
Bitte Vorauszahlung leisten oder Verrechnungsscheck mitschicken.

»Peeker« ist eine aktuelle und zuverlässige Informationsquelle. Ein einziger Tip, den Sie der Zeitschrift entnehmen, kann viel mehr wert sein als die Kosten für ein Abonnement.

Inhaltsverzeichnis einer CP/M-3.0-Disk
(Auszug)

```
Directory For Drive A:  User  Ø

     Name      Bytes   Recs   Attributes   Prot      Update          Access
------------  ------  ------  ----------  ------  --------------  --------------

CPM3     SYS    16k     128 Sys RO       None
DATE     COM     4k      22 Dir RW       Write                   11/Ø3/85 ØØ:ØØ
DIR      COM    16k     126 Dir RW       Write                   Ø7/1Ø/86 18:57
GET      COM     8k      51 Dir RW       Write
HELP     COM     8k      56 Dir RW       Write
HELP     HLP    62k     488 Dir RW Arcv  Write   Ø7/1Ø/86 18:14  Ø7/1Ø/86 18:18
INITDIR  COM    32k     25Ø Dir RW       Write
LINK     COM    16k     123 Dir RW       Write
LOG      DAT     Øk       Ø Dir RW Arcv  None    Ø7/1Ø/86 18:56  Ø7/1Ø/86 18:56
PUT      COM     8k      55 Dir RW       Write                   Ø7/1Ø/86 18:56
RENAME   COM     4k      23 Dir RW       Write                   Ø7/1Ø/86 18:21
SAVE     COM     2k      14 Dir RW       Write
SET      COM    12k      81 Dir RW       Write                   Ø7/1Ø/86 18:52
SETDEF   COM     4k      32 Dir RW       Write                   Ø7/1Ø/86 18:12
.
.
.
Total Bytes     =    326K  Total Records =    24Ø5  Files Found =   29
Total 1K-Blocks =    312   Used/Max Dir Entries For Drive A:   8Ø/ 128
```

Abb. 1

zeugt; bei der Herstellung des DIR-Listings hat sie deshalb noch 0K Länge.)

Bemerkenswert: Zum Lieferumfang von CP/M 3.0 gehört das Assembler-Paket von Digital Research, bestehend aus MAC/RMAC (Macro-Assembler), SID (Debugger), LIB(rary-Manager), LINK(er) und XREF (Cross Reference).
SID hat im Gegensatz zu DDT die Möglichkeit, permanente Breakpoints zu setzen, kann Symbol-Tabellen laden, ist aber ein 8080-Debugger wie DDT und somit nicht Z80-fähig. LIB kann jede Library (z.B. von Sprachcompilern nach Microsoft-Standard, etwa FORTRAN-80, COBOL-80 usw.) wie eine Datenbank indizieren, d.h. der LINKer muß die betreffende Library nicht sequentiell durchforsten (was ja z.B. bei FORTRAN eine respektable Pause erzwingt), sondern kann gezielt auf die Module zugreifen, die wirklich benötigt werden. Das bewirkt verkürzte Laufzeiten bei LINK. Beispiel: Fortran-Programm mit nur einem PRINT-Statement und dem Compiler FORTRAN-80:

```
LINK.COM
Library nicht indiziert: Zeit: 62s
Library indiziert:       Zeit: 49s
```

Der Vorteil wird aber erst beim Linken größerer Programmpakete und umfangreicher Libraries richtig deutlich.

Einige weitere Features bei anderen Kommandos in Stichworten:
ERA(SE) kann mit der CONFIRM-Option laufen, d.h. die Frage „Löschen Ja/Nein" erscheint bei jeder einzelnen Datei. REN(AME) läßt in beschränktem Maße Wildcards zu.
Das alte Sammelkommando STAT ist hier auf verschiedene Kommandos verteilt worden, z.B. auf SHOW (SPACE, LABEL, USERS, DRIVE), SET (Fileattribute), DEVICE (Zuweisung von Einheiten, war bei der getesteten Version nicht implementiert); dies ist allerdings auch mit einer Inflation der Anzahl möglicher Optionen verbunden.
Die Standardroutinen ED.COM und PIP.COM sind verbessert bzw. erweitert worden: PIP z.B. beinhaltet die schon genannte A(rchive)-Option zum schnellen File- und Disketten-Backup und kann Files zwischen beliebigen Userebenen kopieren. ED hat offenbar als wesentliche Neuerung die Möglichkeit zum Verschieben von Zeilenblöcken erhalten.
Es gibt aber aus Anwendersicht auch Nachteiliges zu berichten. Das vertraute SAVE-Kommando z.B. funktioniert hier ganz anders. Aus Gründen, die mit dem CCP zusammenhängen (s. unten), muß man SAVE schon starten, bevor im Speicher die „SAVEnswerten" Daten erzeugt werden. Man muß sozusagen schon vorher wissen, ob hinterher eine Situation entsteht, in der ein Speicherbereich gesichert werden soll. Den einfachsten Fall stellt das Arbeiten mit SID dar: Zuerst SAVE-Kommando, dann SID starten, Verarbeitung ablaufen lassen, SID beenden; nun meldet sich wie aus heiterem Himmel das SAVE-Kommando automatisch zurück und will Speicherlänge, Dateinamen, etc. wissen. Umständlich – aber wahrscheinlich nicht anders zu realisieren!

7. Standard-Software und Kompatibilität

Getestet wurden u.a. Wordstar 3.0, dBase II und Turbo-Pascal. Bei keinem Programmsystem tauchten Schwierigkeiten auf.
Wie schon vorab vermutet, lief das ohnehin schnelle Turbo-Pascal wegen der hohen Taktfrequenz noch um einiges schneller. Die numerische Geschwindigkeitssteigerung um den Faktor 3-4 (Softcard 2 MHz gegenüber CP/M+ mit 6/8 MHz) konnte bei Verarbeitungen im Speicher (allerdings nur in der TPA-Bank) praktisch bestätigt werden.
Problematisch mit der Kompatibilität zum „alten" CP/M wird es aber, wenn Programme nicht die Standard-BDOS- und -BIOS-Schnittstellen benutzen oder sich quasi als „Ersatz-CCP" etablieren wollen. Beispiel hierfür ist das System Power, mit dem man einen vollen Absturz erlebt. Auch das MicroShell-Paket stürzt ab, wenn versucht wird, Shell-Files mit Console-Eingaben abzuarbeiten.
Ein wesentlicher Teil der Möglichkeiten gerade dieser beiden Programme ist aber beim CP/M 3.0 schon im System integriert, so daß der „Abschied" nicht sehr schwer fällt.
Nicht ganz so leicht fällt der Verzicht auf GBASIC: Da die Umschaltroutine zum 6502 hier an anderer Stelle (und in einer anderen Bank) liegt, kommt GBASIC leider nicht zu seinen Applesoft-Routinen.
Ob noch weitere Inkompatibilitäten bestehen, konnte von den Autoren nicht geprüft werden. Vielleicht haben einige Peeker-Leser schon Erfahrungen in dieser Richtung gesammelt.

8. Assembler-Schnittstellen, BDOS- und BIOS-Funktionen

Zwei Begriffe vorab: Der Speicherbereich von 64K auf dem Apple Motherboard wird als „Apple-Bank" bezeichnet und hat die Banknummer 0. Die zusätzlichen 64K auf der Z80H-Karte werden „TPA-Bank" genannt: Die Nummer dieser Bank ist 1. (Theoretisch kann CP/M 3.0 noch weitere Bänke verwalten.) Wie wird zwischen den Bänken umgeschaltet? Mit einem Z80-OUT-Befehl, der sonst beim CP/M 2.2 nie verwendet wird, schreibt man eine 0 oder eine 1 auf einen Z80-Port, der nur aus einem Flip-Flop besteht. Dessen logischer Zustand selektiert dann die physikalische Speicherbank 0 oder 1.
Nun zu BDOS und BIOS. Das BDOS (der geräteunabhängige Teil von CP/M) wurde stark erweitert. Die höchste vergebene BDOS-Funktionsnummer ist 152; freilich klaffen bis dorthin riesige Lücken – und

doch kommen etwa 70 Funktionen zusammen.
Der Zugriff bleibt allerdings wie von CP/M 2.2 bekannt: Funktionsnummer im Register C und notwendige Parameter in Register DE übergeben, dann „CALL 0005H" und die „Antwort" steht in A und HL.
Hier einige der „Neulinge":

46: Get Free Disk Space
Input: Drive in E
Output: In den ersten 3 Bytes des DMA-Puffers steht (binär) die Anzahl freier (128-Byte-) Records.

47: Chain to Program
Input: 0FFh in E, Kommandozeile im DMA-Puffer (Ende 00h)
Wirkung: Die Kommandozeile und damit bspw. ein Programm wird ausgeführt, ohne daß der Benutzer eingreifen muß. (Vergleichbar mit dem CHAIN-Kommando beim alten INTBASIC)

111/112: Print/List Block
Input: Adresse des Character Control Blocks (CCB) in DE (CCB enthält Adresse und Länge eines Strings)
Wirkung: String wird (Tabs expandiert) auf Bildschirm oder Drucker ausgegeben.

152: Parse File Name
Input: Adresse des „Parse File Name"-Control Blocks (PFCB) in DE. Der PFCB enthält die Adresse eines ASCII-Strings und die Zieladresse für einen FCB (= File Control Block).
Wirkung: Hiermit wird ein kompletter FCB (an der angegebenen Speicherstelle) aus der Information des ASCII-Strings (Filename, Drive,...) erzeugt.

Diverse BDOS-Aufrufe beschäftigen sich mit der erweiterten Fileverwaltung: Disk-Labels, Passwords, Datumstempel etc.
Die interessanteste Funktion ist aber wohl folgende:

50: Direct BIOS Calls
Input: Adresse eines BIOS-Parameter-Blocks (BPB) in DE. Dieser BPB enthält der Reihe nach die BIOS-Funktionsnummer und 7 Bytes zur Übergabe der Register A, BC, DE, HL.
Wirkung: Ausführung der gewünschten BIOS-Funktion. Dabei übernimmt BDOS die Auswahl der richtigen Speicherbank.

Eine Warnung: Versuchen Sie nie, eine BIOS-Funktion durch direkten Einsprung in die BIOS-Sprungleiste aufzurufen! Bei mehr als der Hälfte der Aufrufe landen Sie im Niemandsland, da neben der Sprungadresse auch noch Informationen über die anzuwählende Bank (Apple-Bank oder TPA-Bank) benötigt werden. Verwenden Sie ausschließlich die BDOS-Funktion 50 = Direct BIOS Calls.
Folgen wir der BDOS-Funktion 50 und lassen wir das BIOS Revue passieren.
Damit uns das Umdenken nicht zu schwer gemacht wird, sind die ersten 16 Funktionen in Namen und Auswirkung gegenüber CP/M 2.2 gleich geblieben. Das BIOS wartet aber noch mit weiteren 19 Funktionsnummern auf (zumindest in der ALS-Version von CPM 3.0).
Die Funktionen 17 bis 32 lassen sich in drei Gruppen einteilen:

– Erweiterte Status-Funktionen:
Der bisher nicht abfragbare Status einer seriellen Karte (AUX) und der Bildschirmausgabe (CON) werden durch die Funktionen AUXIST (18), AUXOST (19) und CONOST (17) erfaßt.

– Die Verwaltung der Speicherbänke:
Hierzu zählen MOVE (25), SELMEM (27) und SETBNK (28). MOVE nutzt den LDIR-Befehl des Z80 aus, um Speicherblöcke zu verschieben. SETBNK merkt sich die gewünschte Banknummer (0 = Apple-Bank, 1 = TPA-Bank), während SELMEM diese dann einschaltet.

– Die (beim ALS-CP/M 3.0) nicht implementierten Funktionen:
Das sind folgende: XMOVE (Bank-zu-Bank-Transfer), FLUSH (Puffer säubern), MULTIO (mehrere Sektoren lesen/schreiben), TIME (Hardware-Uhr), DEVINI und DEVTBL (Verwalten von Ein-/Ausgabekanälen) und drei weitere „leere" Funktionen (einfacher RET-Befehl).

Die Funktionen 33 bis 35 (AREAD, AWRITE, ACALL) betreiben die Kommunikation des Z80-Prozessors mit dem 6502-Prozessor. AREAD liest mit dem 6502 den Wert an der Adresse (HL) (Zeiger-Adressierung) in das A-Register ein. AWRITE schreibt umgekehrt von A nach (HL). ACALL läßt den 6502 ein Unterprogramm ausführen, dessen Adresse in HL übergeben wird.
„Wozu dieser Umstand?" werden Sie fragen, „Ich lasse lieber gleich alles vom Z80 erledigen." Doch aufgrund der Speicherorganisation, die wir anschließend besprechen, erreicht der Z80 nicht alle Adressen in der Apple-Bank; hierfür ist der Umweg über AREAD und AWRITE erforderlich. Die Daseinsberechtigung von ACALL ist klar, wenn man weiß, daß das gesamte Disk-I/O stets von 6502-Routinen abgewickelt wird.

9. Boot-Vorgang und Speicherverwaltung

Nach dem Einschalten liest der Apple wie üblich Spur 0 Sektor 0 nach $800 und springt nach $801. In dieser Routine stehen Informationen über Anzahl und Zieladresse der von den Systemspuren zu ladenden Sektoren. Nun werden die Prozessor-Umschaltroutinen (Z80 ↔ 6502) in die Page 3 geschoben; erst dann beginnt die Suche nach der Z80H-Karte. Diese übernimmt nach Ansprechen von $Cn00 das Kommando und versetzt den 6502 in den „Halbschlaf". Auf der Z80-Adresse 0000h (entspricht Apple-Adresse $9000,

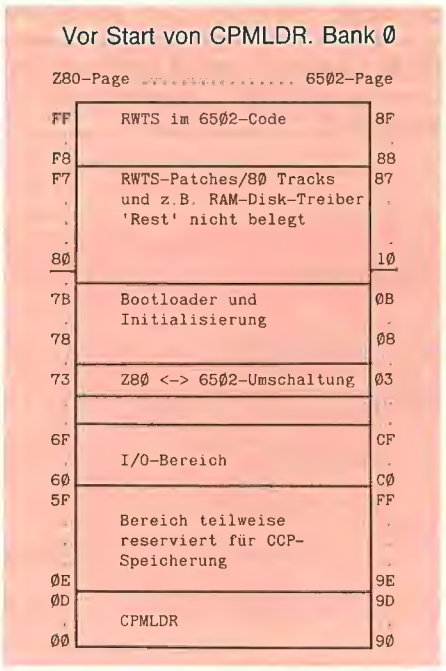


Abb. 2

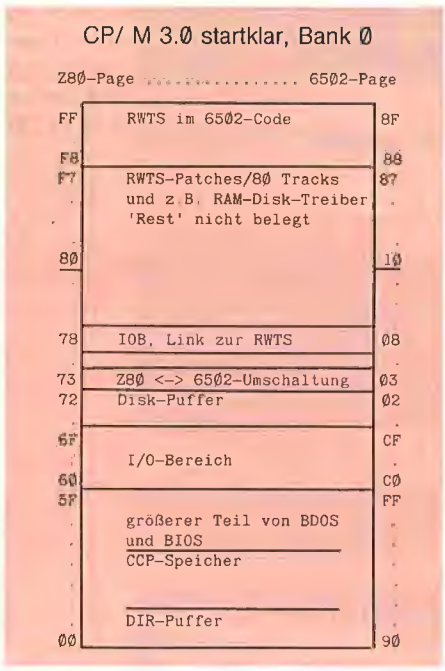


Abb. 3

s. unten) liegt der CP/M-Loader (CPMLDR) – ein Mini-CP/M mit allen Funktionen, um das eigentliche Betriebssystem (die Datei CPM3.SYS) von Diskette zu laden.

Kleinere Teilaufgaben (z.B. das Einschalten der 80-Zeichenkarte) werden dabei wieder von der kurzzeitig aktivierten 6502-CPU erledigt.

In **Abb. 2** ist die Belegung der Apple-Bank nach dem Laden der Systemspuren, in **Abb. 3** nach Ablauf des CPMLDR dargestellt. Hierbei ist auch gleich die etwas unorthodox anmutende Umrechnung der Adressen Z80 ↔ 6502 erkennbar.

Die letzten Aktionen beim Booten sind die Ausgabe der Startmeldung und das Ausführen der BIOS-Funktion 0, des Kaltstarts. Dieser initialisiert den CP/M-Vorsatz (Adreßbereich 0 – 100h) und lädt den CCP aus der Apple-Bank. Vor dem üblichen Prompt-Zeichen sucht der CCP – wie bereits geschildert – nach der Datei PROFILE.SUB und führt sie gegebenenfalls aus.

Die TPA-Bank enthält nur kleine Teile von BDOS und BIOS sowie Datenpuffer. Der Rest steht für Programme zur Verfügung. Die 6/8MHz-Taktfrequenz der Z80H-Karten gilt nur für diese TPA-Bank, in der Apple-Bank geht es mit den bekannten 2 MHz gemächlicher zu. Anwenderprogramme laufen also drei- bis viermal so schnell ab – sofern keine Systemfunktionen ausgeführt werden, die i.allg. in der Apple-Bank ablaufen.

Um zwischen der TPA-Bank und der Apple-Bank überhaupt kommunizieren zu können, verlangt CP/M 3.0 einen Speicherbereich, der von beiden Bänken aus zugänglich ist, den sog. COMMON-Bereich. Je nach Typ der Z80H-Karte ist das ein Bereich bei den hohen Adressen, dessen Länge hardwareabhängig ist. Er liegt physikalisch stets im RAM der Z80H-Karte. Der adressenmäßig parallel liegende Speicherbereich der Apple-Bank ist nicht vom Z80 (aber natürlich vom 6502) erreichbar. Nebenbei soll noch erwähnt werden, daß die 2. Bank bzw. 4K der Language-Card nicht benutzt werden.

Die COMMON-Grenze bei 8000h gilt für die ALS-Karte.

Interessant ist, daß auch nach dem Booten in der Apple-Bank eine Kopie des CCP gespeichert wird. Vorteil: Nach dem Ende eines transienten Kommandos oder eines Programms kann der CCP aus der Apple-Bank in die TPA-Bank gefahren werden, ein Nachladen von der Diskette ist überflüssig. Diese Methode bringt Zeitvorteile. Da der CCP stets nach 100h in die TPA geladen, also wie ein transientes Programm behandelt wird, werden die vorher ab 100h stehenden Daten überschrieben. Mit diesem Hintergrund wird es Ihnen leicht fallen, die etwas umständliche Handhabung des SAVE-Kommandos zu verstehen.

Noch eine Bemerkung zur Speicherverwaltung auf Disketten: Hier gibt es aus Anwendersicht nichts Neues zu berichten. Die einzige Ausnahme ist die, daß bei Laufwerken mit über 40 Spuren eine Blocklänge von 2K zugrunde gelegt wird. Bei CP/M 2.2 wird hierbei oft 4K verwendet, was gerade bei kleinen Files viel Platz verschwendet. Dadurch entsteht in der Praxis ein Problem: Wie liest man „alte" Disketten mit 4K Blocklänge beim CP/M 3.0 ein? Für diesen Zweck lag bei der getesteten Version die Utility IOFORMAT vor, mit der ein einzelnes Laufwerk unter CP/M 3.0 auf die größere Blocklänge getrimmt werden kann – sowohl für das Lesen als auch für das Schreiben.

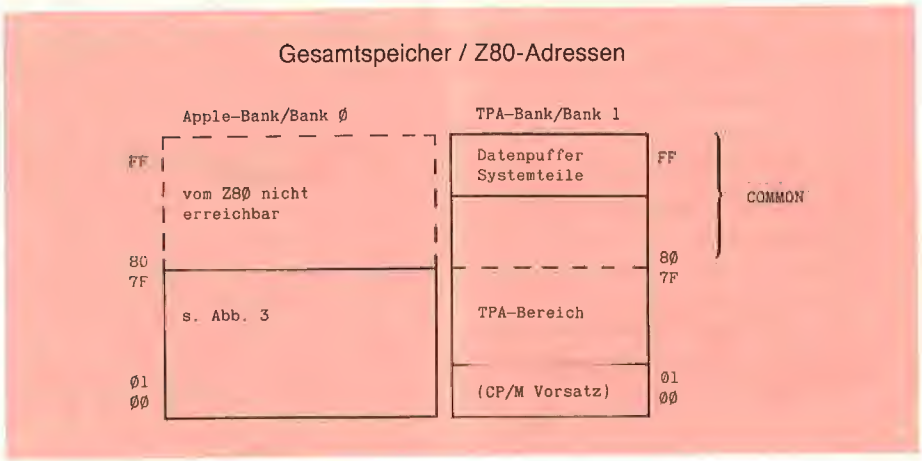


Abb. 4

Code Optimizer

Mit Quelltext der Kyan-Library

getestet von Matthias Meyer

1. Einsatzgebiet

Der Code Optimizer für Kyan-Pascal wurde entwickelt, um den vom Compiler erzeugten Objectcode wesentlich schneller und kürzer zu machen, als man es bisher von dieser Programmiersprache her gewohnt ist. Er ist ein wertvolles Hilfsmittel für diejenigen Pascal-Programmierer, denen der vom Standard-Compiler erzeugte Objectcode nicht schnell oder nicht speichereffizient genug ist, um die geforderte Aufgabe in Pascal zu lösen. Der Code-Optimizer enthält den Quellcode der gesamten Kyan-Pascal Runtime-Library und führt zahlreiche Vereinfachungen an dem vom Compiler erzeugten Assembler-Quellcode durch, die nachfolgend im einzelnen beschrieben werden.

2. Leistungsumfang

2.1. Überblick

Die Kombination aus Kyan-Pascal-Compiler und integriertem Assembler für die Apple-II-Familie wird mit diesem Software-Toolkit von Kyan, Inc. um eine weitere Zwischenstufe bis zur Erzeugung des ablauffähigen Maschinencodes erweitert: Das Pascal-Programm wird erst zu einem Assembler-Quellcode compiliert. Dieser wird darauffolgend so optimiert, daß nur die im Programm verwendeten Segmente der Runtime-Library in das Quellprogramm eingebunden und zahlreiche Macro-Aufrufe zusammengefaßt werden. Schließlich wird das so entstandene optimierte Assembler-Programm zu ausführbarem Maschinencode assembliert.
Grundsätzlich ist der Optimizer in der Entwicklungsphase eines Programms völlig entbehrlich. Die Architektur des Kyan-Pascal-Systems erzielt in den meisten Fällen recht gute Ergebnisse. Compiler und Assembler erzeugen einen schnellen und relativ kompakten Code. Probleme ergeben sich erst dann, wenn die Anwendungen in Größe und Komplexität wachsen und der Programmierer durch die Grenzen der Apple-II-Hardware größere Einschränkungen in Kauf nehmen muß. Der Code Optimizer beseitigt hier gleich zwei Probleme auf einmal: Einerseits wird der zur Verfügung stehende Speicherplatz besser genutzt, andererseits wird die Ablaufgeschwindigkeit vieler Pascal-Programme wesentlich erhöht.

2.2. Arbeitsweise

Der Code Optimizer arbeitet als unmittelbare Zwischenstufe zwischen Pascal-Compiler und Macroassembler. Der Optimizer liest den vom Compiler erzeugten Textfile (P.OUT) und bewirkt folgendes:

1. Er entfernt das Code-Segment zum Laden der Runtime-Library.
2. Er fügt „Include"-Anweisungen für die zwei Library-Quellfiles und für den Optimizer-Macrofile ein.
3. Er liest und überprüft den File P.OUT und ersetzt bestimmte Kombinationen von Compiler-Macros durch Optimizer-Macros.
4. Er erzeugt einen neuen Textfile mit dem Namen OP.OUT und speichert diesen auf der Arbeitsdiskette.
5. Er ruft den Kyan-Macroassembler auf, der dann den File OP.OUT assembliert und einen optimierten Objectcode-File erzeugt, welcher die Runtime-Library im Gegensatz zu normal compilierten Programmen nicht mehr benötigt.

2.3. Was wird optimiert?

Der Code Optimizer erzielt seine hauptsächlichen Optimierungen in den nachfolgend aufgeführten Gebieten. Es ist möglich, daß sich manche Gebiete überlappen; in diesen Fällen kommt es zu mehrfachen Optimierungen.

1. Lokale und globale Variablen, denen eine Konstante zugewiesen wird.
2. Globale Variablen, denen eine andere globale Variable zugewiesen wird.
3. Verweise auf indizierte Variablen (Feldelemente).
4. Formeln, die Integer-Konstanten und globale Variablen beeinflussen.
5. Globale Variablen und Konstanten, die zum Inhalt des Stapels addiert oder davon subtrahiert werden.
6. (-R-Option aktiv): Verweise auf von Konstanten indizierte globale Variablen
7. (-R-Option aktiv): Verweise auf von Variablen indizierte globale Variablen

Diese Kategorien repräsentieren laut Handbuch ca. 70% der Ausführungszeit eines typischen Pascal-Programms. (Die -R-Option bewirkt, daß der Optimizer Bereichsprüfungen bei Zugriffen auf Feldelemente wegläßt.)

2.4. Geschwindigkeitsmessungen

Zur Demonstration findet man auf der Diskette ein einfaches Benchmark-Programm. Um auch den Vergleich mit anderen Pascal-Compilern zu ermöglichen, nachfolgend das kurze Listing dieses Programms :

```
PROGRAM SIEVE;
(* SIEVE: STANDARD PASCAL BENCHMARK
   PROGRAM.  CALCULATES # OF PRIMES
   BETWEEN Ø AND 16384 (SIZE*2) *)
CONST SIZE=8192;
VAR
  COUNT,I,K,PRIME:INTEGER;
  F:ARRAY [Ø..SIZE] OF BOOLEAN;
BEGIN
  WRITELN('1 ITERATION');
  COUNT:=Ø;
  FOR I:=Ø TO SIZE DO F[I]:=TRUE;
  FOR I:=Ø TO SIZE DO
  IF F[I] THEN
  BEGIN
```

```
      PRIME:=I+I+3;
      K:=I+PRIME;
      WHILE K<=SIZE DO
      BEGIN
        F[K]:=FALSE;
        K:=K+PRIME
      END;
    COUNT:=COUNT+1;
  END;
  WRITELN(COUNT,' PRIMES')
END.
```

Das Programm berechnet die ersten 1899 Primzahlen in folgender Zeit:
Nicht optimiert: 16,2 Sekunden
Optimiert: 6,9 Sekunden.
Diese Zeiten wurden von mir mit einer normalen Stoppuhr auf einem nichtgetunten Apple IIe gemessen. Die Geschwindigkeitssteigerung durch den Optimizer ist also beachtlich.

Bewertung: Der Code Optimizer ist für denjenigen interessant, der noch mehr aus seinem Pascal-System herausholen möchte oder kommerzielle Programme auf der Basis dieses Pascal-Systems in der Entwicklung hat. Der Einsatz des Optimizers auf normalen 140K-Laufwerken erscheint wenig sinnvoll. Zum einen wird ein weiterer großer Assembler-Quellfile erzeugt, der bei längeren Programmen zwangsläufig zum Überlaufen der Diskette führt, zum anderen benötigt der Assembler zum Verarbeiten der gesamten Source Code Library wesentlich mehr Zeit, als dies beim normalen Compilieren der Fall ist. Dies sind jedoch Beschränkungen, die durch die Hardware bedingt sind. Eine Festplatte sollte hier Abhilfe leisten. Der Optimizer selbst macht jedenfalls den ohnehin nicht schlechten Kyan-Pascal-Compiler noch um einiges besser.

3. Dokumentation

Toolkit VI von Kyan, Inc. besteht aus einer einseitig beschriebenen Diskette und einer Loseblattsammlung von 16 Blättern zum Einheften in einen Ordner, der das gleiche Format wie der des Kyan Pascal User Manuals hat. Ein von Kyan, Inc. für diesen Zweck vertriebener Ordner kostet US$ 9.95 incl. Porto. Man kann sich aber statt dessen auch einen gewöhnlichen DIN-A5-Ordner kaufen und alle Blätter nochmals lochen.

Bewertung: Die englischsprachige Dokumentation beschreibt alle wesentlichen Abläufe des Code Optimizers im Detail. Im Anhang findet man Hinweise, wie man den Optimizer in Verbindung mit den Toolkits II und III (MouseText und Advanced Graphics) anwenden kann und wie die Runtime-Module dieser Toolkits mit Hilfe der mitgelieferten Hilfsprogramme zu relozieren (im Speicher zu verschieben) sind. Dann wird die Speicherplatzeinsparung bei Verwendung des Optimizers in Verbindung mit den Toolkit-Demo-Programmen tabellarisch gegenübergestellt. Schließlich folgen noch Anmerkungen über die auf der Diskette enthaltenen Quellfiles. Die Dokumentation beantwortet alle wesentlichen Fragen und ist daher als komplett zu bezeichnen.

Ein-Blick

Name:	Code Optimizer Toolkit
Konfiguration:	Apple II/II+/IIe/IIc Kyan Pascal ab Vers. 2.Ø evtl. weitere Toolkits
Einsatzgebiet:	Optimierte Compilierung
Gesamtwertung:	*****
Dokumentation:	**** (3Ø Seiten)
Zweckdienlichkeit:	*****
Verpackung:	****
Preis/Leistung:	**** ($149.95)
Hersteller:	Kyan Software, Inc., San Francisco, USA

(Die beste Bewertung entspricht 5 Sternen.)

MS-DOS auf dem Apple IIe

Ein Erfahrungsbericht

von Dr.-Ing. H. Gäthje

1. Ausgangsposition

Seit April 1986 gehört mein Apple IIe auch zum sog. Industriestandard. In Slot 2 befindet sich die C86-Erweiterungskarte der Firma A. Peter & Partner aus Berlin. Fast alle betriebssystemgestützten IBM-PC-Programme sind auf diesem 16-Bit-Computer mit seinem eigenen Speicherbereich von 512K lauffähig.
Der zum Test benutzte Apple IIe ist folgendermaßen bestückt:

Slot 1: Epson-Interface für Drucker FX-80+
Slot 2: C86-Karte mit 8086-CPU und 512K RAM
Slot 3: 80-Zeichenkarte mit 64K
Slot 4: Z-80-Softcard
Slot 5: frei
Slot 6: Apple-Interface für 2 Diskettenlaufwerke mit 140K, wahlweise Erphi-Controller für 2 Diskettenlaufwerke mit 640K.

Erhältlich zur 8086-Steckkarte sind die Betriebssysteme MS-DOS 2.11 und CP/M 86 sowie 2 RAM-Floppy-Disketten für die Systeme CP/M 80 und DOS 3.3. Das System CP/M 86 habe ich nicht mitbestellt.

2. C86-Karte als RAM-Disk

2.1. DOS 3.3

Die mitgelieferten RAM-Floppy-Disketten enthalten entsprechende Installierungsprogramme. Unter DOS 3.3 läßt man das Programm beim Booten durch das Hello-Programm aufrufen und erhält zusätzlich zur RAM-Disk s3,d1 die virtuellen Disketten s1,d1 bis s1,d4. Die Übertragung von Files von den Disketten zu den RAM-Disks und zwischen den RAM-Disks ist möglich; Kompatibilitätsprobleme treten dabei nicht auf. Es ist sogar möglich, mit einer Systemkonfiguration von 4 externen Laufwerken zu arbeiten. Folgende Diskettenaufrufe stehen dann zur Verfügung:

s1,d1: C86 128K
s1,d2: C86 128K
s1,d3: C86 128K
s1,d4: C86 128K
s3,d1: 80-Zeichenkarte 64K
s5,d1: Apple-Drive 1 140K
s5,d1: Apple-Drive 2 140K
s6,d1: Erphi-Drive 1 640K
s6,d2: Erphi-Drive 2 640K.

Dieses System ist nur für Freaks mit guten Nerven geeignet. Bei diesen vielfältigen Möglichkeiten des Datentransfers raucht nicht nur der Kopf, deshalb sollten die Blenden an der Rückseite des Apple entfernt werden. Ein Anschluß einer Speicher-Erweiterungskarte auf 1M ist auf der C86-Karte bereits vorgesehen. Die C86 sollte dann extern gespeist werden, damit die Daten beim Ausschalten des Apple nicht verlorengehen.

2.2 CP/M-80

Wie aus der technischen Beschreibung hervorgeht, hat das Interface der C86-Karte eine eigene Dekodierung des Apple-Adreßbusses, um Interferenzen mit anderen Slotkarten (Saturn 128K, Super Serial Card, Z80-Karte, 80-Zeichenkarte usw.) zu vermeiden. Sie ist deshalb für den Betrieb in Slot 2 vorgesehen. Starten des mitgelieferten Programms RF.COM unter CP/M-80 initialisiert die RAM-Disk F: mit 512K. Bei der unter DOS 3.3 beschriebenen Konfiguration treten Kompatibilitätsprobleme auf. Deshalb muß das Interface in Slot 5 entfernt werden. Dann stehen die Floppies A:, B:, C: und F: zur Verfügung. Beim Transfer der Daten von C: nach F: stürzte das System ab. Bei einer 512K-RAM-Floppy fallen die 64K der 80-Zeichenkarte nicht besonders ins Gewicht.
Schreibt man sich eine kleine SUBMIT-Datei, wird automatisch die RAM-Floppy F: initialisiert, die gewünschten Files in das RAM gespeichert und das gewünschte Programm gestartet. Auch größere dBase-Dateien haben jetzt beim Sortieren ihren Schrecken verloren.

3. C86-Karte als MS-DOS-Rechner

Um mit dem Betriebssystem MS-DOS 2.11 zu arbeiten, wird die Bootdiskette in das Laufwerk A: gesteckt und ein Kaltstart durchgeführt. Der Apple wird für die C86-Karte vorbereitet. Nach erfolgtem Booten ist ein Diskettenwechsel erforderlich. Diese Diskette enthält das für den Apple modifizierte MS-DOS-Betriebssystem. Nach dem Drücken der Return-Taste wird MS-DOS automatisch initialisiert, zusätzlich wird eine RAM-Disk C: eingerichtet. Die AUTOEXEC.BAT-Datei ist so gestaltet, daß man sich nach dem Lauf in der virtuellen Disk C: befindet. Der freie Speicherplatz auf der RAM-Disk C: beträgt 380K. Mit dem Editor EDLIN ist diese Datei leicht zu ändern. Die angeschlossenen Laufwerke haben die Bezeichnung ‚A:' und ‚B:'. Sind Apple-Laufwerke vorhanden, so steht eine Kapazität von 138K pro Laufwerk zur Verfügung. Ein neuer Diskettencontroller ist nicht erforderlich.

Es ist möglich, komplette Anwendersoftware, z.B. dBase II oder Multiplan, in die RAM-Disk zu kopieren. Beim Bearbeiten der Dateien bzw. Spreadsheets und Laden der Programme ist der Geschwindigkeitsunterschied zum CP/M-System enorm. Die CPU 8086 ist mit 8MHz getaktet. Damit stellt der „MS-DOS-Apple" den IBM-PC in den Schatten (Anmerkung: Beruflich arbeitet der Verfasser mit einem IBM AT 02). Fast alle MS-DOS-Kommandos können aufgerufen werden. Dateiverzeichnisse (Subdirectories) sind auf Diskette und RAM-Disk mit dem Befehl MKDIR zu errichten. Damit wird für die bisherigen CP/M-USER mit Hilfe dieser Karte eine neue Ordnung eintreten. Die Umstellung zur neuen MS-DOS-Befehlsstruktur ist nicht schwierig, zum Kopieren ist nur eine andere Denkrichtung nötig. Hierzu ein Beispiel:

CP/M-80:
PIP B:NEU.TXT=A:ALT.TXT
MS-DOS:
COPY A:ALT.TXT B:NEU:TXT

Der FORMAT-Befehl ist bei den Apple-MS-DOS-Disketten für die Apple-Laufwerke nicht anwendbar. Mit einem Bitkopierer unter DOS 3.3 ist eine Kopie der Systemdiskette zu erstellen, anschließend werden mit dem ERASE-Befehl unter MS-DOS die vorhandenen Dateien gelöscht.
Optional ist eine Disketten-Controller-Karte erhältlich, die es ermöglicht, sowohl DOS-3.3-Format als auch IBM-Format zu lesen und zu beschreiben.

4. Gesamtbewertung

Wer einen Apple IIe mit einer Z80-CPU-Steckkarte besitzt und nach einer Speichererweiterung Ausschau hält, sollte DM 998,– investieren und zur C86-Karte greifen. Mit dem Kauf des MS-DOS-Betriebssystems laufen IBM-Programme, die nicht auf den speziellen Zeichensatz oder die Grafik des IBM zugreifen, ohne Komplikationen. Die zusätzlichen Kosten für die reine Speichererweiterung sind minimal. Für den kommerziellen Einsatz ist dieser „erweiterte Apple" nicht geeignet. Das Risiko, daß ein gekauftes Anwenderprogramm, das für IBM oder Kompatible geschrieben wurde, nicht zur Zufriedenheit arbeitet, ist zu groß. Für den Apple-Freak jedoch sollte diese Karte eine Herausforderung sein: er kann sich mit noch weiteren Betriebssytemen „herumschlagen". Auch TURBO-Pascal läuft unter MS-DOS und auf der C86.

Wer in seinen Apple IIe und in die Peripherie schon ca. DM 8000,– gesteckt hat, kann heute nicht mehr auf einen IBM-Kompatiblen umsteigen. Er ist vorerst mit der C86-Karte besser bedient und kauft sich in einem Jahr den 32-Bit-Rechner!

PEEKER Börse

Gelegenheitsanzeigen

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

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.

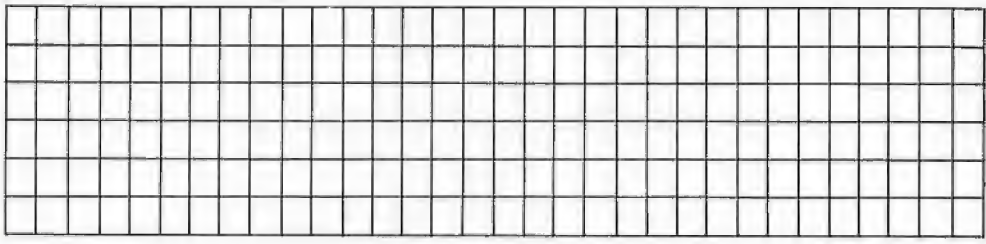
PEEKER

Peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

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

☐ suche Hardware ☐ suche Software ☐ Verschiedenes
☐ biete Hardware ☐ biete Software ☐ Chiffre



Bitte jeweils 32 Buchstaben pro Zeile – einschließlich Satzzeichen und Wortzwischenräume. Bitte Absender nicht vergessen. Mindestens 2 Zeilen. Chiffregebühr DM 6,– zuzügl. MwSt.

PEEKER

Produkt-Karte

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

Anzeige über _____

_____

bitte ich um detaillierte Information.

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

Menge	Produkt und Bestellnummer	á 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 ☐ Mac ☐ 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

Umfrage-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker-Börse
Anzeigen-Service

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

POSTKARTE

Inserent

Straße

PLZ/Ort

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.

Vorher 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

XPER
Ein Expertensystem

getestet von Dagmar Berberich

1. Gegenstand

Die Firma Data Becker kündigt ihr Programmpaket XPER als „Expertensystem" und „Einstieg in das faszinierende Gebiet der künstlichen Intelligenz" an.
Was ist ein Expertensystem überhaupt? Expertensysteme sind Programme, die Informationen über bestimmte Themengebiete gespeichert haben. Aus diesen Bereichen kann der Anwender in einem Frage-und-Antwort-Modus gezielt Informationen abrufen. Viele dieser Systeme sind bereits mit Datenmaterial „bestückt", z.B. Systeme zur automatisierten Diagnoseerstellung in der Medizin.
XPER ist dagegen ein frei programmierbares System, d.h. der Anwender muß zunächst alle Informationen zu dem Gebiet, das er mit Hilfe von XPER bearbeiten möchte, „per Hand" eingeben. Dadurch ist XPER aber auch vielseitiger anwendbar als vorgegebene Informationssysteme. Ist ein solches „Basiswissen" programmiert, kann XPER anhand der gespeicherten Daten einen Suchbaum erstellen und ein gewünschtes Objekt näher bestimmen. Auf diese Weise können Objekte oder Werte gesucht und verglichen werden, Übereinstimmungen und Unterschiede zu anderen Objekten können ermittelt werden, die einzelnen Bearbeitungsschritte und das Endergebnis der Bestimmung können ausgedruckt werden.

2. Lieferumfang

Zum Preis von DM 298,– liefert Data Becker eine einseitig bespielte Programmdiskette und ein deutschsprachiges, 158seitiges Handbuch in einem festen Ringbuchordner. Die Diskette ist angeblich kopiergeschützt und soll sich beim Kopierversuch selbst zerstören. Aus diesem Grund bietet Data Becker umsichtigen Kunden zum Preis von DM 20,– eine Sicherungskopie der Programmdiskette an. Von unserer Originaldiskette ließ sich mit dem Diskettenkopierprogramm „Superquick" jedoch problemlos eine Backup-Kopie anfertigen.
XPER läuft auf einem Apple IIe oder IIc mit mindestens einem Diskettenlaufwerk. Insgesamt kann XPER in dieser Konfiguration 90 Objekte, 50 Variablen und 300 Werte verarbeiten und mit 40 Zeichen/Zeile wiedergeben.

3. Dateneingabe

Da XPER keine vorgegebenen Daten enthält, muß der Anwender alle Informationen, die dem System zur Verfügung stehen sollen, selbst eingeben und damit das XPER-Basiswissen aufbauen. Dabei sind 3 Typen von Informationselementen zu unterscheiden: Objekte, Variablen und Werte.
Als *Objekte* des Basiswissens werden die Elemente bezeichnet, die näher beschrieben werden sollen („Gegenstände" der Informationsverarbeitung), z.B. verschiedene Typen von Mikrocomputern.
Variablen sind allgemeine Eigenschaften der Objekte, die deren nähere Bestimmung und Auswahl ermöglichen. Je höher die Anzahl der Variablen ist, desto schneller und genauer läßt sich ein Objekt bestimmen. In unserem Beispiel mit den verschiedenen Mikrocomputern könnte man als Variablen die Größe des Zentralspeichers, die Anzahl der Laufwerke, die Taktfrequenz, den Preis, die Auflösung des Bildschirms etc. festlegen. Bei der Suche nach Objekten werden die Variablen als Auswahlkriterien in den Fragen, die der Computer dem Benutzer stellt, verwandt.
Den dritten Typ der Informationselemente bilden die *Werte*. Sie stellen die möglichen Antworten auf die Fragen des Computers dar, d.h. sie bestimmen die Variablen näher. Dabei können jeder Variablen maximal 14 verschiedene numerische Werte zugeordnet werden. Jeder Nummer entspricht eine bestimmte Antwort auf die Frage nach der Variablen. In unserem Beispiel könnten auf die Frage „Speicherkapazität" die Antworten 1 bis 3 lauten:
(1) unter 48K
(2) 48-64K
(3) 65-128K.
Bevor XPER als „Expertensytem" genutzt werden kann, müssen die Untersuchungsobjekte, ihre Variablen und die Werte der Variablen eingegeben werden. Ausgestattet mit einem solchen Datenmaterial kann XPER arbeiten.

4. Systemaufbau

XPER besteht aus vier eigenständigen Programmteilen (Modulen), die vom Hauptmenü aus abgerufen werden können.
Der **Editor** dient der Informationseingabe und -korrektur, d.h. dem Aufbau des Basiswissens. Eine qualitative und quantitative Ergänzung der Datengrundlage ist jederzeit durch das Hinzufügen zusätzlicher Objekte, Variablen und Werte möglich. Mit Hilfe des Editors können Gruppen von Objekten zusammengefaßt und mit anderen gleichartigen Gruppen verglichen werden. XPER stellt die Unterschiede und Gemeinsamkeiten fest und erläutert diese auf Anfrage. Zwischen einem gegebenen Objekt und den restlichen einer Gruppe kann das System die *Distanz nach Jaccard*, d.h. den Grad der Ähnlichkeit ermitteln und in Form einer Maßzahl angeben. (Völlige Übereinstimmung = Distanz Null, völlige Gegensätzlichkeit = Distanz 10000, Ähnlichkeit bis Distanz < 2000.)
Das zweite Modul ist der **Bestimmer**. Er dient – wie der Name schon sagt – der näheren Bestimmung der Objekte. Die Einkreisung eines gesuchten Objektes erfolgt durch Fragen, die das Programm dem Anwender stellt. Als Antworten sind Auswahlpunkte eines Menüs zulässig. Mit jeder Antwort verringert sich die Zahl der in Frage kommenden Objekte, bis der gesuchte Gegenstand charakterisiert ist. XPER bietet auch die Möglichkeit, eine Liste der bei dieser Auswahl ausgeschiedenen Objekte aufzustellen und die Reihenfolge und Gründe des Ausschlusses aufzuzeigen. Entsprechend können die im Lauf der Entscheidung eliminierten Variablen aufgelistet werden; das sind die Eigenschaften, die im Lauf der Abfrage für die Auswahl des gesuchten Objektes irrelevant werden und deshalb nicht in weitere Fragen aufgenommen werden.
Der dritte Programmteil ist der **Reorganisator**. Er dient dazu, Daten des Basiswissens umzustellen und neu zu ordnen. Die Reihenfolge der Objekte, Variablen und Werte kann damit geändert werden. Ganze Dateien (sog. Arbeitsblätter) können mit Hilfe des Reorganisators miteinander gekoppelt werden, sofern sie einheitliche Datentypen enthalten.
Der **Drucker** dient der Ausgabe der gespeicherten Daten. Alle Informationen, also Objekte, Variablen und Werte können in Textform oder als Kreuzungstabelle ausgedruckt werden. Im Modul „Bestimmer" kann ein Protokollmodus angewählt werden, der die stufenweise Auswahl eines Objektes aus der Gesamtzahl der möglichen Objekte wiedergibt.

5. Programmablauf

XPER ist für den Dialogbetrieb Computer–Anwender konzipiert. Aus diesem Grund ist das gesamte System in Form hierarchischer Menüs aufgebaut. Im Hauptmenü, das nach dem Booten der selbststartenden Programmdiskette erscheint, wählt man aus den vier Programmodulen eines aus. Mehrere Stufen von Unterprogrammen ermöglichen im Editor die Eingabe von Objekten, Variablen und Werten bzw. deren Korrektur und Vergleiche, im Bestimmer das Laden der Arbeitsblätter und die Bestimmung gesuchter Objekte durch eine Abfolge von Fragen und im Reorganisator eine Umstellung der Daten.
Mit einer invers dargestellten Abkürzung am linken unteren Bildschirmrand wird angezeigt, in welchem Untermenü man sich gerade befindet.
Die Menüstruktur dient der besseren Übersichtlichkeit von XPER. Auch als Anfänger läßt sich damit bequem arbeiten. Allerdings wird das Erstellen des Basiswissens zur Fleißarbeit, wenn man die gesamte Kapazität von XPER ausnutzt und viele Daten eingibt, weil man immer wieder in das übergeordnete Menü springen muß, um die Variablen und Werte aufzurufen, die man den einzugebenden Objekten zuordnet.

6. Das Handbuch

Das 158seitige Handbuch ist ausführlich, aber in einigen Punkten recht verbesserungswürdig.
Auf der ersten Seite wird auf die zusätzliche Installationsanweisung für die Computertypen Apple II, Commodore 64 und IBM-PC verwiesen, aber daß sich diese am Ende des Handbuchs befindet, muß man erst herausfinden.
Ausgesprochen negativ zu beurteilen ist die schlechte Sprache des Handbuchs; das Original wurde in Frankreich verfaßt und von einem offenbar nicht sonderlich sprachbegabten Übersetzer ins Deutsche übertragen: schon im

Hinweis auf den Namen dieses Übersetzers befindet sich ein Rechtschreibfehler. Auf den 158 Seiten – besonders gegen Ende des Hauptteils – wimmelt es nur so von Grammatik- und Orthographiefehlern. Offenbar wollte man sich die Kosten für das Korrekturlesen ersparen. Für den Anwender wird das besonders ärgerlich, wenn Befehle oder Menüangaben falsch abgedruckt werden, wie das des öfteren vorkommt.
Unglücklich ist auch die Gestaltung der Hilfe-Kommandos, die man in den einzelnen Untermenüs anwählen kann. Es erscheint jeweils eine unstrukturierte Folge von Buchstaben und Funktionskürzeln, die alle möglichen Befehlseingaben anzeigt. Eine erklärende Hilfe findet man erst in dem Teil des Manuals, der als „Systemhandbuch" bezeichnet wird und auf knapp 50 Seiten alle Befehle und Programmpunkte der vier Module erläutert. Leider fehlt ein entsprechender Hinweis zu Beginn des Manuals oder in den angewählten Help-Funktionen.
Den Herstellern war es offenbar auch nicht möglich, bei der Wiedergabe von Bildschirmmenüs eine inverse Darstellung anzudeuten, und sei es nur in Form von Klammern oder Unterstreichungslinien. Der Benutzer kann sich deshalb aussuchen, welcher Punkt zu der betreffenden Gruppe von Objekten gehört, die invers dargestellt wird.

Fazit

Vom „Einstieg in das faszinierende Gebiet der künstlichen Intelligenz" verspricht man sich sicher mehr, als XPER halten kann.
XPER ist eine gute Hilfe für all jene, die größere Mengen einheitlich strukturierter Daten verwalten und diese Datenvielfalt selektiv nach bestimmten Aspekten durchsuchen müssen. Bei einer solchen Aufgabenstellung kann XPER seinem Anwender schnell einen guten Überblick schaffen, da es Vergleiche bilden, Übereinstimmungen, Unterschiede und Distanzen zwischen verschiedenen Objekten ermitteln kann.
Positiv zu bewerten ist die hohe Transparenz der Auswahlentscheidungen, die dadurch ermöglicht wird, daß eliminierte Objekte und Variablen aufgelistet werden und der Grund für die Eliminierung sowie die Reihenfolge des Auschlusses für alle Objekte aufgezeigt werden. Wer ein komfortables „Auswahl- und Verwaltungsprogramm" für seine Daten sucht, wird mit XPER sicher zufrieden sein. Mit DM 298,– muß er dann allerdings auch recht tief in die Tasche greifen.

Privatbuchhaltung mit PriBu

getestet von Dagmar Berberich

1. Gegenstand

PriBu von Intus Software ist ein Programm, das private Einnahmen und Ausgaben verwaltet und damit einen Überblick über die finanzielle Lage seines Benutzers schafft.
Für einen Preis von DM 195,– erhält man eine selbststartende Programmdiskette, eine Datendiskette und ein 41seitiges Manual im Plastikeinband. Die Disketten sind frei kopierbar (z.B. mit SUPERQUICK oder einem anderen Diskettenkopierprogramm). Für jeden Abrechnungszeitraum muß eine neue Datendiskette angelegt werden, da sonst die Daten des letzten Zeitraums überschrieben werden.

2. Hardware

PriBu läuft auf dem Apple II Plus mit 48K, auf dem Apple IIc und IIe mit mindestens einem Diskettenlaufwerk. Komfortabler läßt sich allerdings mit dem Programm arbeiten, wenn man zwei Diskettenlaufwerke besitzt; dann legt man die Programmdiskette in Laufwerk 1 und die Datendiskette in Laufwerk 2. Die Daten können über einen Drucker ausgegeben werden. Das Interface für den Drucker muß sich im Standardslot 1, für die angeschlossenen Diskettenlaufwerke in Slot 6 befinden.

3. Software

Mit PriBu können Sie 3 verschiedene Arten von Konten anlegen und bearbeiten.
In maximal 19 *Geldkonten* (Kontonummer 1-19) notieren Sie Geldbewegungen, z.B. Ihres Sparbuches, des Girokontos oder der Barkasse. In bis zu 70 *Sachkonten* (Kontonummer 20-89) werden Ausgaben wie Miete, Strom, Urlaub etc. festgehalten. Die dritte Kontenart bilden die *Außenkonten*, die nicht in die aktuellen Abrechnungen miteinbezogen, sondern separat geführt werden. Hier können z.B. der Bausparvertrag verwaltet oder steuerlich absetzbare Ausgaben aufgelistet werden. Insgesamt können 10 Außenkonten (Kontonummer 90-99) von PriBu verarbeitet werden.
Alle Geldbewegungen werden festgehalten: Einnahmen werden in Geldkonten addiert, Ausgaben werden in Sachkonten vermerkt und in Geldkonten abgezogen. Ein monatlicher Abschluß gibt die Gesamtfinanzlage wieder. Daneben kann der Anwender jederzeit den augenblicklichen Stand seiner Konten abfragen und ausdrucken lassen, ebenso ist ein interner Finanztransfer von Konto A zu Konto B möglich.
Das Programm ist als Menü aufgebaut und daher sehr benutzerfreundlich. Fehlermeldungen weisen auf ungültige Eingaben hin, und der Programmablauf wird so lange blockiert, bis eine gültige Eingabe erfolgt.

4. Programmablauf

Im ersten Menü wählt man zwischen dem Programm PriBu Grund und dem Arbeitsprogramm PriBu. Vor der ersten Benutzung müssen mit PriBu Grund die verschiedenen Konten angelegt werden, bei späteren Abläufen können bereits erstellte Konten von der Datendiskette übertragen werden. Dabei ist es wichtig, für die drei verschiedenen Kontenarten (Geld-, Sach-, Außenkonto) Kontonummern aus dem erlaubten Bereich zu wählen und nach der Eingabe alle Werte auf der Datendiskette zu speichern.
Um nach der Anlage der Konten mit PriBu Grund in das Arbeitsprogramm PriBu zu gelangen, muß DOS neu geladen werden. In PriBu erscheint nach Eingabe des laufenden Monats das Hauptmenü, in dem durch Anwahl der Zahlen die entsprechenden Unterprogramme aufgerufen werden.

Tabelle : PriBu-Unterprogramme

E Ende
1 Buchen
2 Abschluss
3 Drucken
4 Neues Konto anlegen
5 Aufteilung eingeben
6 Konto verbessern
7 Datum ändern
A Alles auf Disk

Mit den Unterprogrammen selbst arbeitet man schon nach kurzer Zeit ohne Probleme.

5. Manual

Das mitgelieferte 41 Seiten starke Manual reicht aus, um sich in der Menüstruktur des Programms zurechtzufinden, könnte aber in stilistischer, didaktischer und inhaltlicher Hinsicht ausgebaut und verbessert werden.
Bei der Neuanlage von Konten wären Beispiele oder Help-Funktionen (die im gesamten Programm fehlen) für den Erstanwender nützlich. Manche Hinweise erspart sich der Autor ganz oder bringt sie erst am Ende des Kapitels. Die Übersichtstabellen, die die Abfolge der Menüpunkte, ihre Auswahl und Bedeutung darstellen sollen, sind leider recht unübersichtlich geraten.
Im Anhang findet man ein kurzes Glossar mit den wichtigsten Fachbegriffen, die in PriBu verwendet werden. Dem Anwender, der wenig Vorkenntnisse über Ablauf und Grundsätze der Buchhaltung mitbringt, wäre mit einer etwas ausführlicheren Darstellung von Sinn und Zweck von Gegenkonten, Zwischenkonten und Kontenausgleich besser gedient, zumal PriBu sich als Buchhaltungsprogramm für den Privatanwender versteht.

Fazit

Das Handbuch zum Programm ist zwar verbesserungsfähig, PriBu selbst aber bietet dem privaten Benutzer genügend Möglichkeiten, seine gesamten Finanzen zu ordnen. Das Programm eignet sich für alle, die gerne eine klare Übersicht über den jeweiligen Stand ihrer Geldmittel haben und bereit sind, dafür ihre Einnahmen und Ausgaben stets zu aktualisieren. Bleibt zu hoffen, daß ihnen das Programm auch hilft, gut mit den eigenen Finanzen zu wirtschaften.

DCODE – Utilities für Applesoft-BASIC

Ein Erfahrungsbericht

von Franz-Josef Hüskens

Der Apple II wird zwar mit einer eingebauten höheren Programmiersprache, dem Applesoft-BASIC, geliefert, bietet dem BASIC-Programmierer jedoch wenig Unterstützung. Fehlerhafte Programme oder Programmzeilen können nur umständlich berichtigt werden. Eine Umnumerierung der Zeilen und das Mischen zweier BASIC-Programme sind ohne zusätzliche Software überhaupt nicht möglich.
Die drei Utility-Programme, die in DCode enthalten sind, greifen dem geplagten BASIC-Programmierer bei seiner Tätigkeit hilfreich unter die Arme und ermöglichen u.a.:

1. Verringerung des Platzbedarfs
2. Vergleich zweier Programme bzw. Files
3. direkte Anzeige von Eingabefehlern
4. bessere TRACE-Ausführung.

Die DCode-Programme befinden sich in zwei Versionen auf einer Seite der mitgelieferten Diskette. Eine Version arbeitet unter DOS 3.3, die andere unter ProDOS. Die Diskette sollte jedoch nur dann gebootet werden, wenn man die ProDOS-Version von DCode benutzen will. Doch nun zu den Programmen im einzelnen.

1. COMPACT

Compact verringert den vom BASIC-Programm belegten Speicherplatz auf drei verschiedene, frei wählbare Arten: In das Programm eingestreute REM-Zeilen (Bemerkungen) können gelöscht werden, die Namen der Variablen können verkürzt werden, auf Wunsch werden in eine Programmzeile so viele Befehle wie möglich gepackt.
Das Programm wird mit „BRUN COMPACT" oder – falls es sich bereits im Speicher befindet – mit „&" gestartet. Danach erscheint das Menü des Programms, das folgende Optionen beinhaltet:

1. REMs entfernen
2. Zeilen verknüpfen
3. Variablennamen verkürzen
4. Variablen umbenennen
5. Programmteil „packen"
6. Tabelle der Variablen drucken

C Compact starten
Q Programm beenden

Durch die Eingabe der Ziffer oder des Buchstabens wird die entsprechende Option aktiviert.
Was bewirken die einzelnen Auswahlpunkte?

C Compact starten und **Q Programm beenden**
Diese beiden Menüpunkte erklären sich aus ihrem Namen.

1. REMs entfernen
Kommentare (REM-Statements) in einem Programm dienen zwar der besseren Lesbarkeit, verschwenden jedoch jede Menge Speicherplatz. Jedes Zeichen, das dem REM-Kommando folgt, belegt ein Byte im Speicher. Die „Arbeitsversion" eines Programms sollte aus diesen Gründen keine REMs enthalten.

2. Zeilen verknüpfen
Jede Zeilennummer, die entfernt werden kann, spart vier Speicherzellen. Compact packt daher möglichst viele Befehle in *eine* Programmzeile. Zeilen, die von anderen Zeilen aus angesprungen werden, werden nicht gepackt. Man muß aber beachten, daß Compact auf diese Art Programmzeilen schaffen kann, die so lang sind, daß sie nicht mehr editiert werden können.
Die DCode-Diskette bietet aber auch hier Hilfe in Form des Programms LINE.SPLITTER. Damit werden gepackte Programmzeilen ungefähr in der Mitte „auseinandergeschnitten". Der „obere Teil" der Zeile erhält dabei eine Zeilennummer, die um eins höher ist als die des „unteren Teils". Ist diese Nummer bereits belegt, wird der Benutzer aufgefordert, sie zu ändern.

3. Variablennamen verkürzen
Wie bekannt sein dürfte, beachtet der Applesoft-Interpreter nur die beiden ersten Zeichen eines Variablennamens. Der Name selbst kann zur genaueren Beschreibung der Variablen beliebig lang sein. Wird Option 3 gewählt, verkürzt Compact alle langen Variablennamen auf zwei Zeichen.

4. Variablen umbenennen
Alle Variablen eines Programms werden so umbenannt, daß die Bezeichnungen so kurz wie möglich werden. Die am häufigsten verwendeten Variablen erhalten dabei Namen mit nur einem Buchstaben.

5. Programmteil packen
Bei Auswahl dieses Menüpunktes muß der Benutzer den Bereich, der gepackt werden soll, angeben (Anfangs- und Endzeilennummer).

6. Tabelle der Variablen drucken
Werden Option 6 und 4 gewählt, so werden die neuen Variablennamen auf dem an Slot 1 angeschlossenen Drucker ausgedruckt.

Bei der Auswahl der einzelnen Optionen wird man feststellen, daß sich verschiedene Punkte gegenseitig ausschließen. (Im Handbuch wird darauf gesondert eingegangen.) Dies wird jedoch vom Programm automatisch berücksichtigt.
Nachdem alle gewünschten „Packoptionen" gewählt sind, startet man das Programm mit „C". Hat man Option 4 gewählt, so werden nach einer kleinen Pause die veränderten Variablennamen gezeigt.
Nach Beendigung des Packvorgangs erhält man einen Hinweis über die alte und neue Länge des Programms und die Zahl der eingesparten Bytes.
An das Ende eines BASIC-Programms kann man zusätzliche Bytes anhängen (z.B. um ein Maschinenprogramm in das BASIC-Programm einzubetten), indem man den Zeiger für das Programmende verändert. Compact findet solche „Extrabytes" und fragt, ob diese gelöscht werden oder bestehen bleiben sollen.
Sind alle Packarbeiten erledigt, zeigt Compact noch unnötige Zeilen. Wurde ein Programm häufig verändert, so können Programmzeilen im Speicher stehen bleiben, die während eines Programmlaufs nie benutzt werden. Compact gibt die Zeilennummern solcher Zeilen aus, überläßt es aber dem Programmierer bzw. Benutzer, diese Zeilen zu löschen.
In einem zehnseitigen Teil des Handbuches werden die Möglichkeiten von Compact dargestellt, Tips zur richtigen Anwendung gegeben und die Vor- und Nachteile verkürzter Programme beschrieben.

2. COMPARE

Oft kommt es vor, daß man auf seinen Disketten ein Programm in verschiedenen oder gleichen Versionen gespeichert hat. Compare vergleicht zwei Applesoft-Programme und berichtet über gleiche und unterschiedliche Programmzeilen. Außerdem zeigt es auch die Zeilen an, die nur in einem von beiden Programmen auftauchen.
Dazu benutzt Compare folgende Codes:

1 Zeile gibt es nur im ersten Programm
2 Zeile gibt es nur im zweiten Programm
S Zeile ist in beiden Programmen gleich
D Die Zeilennummer ist in beiden Programmen enthalten, die Zeileninhalte sind jedoch unterschiedlich.

Mit Compare ist auch der Vergleich von Text- bzw. Binärfiles möglich. Hier wird jedoch nur darüber informiert, ob die Files identisch sind oder nicht.

Wie Compact kann Compare nach dem Laden mit „&" erneut gestartet werden. Dies bedeutet aber auch, daß nur eines dieser Programme im Speicher gehalten werden kann. Die Koexistenz der dritten DCode-Utility ist jedoch mit jeweils einem der beiden Programme möglich.

3. D.BUG

D.Bug ist wohl das wichtigste Werkzeug der DCode-Diskette. Es stattet den Apple mit 11 neuen Befehlen aus, die wie jeder andere Applesoft-Befehl aufgerufen werden.
Nach dem Laden des Maschinenprogramms mit „BRUN D.BUG" werden die neuen Befehle installiert. Sie stehen damit dem Programmierer zum Programmieren und zur „Entwanzung" (debugging) zur Verfügung. D.Bug belegt ca. 5.5K Speicher und kann gleichzeitig mit „GPLE" und „Double Take" sowie „Compact" oder „Compare" arbeiten.
Im Handbuch wird auf 18 Seiten die Benutzung des Programms und der neuen Befehle beschrieben und anhand von Beispielen erklärt. Hier folgt eine kurze Aufzählung der neuen Kommandos und ihrer Wirkungen.

– FAST FINDER dient zum Suchen (und Finden) „irgendwelcher" einzugebender Zeichenfolgen. Nach der Eingabe von „F" verlangt das Programm ein Suchkriterium. Die Eingabe von Wildcardsymbolen („§" oder „@" für ein, „#" für mehrere Zeichen) ist möglich. „Find" zeigt dann jede Zeilennummer, die die gesuchte Zeichenfolge enthält. Die „FIND AND LIST"-Funktion bietet eine zweite Möglichkeit, nach beliebigen Zeichen zu suchen. Sie wird mit „FL" aufgerufen und zeigt die betreffende Zeile, in der die gesuchte Zeichenfolge zur besseren Kennzeichnung invers ausgegeben wird.

– PROGRAM CHECKER dient zum Überprüfen eines bestehenden Programms. Durch Eingabe von „C" wird das gespeicherte Applesoft-Programm untersucht. D.Bug findet dabei Syntax-Fehler (Anzeige des Interpreters: ?SYNTAX ERROR) wie HME (statt HOME) und Zuweisungsfehler (?TYPE MISMATCH ERROR) wie A = "TEST" statt A$ = "TEST". Angezeigt werden Syntax-Fehler mit einem inversen <?> und Zuweisungsfehler mit inversem <#> nahe der Fehlerstelle. Falsche Wörter innerhalb von Anführungszeichen sowie in REM- und DATA-Statements kann das Programm natürlich nicht finden. D.Bug checkt nicht nur fertige Programme, sondern überprüft auch jede Eingabe von der Tastatur und zeigt Fehler sofort an. Ältere Apple-Besitzer erinnern sich hier an Integer-BASIC. Im Direkt-Modus kann CHECK sogar zeitweise abgeschaltet werden.

– EASY LISTER. Nach Eingabe von „L" wird das im Speicher befindliche Applesoft-Programm gelistet. Die Interpreter-Syntax – L10,100 oder L10 usw. – bleibt dabei bestehen.

– WINDOW TRACER stellt vier neue Kommandos zur Verfügung: Trace, Notrace, Variables und Window. Die Anfangsbuchstaben können als Kürzel für die neuen Befehle benutzt werden.
Die Eingabe von „Trace" oder „T" vor dem Start eines Programms stellt im unteren Bereich des Text-Bildschirms ein Fenster zur Verfügung, in dem nicht nur die Zeilennummern, sondern auch die jeweils durchgeführten Befehle einer Programmzeile angezeigt werden. Auf Wunsch werden sogar verschiedene Variablen und deren Werte angezeigt.
Mit „V:" werden die Variablen definiert, die während des Programmlaufs dokumentiert werden sollen. Nach dem Doppelpunkt müssen natürlich die entsprechenden Namen angegeben werden.
„Window" oder „W" erlaubt dem Benutzer, die Größe des Fensters (s. Trace) zu definieren.
„Notrace" oder „N" schaltet die Trace-Funktion ab.

Während das Benutzer-Programm läuft, kann mit verschiedenen Tasten die Größe des Fensters geändert und die Geschwindigkeit der Trace-Ausgabe gesteigert oder verringert werden. Mit der Leertaste kann das Programm in Einzelschritten durchlaufen werden, und mit den Apfel-Tasten bzw. Paddle-Knöpfen ist ein Ab- und Anschalten der Trace-Funktion möglich.

– DUMP TRACER dokumentiert ein Programm, nachdem dieses angehalten oder aufgrund einer Fehlermeldung abgebrochen wurde. Dazu stehen folgende Befehle bereit:

„Dump" (D) zeigt die im Dump-Puffer gespeicherten, zuletzt ausgeführten Befehle. Die Größe des Dump-Puffers kann mit „Size" (S) festgelegt werden. Maximal kann der Puffer über 10 000 Befehle halten, die minimale Befehlsanzahl ist 5. Mit „Zap" (Z) wird der Puffer gelöscht. Bei Benutzung des Dump-Befehls kann mit verschiedenen Tasten die Ausgabe der gespeicherten Daten gesteuert werden: Links- und Rechtspfeil kontrollieren die Anzeigerichtung, Ctrl-S unterbricht die Anzeige und die Leertaste zeigt jeden Befehl einzeln an. Mit Ctrl-C wird der Dump unterbrochen.

– BREAKPOINTS (Unterbrecherpunkte) können mit D.Bug ebenfalls benutzt werden. Will man beim Programmtest wissen, wann oder wie eine Variable einen bestimmten Wert erreicht hat oder wann ein bestimmter Befehl ausgeführt wird usw., können Breakpoints sehr nützlich sein. Mit D.Bug besteht die Möglichkeit, bis zu acht Breakpoints zu programmieren. Dabei ist es möglich zu testen, ob eine Variable einen bestimmten Wert erreicht hat, ein bestimmtes Kommando ausgeführt wurde und/oder eine Zeile zum x-ten Male durchgeführt wurde. Die Anzeige einer Liste der gesetzten Breakpoints ist ebenso möglich wie das Entfernen der Unterbrecherpunkte. Die Breakpoints können zeitweise abgeschaltet werden. Reaktivierung ist natürlich auch möglich.

Fazit

Zusammenfassend kann man sagen, daß DCode eine Programmsammlung ist, die für den Applesoft-Programmierer ein mächtiges und sehr nützliches Hilfspaket zur Verfügung stellt. Der Preis von etwa DM 160.– hält sich in Grenzen und ist für meine Begriffe gerechtfertigt.

Kumuliertes Inhaltsverzeichnis

Peeker 1/1984 bis 9/1986

Das kumulierte Register aller bisherigen Peeker-Beiträge mußte mit Heft 9/1986 abgeschlossen werden, weil die Datei für den „Wortsucher" sonst zu groß geworden wäre. Die Hefte 10 bis 12/1986 werden als gesonderte Datei auf der Sammeldiskette für das Dezember-Heft nachgeliefert. Die Beiträge sind in Groß- und Kleinbeiträge und entsprechende Rubriken sowie innerhalb der Rubriken nach dem Alphabet sortiert worden. Mit dem „Wortsucher" (s. gesonderter Aufsatz) können Sie darüber hinaus nach Heften, Autoren sowie beliebigen Stichwörtern suchen. Auf der Sammeldiskette wird jeder Beitrag mit einem zweistelligen Rubrik-Kürzel (AP usw.) eingeleitet, so daß man auch nach Rubriken suchen kann.
us

1. Großbeiträge

AP = Applesoft
AS = Assembler
CP = CP/M
DO = DOS 3.3
DR = Drucker
GL = Grundlagen
GR = Grafik
HA = Hardware
HO = Hobby
KY = Kyan-Pascal
LO = Logo
MA = Macintosh
MB = MBASIC
PD = ProDOS
PX = Praxis
RE = Recht
SC = Schule
TE = Technik
TK = Telekommunikation
TU = Turbo-Pascal
UC = UCSD-Pascal
UT = Utilities

2. Kleinbeiträge

BU = Bücher
HP = Händlerprofil
KU = Kurzberichte
LE = Leserbriefe
NA = Nachträge
PR = Produkte
TB = Testberichte
VE = Vermischtes

1. Großbeiträge

AP: Applesoft
– Ampersand macht's möglich. INSTRING-Befehl in Applesoft. Dr. Jürgen B. Kehrel. 2/84/40
– Applesoft-Erweiterungen. Eine Sammlung von Ampersand-Routinen. Markus Enzinger. 6/85/43
– Applesoft simuliert Turtle Graphic. Harald Grumser. (Ergänzung 2/84/10) 1/84/26
– Behandlung reeller Zahlen in Applesoft. Konrad Steinmeier. 3/86/12
– Buchkalkulation. Ein Applesoft-Kalkulationsprogramm für Verlage. us. 9/86/56
– COLUMN80.DEMO, VTAB-HTAB-Demo. us. 5/85/69
– Funktionenprogramm für Apple II+. Stefan Hubertus Weil. 9/86/62
– Haushaltsverwaltung auf dem Apple IIc/IIe. Thilo Schön. 4/86/42
– IF-THEN-ELSE-Befehl simuliert in Applesoft. Franz-Josef Hüskens. 2/84/48
– INALL, INPUT für DOS/ProDOS, (Berichtigung 5/85/50) 4/85/70.
– INPUT für alle Fälle. Tastatur – DOS – ProDOS. Harald Grumser. 5/85/50
– Komfortabler INPUT von Strings. Ulrich Kußmann. 12/85/20
– Menü-Generator für den Apple II. Dr. H. Kersten. 4/85/20
– Müllabfuhr wie ein Blitz. Eine ultraschnelle Garbage-Collection-Routine. Harald Grumser. (Berichtigung 8/85/70) 1-2/85/32
– Print-Using. Rechtsbündige Zahlenformatierung. us. 11/85/49
– RANDOM.DEMO. Random-Zahlen-Generator. us. 5/85/69
– Referenztest. Zeilenverweise in Applesoft-Programmen. Ludger T. Engbert. 11/85/55
– SCREEN80.SAVER. us 4/85/76
– Tabelle der wichtigen Applesoft-Interpreter-Routinen. Harald Grumser. 12/85/30
– Umwandlung in Großbuchstaben. Harald Grumser. 9/85/31
– Verkettung von Applesoft-Programmen. Variablenübergabe unter DOS 3.3. Frank Bühler. 7/86/18
– Wie man Arrays löscht. Christiane Breit und Thomas Schneyer. 2/84/51

AS: Assembler
– Assembler-Pseudo-Opcode-Referenztabelle. Dr. Jürgen B. Kehrel. 8/85/51
– Bildschirmmasken in der Language Card. Carl Frieder Mahr. 2/86/41
– 65C02-Disassembler für Apple IIe und II Plus. us. 1/84/14
– Disassembler für „geheime" 6502-Opcodes. Christoph Bregler. 5/85/31
– Fakultäten. Roland und Manfred Fietkau. 8/85/56
– Flag Monitor. Eine Taktzähler-Utility. Michael G. Schneider. 9/85/32
– Gemeinsame F8-Monitor-Routinen. Apple II+/e/c/enhanced. us. 5/86/52
– Hex-Dez-Konvertierung für 32-Bit-Zahlen. Harald Grumser. 7/85/69
– 68000-Kompaktkurs. Mit umfangreichen Befehlstabellen. Pit Capitain. 9/86/6
– 6502 leicht gemacht. us. 7/85/33
– Makros für die neuen Befehle des 65C02. Dipl.-Oec. Edgar Meyzis. 4/85/30 (Berichtigung 5/86/74)
– Speichern Sie den Bildschirm ab! Organisation des 40-Z/Z-Apple- und des 80-Z/Z-Videx-Bildschirms. Dr. Jürgen B. Kehrel. 1-2/85/42
– Tabelle der Interpreter- und Monitor-Adressen. 5/85/72
– Vorlesestunde. Apple und S.A.M., ein hilfreiches Gespann. Dr. Jürgen B. Kehrel. 7/85/71
– Z80-Assembler-Programmierung unter CP/M. Dr. H. Kersten. 11/85/33

CP: CP/M
– Adreßverwaltung mit dBase II. Listing des überarbeiteten Quelltextes. Ing. (grad.) Ernst Fischer. 6/86/52
– Adreßverwaltungsprogramm mit dBASE II. Ing. (grad.) Ernst Fischer. (Ergänzung 10/85/21) 8/85/40
– Apple-CP/M. Mit einem RAM-Disk-Driver für die 64K-Karte. Karl-Walter Bott. 6/85/55
– CP/M für Einsteiger. Jörg Lange. 4/85/59
– Das CP/M-Directory näher beleuchtet. Der Aufbau von 16-Sektor-CP/M-Disketten. Dr. Jürgen B. Kehrel. 5/85/39
– GETCPM: Konvertierung von CP/M- in DOS-Textfiles. Applesoft-Version. Thomas Fink. 3/85/43
– Weitere Wordstar-Modifikationen. Patches für Epson FX-80 mit DDT. Dieter Conrad. 3/86/56
– Wordstar druckt internationale Zeichensätze. Frei definierte Sonderzeichen auf dem FX-80 nutzen. Dipl.-Ing. H.A. Rohrbacher. 8/85/45
– Wordstar mit allen FX-80-Schriftarten. Dipl.-Ing. H.A. Rohrbacher. 7/85/57
– Wordstar-Transfer-Refiner. Konvertierung von CP/M-Wordstar- in DOS-Applewriter-Textfiles. Dr. Jürgen B. Kehrel. 3/85/35

DO: DOS 3.3
– Der File-Manager des DOS 3.3. Harald Grumser. 5/85/6
– Disk-Chirurg. Ein Programm zur Überprüfung schadhafter Diskettensektoren. Wolf-J. Faust. 11/85/24
– DOS-Mover. Erhöhung der Speicherkapazität um über 30%. Harald Grumser. 2/86/17 (Leserergänzung 5/86/73)
– Ein neues FID für DOS 3.3. Mit Auto-FID für RAM-Disk-Besitzer. Harald Grumser. 9/85/12
– Poor Man's RAM-Disk. RAM-Disk-Driver für die Language Card. us. (Ergänzung 6/85/69) 1-2/85/8

DR: Drucker
– Großformatiger HGR-Ausdruck. Mark Liebrand. 11/85/20
– Hardcopy auf Olympia ESW 100 RO. Ein Typenraddrucker wird grafikfähig. Ralf Menssen. 6/85/34
– Hires-Grafik-Dump für Epsondrucker. Jürgen und Dieter Geiß. 1/84/40
– Imagewriter kurz und bündig. Die wichtigsten Steuerbefehle. Thorsten Schunk. 12/85/8
– Peeker-Quickie. DUMP80REL.
– SCREEN80. 80-Z/Z-Bildschirm-Dump. us. 4/85/33
– Sonderzeichen auf dem Imagewriter. Mit BITEDITOR- und ANIMATRIX-Zeichensätzen. Carl Frieder Mahr. 4/86/32

GL: Grundlagen
– Binäres Rechnen mit Papier und Bleistift. Teil 1: Addition und Subtraktion. us. 2/86/6
– Binäres Rechnen mit Papier und Bleistift. Teil 2: Multiplikation und Division. us. 4/86/6
– Das Wahre ist eins. Aussagenlogik für Programmierer. us. 8/86/32
– Wie funktioniert Quicksort? Mit einem Demonstrationsprogramm. us. 1/86/6 (Berichtigung 5/86/73)

GR: Grafik
– Applesoft-HGR-Erweiterung in Assembler. Klaus Schäfer. 1/86/30 (Berichtigung zu AGE 5/86/73)

– Applesoft-Interpreter-Erweiterungen für Double-Hires. Dr. Wolfgang Braun. 10/85/14
– ASCII-Text im HGR-Bildschirm. Mit einem Zeichengenerator-Programm. Markus Geltenpoth. 12/85/16
– CONVERT560. Konvertierungsroutine für Double-Hires-Drucker-Dump. Marc van Woerkom. 5/85/25
– Das Farbbit. Pseudo-Double-Hires auf dem Apple II Plus. H.-D. Siewert. 6/85/16
– Double-Lores-Applesoft-Erweiterungen. Karl-Walter Bott. 8/85/32
– Dreidimensionale Funktionsdarstellung. Alfred Böhm. 2/86/43
– Graf-quattro. Künstlerische Grafik für jedermann. Teil 1: Wie ein Supereditor entsteht. Nino G. Barbieri. 4/85/14
– Graf-quattro. Teil 4: Der Grafik-Editor. Nino G. Barbieri. 10/85/6
– Graf-quattro. Teil 2: Schneller als der Schall. Nino G. Barbieri. (Ergänzung 10/85/69) 6/85/6
– Graf-quattro. Teil 3: Tricks über Tricks. Nino G. Barbieri. 8/85/24
– PLOT 3.E. Ein neuer Funktionsplotter für den Apple IIe/c mit Double-Hires-Grafik. Hans-Martin Eng. 6/86/6
– PLOT 2.0. Ein komfortabler Funktionsplotter. Hans-Martin Eng. (Berichtigung 9/85/69) 5/85/17.
– Superdump. Das universelle Hires-Grafik-Dump-Programm. Jürgen Geiß. (Ergänzung 8/85/50) 6/85/22
– Trickfilmgrafik für Spielprogramme. Mit einem Sprite-Editor. Bernd Klawonn. 11/85/6 (Berichtigung 5/86/73)
– Wie man die Grafik verdoppelt. Teil 2: Double Hires. Karl-Walter Bott. 2/84/24. Vgl. 1/84/32
– Wie man die Grafik verdoppelt. Teil 1: Double Lores. us. 1/84/32

HA: Hardware
– Accelerator IIe. Die neue Superkarte. us. 1/84/19
– Accelerator-Karte abstellen. us. 3/85/66
– Aufbau und Funktion von Diskettensystemen. Dipl.-Ing. Gerhard Berg. 3/85/10
– Balfer-Interface. RAM-Disk-Driver für Apple IIe. us. 5/85/12
– Der neue 65C02-Prozessor. us. 1/84/6
– Die neuen ROMs für den Apple IIe. Teil 3: Applesoft-Interpreter. Harald Grumser. 10/85/34
– Die neuen ROMs für den Apple IIe. Teil 1: Grundlagen. Teil 2: Standardein- und -ausgabe. 10/85/22 us
– EPROM-Programmiergerät für den Apple II. Dr. Roland Schule. 10/85/40 (Leserergänzung 5/86/73)
– Konfiguration der seriellen Schnittstellen beim IIc. Enno Klatt. 2/86/34
– Netzwerk für Apple-Rechner. Low-Cost-Rechner-Kopplung für jedermann. Alexander Niemeyer. 8/86/50
– ProDOS-Uhrkartentreiber. Realtime-Clock „TIME II". Dipl.-Ing. Achim Sukker. 7/86/22
– Prometric. Ein Apple-Kompatibler mit 65C02C. us. 2/84/64
– Super-HGR für NEC- und ITOH-Drukker. Rainer Hammerschmidt. 9/85/30
– Wie man Programme „entschützt". Ein „Hardbreaker" für Apple II+ und IIe. Willi Porten. 9/86/34

HO: Hobby
– Abenteuer in Green-Sky. Adventure-Spiel Below the Root. Thomas Bühner. 8/86/67
– Cosmo-Crumble. Ein Weltraum-Action-Spiel. Oliver Steinmeier und Alexander Niemeyer. 12/85/6
– Die Berechnung der Kreiszahl Pi. Pi mit bis zu 15.500 Stellen Genauigkeit. Roland und Manfred Fietkau. 5/86/42
– Die Gewinner des Primzahlen-Wettbewerbs. 2/84/70
– Ein Tag im Stellwerk. Spiel Train Dispatcher. Thomas Bühner. 8/86/68
– Fehlerjagd im Computerlabor. Spiel I.C. Silver. Thomas Bühner. 8/86/69
– Geschwindigkeit ist keine Hexerei. Erastosthenes erneut betrachtet. Michael G. Schneider. 3/85/56
– Grafik-Demonstrationen. Ralf Knoke. 8/85/67
– Grafik-Demonstrationen. Teil 2. Ralf Knoke. 1/86/61
– Mousory. Memory-Spiel mit der Apple-Maus. Wolfgang Landgraf. 7/86/12
– Pascal-Preisausschreiben. 7/85/62
– Pascal-Preisausschreiben. Die Gewinner. 9/85/58
– Puzzle mit der Maus. Joachim Mette. 10/85/65
– Pyramid Pitty. Ein Reaktionsspiel. Michael Matzat. 7/85/6
– SOLITAIRE und wie man es löst. Stefan Maas. 6/85/64
– Was ist Logo? Eine Kurzeinführung. Kurt Rudl. 10/85/64
– Wer ist der Schnellste? Primzahlen-Preisausschreiben. 1/84/58
– Zeichenjagd. Ein Einzeiler. Hans-Peter Lendle. 8/85/69
– Zweistimmige Melodien. Ein intelligentes Tonprogramm für den Apple II. Jörg Schmidt. 4/86/60

KY: Kyan-Pascal
– Kyan Pascal – Aufbaukurs. Datentypen, Unterprogramme, Strings, Dateien. us. 3/86/32
– Kyan-Pascal. Ein vollcompilierendes 6502-Pascal. Herbert Pohl. 1/86/50
– Kyan-Pascal. Grundkurs. us. 2/86/56
– Kyan-Pascal 2.0. Tips und Tricks. us. 6/86/47
– Kyan-Pascal und Assembler. us. 4/86/48
– Stringbefehle für Kyan-Pascal. Matthias Meyer. 9/86/50

LO: Logo
– Einblicke in Logo. Apple Logo II. Dipl.-Päd. Ernst F Haas. 8/86/22

MA: Macintosh
– Apple-II-Monitor für den Mac. Wie zu Wozniaks Zeiten. Pit Capitain. 5/85/41
– Datenübertragungsraten beim Macintosh. Arne Schirmacher. 7/86/54
– Der neue Macintosh. Erste Eindrücke zum Mac Plus. us. 3/86/6
– Die Maske fällt und Mecki seh' ich nun, entblößt von allen Reizen. Der Macintosh im Leistungstest. us. 3/85/44
– Handmac. Ein Macintosh-Paket für den Handwerker. Harald Grumser. 5/86/55
– Ikonen und Deixis oder die Philosophie des Macintosh. us. 1-2/85/74
– Microsoft Basic leicht geMACht. Teil 3: Die Ein- und Ausgabe. Pit Capitain. 4/85/63
– Microsoft Basic leicht geMACht. Teil 4: Die Grafik. Pit Captain. 8/85/60
– Microsoft Basic leicht geMACht. Teil 2: Funktionen und Programmsteuerbefehle. Pit Capitain. 1-2/85/78
– Microsoft Basic leicht geMACht. Teil 1: Operanden und Operatoren. Pit Capitain. 2/84/54
– MS-File, MS-Word und Imagewriter II. Anspruch und Wirklichkeit im Büroeinsatz. Dr. Martin Wolter. 5/86/57

MB: MBASIC
– Lohn- und Einkommensteuer 1984. Ein umfassendes CP/M-MBASIC-Programm. Volker Duske. 1-2/85/45
– MBASIC für den Applesoft-Profi. Jörg Lange. 9/85/48

PD: ProDOS
– Applesoft-Editor-Macros. Ein Quickie für den PRODOS.EDITOR. us. 1-2/85/86
– Block-Tracer für ProDOS. us. 5/85/51
– Edit. Ein Luxus-Disk-Editor für ProDOS. Teil 1: Befehlsauswertung und Diskettentreiber. Arne Schäpers. 5/86/8
– EDIT. Ein Luxus-Disk-Editor für ProDOS. Teil 3: Die Makrosprache. Arne Schäpers. 8/86/40
– EDIT. Ein Luxus-Disk-Editor für ProDOS. Teil 2: Dumpen und Editieren. Arne Schäpers. 7/86/24
– EDIT. Übungen zum Disk-Editor. 6/86/24
– FORMAT-Befehl für ProDOS. us. 5/85/56
– MAKESUB. Erstellung zusammenhängender Subdirectories. Arne Schäpers. 3/86/20
– Megaboard- und AFDC-Massenspeicher. Mit ProDOS-Backup-Programmen für die MDB-Festplatte und das Erphi-Subsystem. us. 6/86/30
– Patch für ProDOS.INIT 80 Spuren. Hans-Georg Hüneke. 1/84/56
– ProDOS-Anpassung für Laufwerke mit 40 bis 160 Spuren. Horst Hanke und Hans-Georg Hüneke. 11/85/29
– ProDOS-Fastboot oder wie man ProDOS in 6 Sekunden bootet. Arne Schäpers. Nach einer Idee von us. 9/85/20
– ProDOS für Anfänger. Teil 2: Das definitorische Grundgerüst. us. 4/85/41
– ProDOS für Anfänger. Teil 3: Geheimnisse von BSAVE und BLOAD. us. 8/85/18
– ProDOS für Anfänger. Teil 1: Was ist ProDOS? us. 2/84/12
– ProDOS-Patch für geänderte F8-ROMs. us. (Ergänzung 2/84/10) 1/84/54
– PRODOS.READER. 4/85/46
– ProDOS-Runtime-Library. Eine Muster-Bibliothek für Assembler-, Applesoft- und Pascal-Programme. us. 7/86/40
– ProDOS. Theorie und Praxis. us. 6/85/52
– ProDOS und „Kompatible". Ein allgemeines Patch-Programm. Arne Schäpers. (Berichtigung in Leserbrief 10/85/69) 1-2/85/31
– PROTODOS. Konvertierung von ProDOS- in DOS-3.3-Dateien. Arne Schäpers. 1/86/37
– QUICKCOPY. Ein schnelles ProDOS-Kopierprogramm für 35-80 Spuren. us. 1-2/85/22
– Trace-Bug. 4/85/46
– Was Geschäftsleute vom ProDOS-FILER wissen sollten. us. 2/84/20

PX: Praxis
– Der Apple als Werkzeug für den Betriebswirt. Teil 1: Das Simplex-Verfahren in der Controllerpraxis. Willy Holtkamp. 1/86/56
– Hardware-Kurztip. Anschluß eines Brother HR-15 XL II an den Apple IIc. Hans-Peter Sprauer. 7/86/59
– Matrizenrechnung in der betriebswirtschaftlichen Praxis. Dipl.-Betriebswirt Willy Holtkamp. 2/86/29
– Registerprogramm für Buchregister und zweisprachige Glossare. us. 7/86/6
– Schreiben mit der Maus. Textverarbeitungsprogramm Mousewrite. Thomas Bühner. 7/86/56

RE: Recht
– Ave auctor, plagiarii te salutant. Software und Zitatrecht. us. 9/85/6
– Knock, Knock! Who's there? Software und Kopierrecht. us. 4/85/6
– Rechtsprechung zum Software-Urheberrecht. RA Dieter Naber. 10/85/47
– Urheberrechtsnovelle. us. 9/85/11

SC: Schule
– Höhere Präzision bei den vier Grundrechenarten. Konrad Steinmeier. 3/85/30
– Testgenerator für Legastheniker. Dr.Wolfgang Kersken. 3/85/22
– Vokabeltrainer. Vokabeln lernen mit dem Apple II. Thomas Zink. 1/86/20 (Berichtigung 5/86/74)

TE: Technik
– Bit-Editor. Zeichensatz-EPROMs für die Videx-Karte. Joachim Klamt. 7/85/29
– Der Apple II als persönliches Ingenieurwerkzeug. Dipl.-Ing. Bernd Worms. 3/85/6
– Die AP33-Megawarp-RAM-Karte. Mit einem RAM-Disk-Driver für ProDOS. us. 7/85/8
– Die RAM-Karten von IBS. Mit Quellcode eines RAM-Disk-Drivers. us. 1-2/85/18
– Ein intelligenter Harddisk-Controller. Aufbau und Arbeitsweise des Megaboards. Dr.-Ing. T. Mulica. 4/86/20
– FORMATTER. Ein universelles Formatierungsprogramm. Arne Schäpers. (Ergänzung 9/85/69) 7/85/20
– Fourier-Analyse. Peter Walber. (Berichtigung 9/85/69) 6/85/38
– Meßwertverarbeitung. Praktikumsauswertung mit dem Apple. Volker Herrmann. 3/86/58
– Testprogramm für Apple-II-Diskettensysteme. Analyse von Disketten, Laufwerk und Controller. Dipl.-Ing. Gerhard Berg. 8/85/6

TK: Telekommunikation
– Modems kommen in Mode. Praktische Ratschläge für die Telekommunikation mit Modem. Matthias Pohl. 2/84/36
– Terminal-Programm für den Apple II. Andreas Lecreux. (Ergänzung 6/85/72) 4/85/34
– Weltweite Datenübertragung mit dem WS 2000. Ein Universalmodem für alle Normen. Matthias Pohl. 1-2/85/84

TU: Turbo-Pascal
– CP/M-56K-RAM-Disk unter Turbo-Pascal. Bernd Eichinger-Wieschmann. 12/85/62
– Filer für CP/M. Ein Dateikopierprogramm in Turbo-Pascal. Gerhard Runge. 9/86/40

KURZBERICHTE

– Fußballweltmeisterschaft 1986 in Mexiko. Ein Simulationsprogramm in Turbo-Pascal 3.0. Manfred Stertenbrink. 5/86/6
– READPAS. Konvertierung von UCSD- in Turbo-Pascal-Textfiles. Michael Erperstorfer. 1/86/48 (Berichtigung 5/86/73)
– Turbo-Pascal Schritt für Schritt. Dr. Ekkehard Kaier. 1/84/51

UC: UCSD-Pascal
– Apples Maus lernt jetzt auch Pascal. Jürgen Geiß. 4/85/48
– Designer. Zeichensatz-Generator in Apple-Pascal. Gabor Herr. 1/86/43
– Dreiecksberechnungen in UCSD-Pascal. Samuel Schmid. 9/86/48
– GETDOS: Konvertierung von DOS- in Pascal-Textfiles. Jürgen Geiß. 1-2/85/70
– GETPAS: Konvertierung von Pascal- in DOS-Textfiles. us. 1-2/85/68
– Interaktive Funktionseingabe. Gerhard Röhner. 4/86/58.
– Pascal-Diretory unter der Lupe. Dieter Geiß. 1-2/85/64
– Pascal 1.2. Eine Richtigstellung. Bernard Condrau. 10/85/60
– Pascal-Kompaktkurs für Applesoft-Programmierer. Hinweis für Sammeldisk. us. 10/85/46
– Pascal-Kompaktkurs. UCSD- und Turbo-Pascal. Teil1: Grundlagen für Applesoft-Programmierer. us. 12/85/33
– Pic-Edit. Ein universeller Grafik-Editor in 128K-Apple-Pascal 1.2. Jürgen Geiß. 8/86/6
– RAM-Disk-Driver für Pascal 1.1. Michael Schröter. (Berichtigung 7/85/52) 6/85/48
– SUPERDUMP für Apple-Pascal 1.1 und 1.2. Grafik-Ausdruck für Epson-FX80 und Imagewriter I. Jürgen Geiß. 6/86/40
– Superschnelle Bildschirmausgabe in Apple-Pascal. Dr. Wolfgang Braun. 5/86/29
– Tips und Tricks in Pascal. Teil 5: Assembler-Ein/Ausgabe in UCSD-Pascal. Dieter Geiß. 2/86/47
– Tips und Tricks in Pascal. Teil 7: Der externe Aufbau von Files. Dieter Geiß. 5/86/34
– Tips und Tricks in Pascal. Teil 6: Der interne Aufbau von Files. Dieter Geiß. 3/86/48
– Tips und Tricks in Pascal. Teil 4: Die Compiler-Optionen. Dieter Geiß. 11/85/57
– Tips und Tricks in Pascal. Teil 1: Die versteckte Prozedur „Idsearch" oder wie man Schlüsselwörter halbfett druckt. Dieter Geiß. 8/85/48
– Tips und Tricks in Pascal. Teil 2: Kann man den P-Code optimieren? Dieter Geiß. 9/85/40
– Tips und Tricks in Pascal. Teil 3: P-Code-Cruncher für 128K-Pascal. Dieter Geiß. 10/85/50
– Transzendente Funktionen in Pascal. Dieter Geiß. 5/85/35

UT: Utilities
– Befehlserweiterung durch CHRGET-Manipulation. Dr. Jürgen B. Kehrel. 2/86/38
– DOS-3.3-Backup-Programm für die Megaboard-Festplatte. us. 5/86/61
– Erphi-Patch für Superquick. Jetzt auch 160 Tracks in Windeseile kopiert. Dr. Jürgen B. Kehrel. 5/86/59
– Quicksort. Eine superschnelle Applesoft-Erweiterung. Harald Grumser. 1/86/16

2. Kleinbeiträge

BU: Bücher
– Anwenderprogramme. 4/86/65
– 6502 Anwendungen. 1-2/85/93
– Apple Assembler. 2/84/78
– Apple-Assembler lernen. 1/86/69
– Apple DOS 3.3. 1/84/65
– Apple DOS unter der Lupe. 7/86/60
– Apple Grafik. 5/86/68
– Apple II Basic Handbuch. 4/86/64
– Apple IIc. 5/86/67
– Apple II-DOS 3.3-Assembler-Listing. 5/86/68
– Apple II für Technik und Wissenschaft. 7/86/60
– Apple II für Technik und Wissenschaft. 10/85/67
– Apple II leicht gemacht. 1/84/65
– Apple II ROM Listing. 1-2/85/93
– Apple II Tips & Tricks. 2/84/76
– Apple Maschinensprache. 2/84/78
– Apple ProDOS für Aufsteiger (Band 1). 2/84/78
– Applesoft BASIC. 1/86/69
– Apple-Works. 5/86/68
– Arbeiten mit dem Macintosh. 5/86/67
– 68000 Assembly Language Programming (Kurzrezension). 9/86/24
– Auswahl und Einsatz lokaler Netzwerke. 7/86/61
– Basic: Alles über Peek und Poke. 4/86/64
– Basic: Die perfekte Behandlung von Zeichenketten. 4/86/64
– BASIC für Fortgeschrittene. 7/86/61
– Basic: Mathematik per Computer. 4/86/64
– Basic: Programme für Kaufleute. 4/86/64
– BASIC Übungen für den Apple. 2/84/76
– BASIC-Wegweiser für den Apple II. 5/86/67
– Basic: Zahlen-Umwandlungen. 4/86/64
– Betriebssystem CP/M. 4/86/65
– Bewegte Apple-Grafik. 1/86/69
– Bücher für Elektronik + Computer. 2/84/76
– Computereien. 7/86/61
– Computereien 2. 7/86/61
– Computer für Kinder. 7/86/60
– Computergrafische Experimente mit Pascal. 7/86/62
– Computer im Unterricht. 7/86/63
– Computerspiele und Computergrafik. 7/86/62
– Das Apple Macintosh Buch. 10/85/67
– Das Buch zum Apple II. 1-2/85/93
– Das Buch zum Apple-Writer II/IIe. 7/86/60
– Das Floppybuch zum Apple II. 5/86/67
– Das große BASIC-Lernbuch. 5/86/67
– Das Hacker HACKBUCH. 7/86/61
– Das Macintosh Arbeitsbuch. 5/86/67
– Das Modembuch zur DFÜ. 4/86/64
– Das ProDOS-Handbuch. 5/86/67
– Datenmanagement mit dem Apple II. 7/86/60
– dBase II. 1/86/69
– DOS 3.3 – das Diskettenbetriebssystem des Apple II. 4/86/65
– EDV – kein Geheimnis. 7/86/61
– Einführung in die Apple UCSD-Pascal-Systeme. 7/86/60
– Fortgeschrittene 6502 Programmierung. 1-2/85/94
– FORTH ganz einfach. 7/86/62
– Grafik- und Elektronikprogramme für den Apple II. 7/86/60
– Grundkentnisse Pascal. 4/86/64
– ISIS Personal Computer Report. 4/86/64
– Lerne BASIC auf dem Apple. 10/85/67
– Macintosh. 10/85/67
– Macintosh (Anwenderhandbuch). 10/85/67
– Macintosh Benutzerhandbuch. 1-2/85/94
– Mathematik auf dem Apple II, IIe, IIc. 10/85/67
– Mathematische + Wissenschaftliche Programme in Basic. 1/84/65
– Mein BASIC-Manual, Teil 1. 5/86/67
– Mikrowissen A-Z. 7/86/61
– Modelle der Wirklichkeit. 7/86/63
– Now That You Know Apple Assembly Language: What Can You Do With It?. 5/86/67
– Pandasoft-Katalog. 2/84/78
– ProDOS-Analyse. 1/86/69
– Programme in Microsoft-BASIC. 5/86/68
– Programme zur Wahrscheinlichkeitsrechnung und Statistik. 5/86/67
– Programmieren in FORTRAN 77. 7/86/62
– Programmierkurs mit Microsoft-BASIC. 7/86/62
– Programmiersprache BASIC – Schritt für Schritt. 7/86/61
– Programmierung des 68000. 4/86/64
– Programmierung des 6502. 1-2/85/93
– Programmierung des 68000 (Kurzrezension). 9/86/24
– Programmierung des M68000 (Kurzrezension). 9/86/24
– Rechnerunterstütztes Konstruieren (CAD). 7/86/62
– Schulsoftware-Katalog. 2/84/76
– Schulsoftware-Katalog Apple. 7/86/60
– Softwareführer 1986. 4/86/64
– Spielend Programmieren lernen. 7/86/61
– Spielprogramme für den Apple IIe. 5/86/67
– Statistik in Basic. 1/84/65
– SYBEX Mailbox Führer. 7/86/62
– Technisch-naturwissenschaftlicher Pascal-Trainer. 7/86/61
– The Apple in your hand. 2/84/76
– The Custom Apple & Other Mysteries. 7/86/61
– Trainingsbuch zu APPLESOFT-BASIC. 10/85/67
– Was ist wo im Apple. 1/86/69

HP: Händlerprofil
– HIB. Der Computerladen in Nürnberg. 9/85/59
– Orgasoft plant für die Zukunft. 4/85/67
– r + r electronik. 2/84/69

KU: Kurzberichte
– Analytisches Inhaltsverzeichnis der Peeker-Sammeldisketten #1 bis #18. 6/86/64
– An Ear to the Ground. us
– Apple-Bücher im Preisvergleich. 4/85/70
– Apple dementiert „Heute Journal". 10/85/76
– Apple-Geschichte. 9/85/64
– Apple GmbH erhöht Stammkapital. 10/85/76
– Apple-Händler-Netz umstrukturiert. 10/85/76
– Apple IIe-Emulation für Apple III. 9/85/64
– Apple-Kompatible. Hans-Kurt Kuhn. 5/85/66
– Apple-Kompatibler MK II. 9/85/65
– Apple-Paketpreise. 9/85/64
– Applepreise = Mondpreise? us. 8/85/67
– Apple-Schul-Programm AULA. 9/85/65
– Appletalk für Macintosh. 9/85/65
– Auf dem Abstellgleis: Steve Wozniak. 4/85/68
– AUGE und Apple-München. 4/85/68
– BTX für Apple IIc. 10/85/77
– Buchhaltungsprogramm BUCH. 10/85/75
– Design-Preise für Apple IIc. 9/85/65
– Digitalisiertabletts höchster Präzision. 10/85/75
– Druckertisch. 9/85/66
– Ergänzungen zu „ProDOS für Aufsteiger", Bd. 1. Arne Schäpers. 4/85/69
– Erno Unibox. 9/85/65
– Flugsimulator mit deutscher Anleitung. 10/85/75
– 4.000 Frei-Programme. 10/85/75
– Hard- und Software zum Apple IIc. Mathias Pohl. 1/86/67
– Incircuit-Emulator. 9/85/64
– 10 Jahre Apple Computer. Ein Rückblick. Dr. Jürgen B. Kehrel. 5/86/64
– 640K-Drive für IIc. 9/85/64
– 512K für Apple III. 9/85/64
– Kleine Mikrocomputer-Chronik. us. 12/85/64
– Kopierschutz für den Apple II. Matthias Greve. 5/85/61
– Kyan-Club-Nachrichten. 5/86/60
– Kyan-Pascal. 1/86/29
– Laserwriter von Apple. 9/85/65
– Leasing von Apple-Computern. 10/85/76
– Lisa/Mac-Screenswitcher. 9/85/64
– Mac-Leasing für Studenten: 10/85/76
– Macpack. 9/85/64
– Mac-Pressekonferenz bei Apple in München. Harald Grumser. 10/85/77
– Macprint-Satzsystem. 10/85/76
– Macprolog2-Interpreter. 10/85/76
– Mactablet. 9/85/64
– Maus: Mac an Hochschulen. 10/85/76
– Melco-Druckpuffer von Watanabe. 10/85/75
– 20M-Festplatte für Mac. 9/85/65
– Neologismen in der Computersprache. us. 5/85/60
– Neuer Apple-Kompatibler. 10/85/75
– No Orchids für Miss Lisa. Lisa und die Folgen. us. 7/85/67
– Notstromaggregat. 9/85/66
– OKI-Drucker für Apple II. 10/85/76
– Plazierung von Apple-Computern. 2/86/68
– Präsentationsgrafiken mit (G)RO-GRAF. 10/85/75
– Pressestimmen zur Wirtschaftslage von Apple. 9/85/69
– ProDOS-Bücher und -Aufsätze 4/85/69
– Siemens entdeckt neue Primzahl. 8/85/74
– Sound Sampling System. 10/85/76
– Spooler für Epson-Drucker. 9/85/65
– Teletex für den Apple II. 10/85/77

– Transfer II Terminalprogramm. 9/85/65
– Turbo-Lader Graph. 10/85/76
– Wie gibt man Maschinenprogramme ein? 6/85/70

LE: Leserbriefe
– 2/84/8, 4/85/72, 5/85/70, 6/85/69, 5/86/72, 2/86/64, 9/85/66, 7/85/73, 3/85/77, 10/85/68
– Dieter Geiß beantwortet UCSD-Pascal-Leserbriefe. 5/86/69
– Jörg Bliesener beantwortet CP/M- und Wordstar-Leserbriefe. 6/86/68
– Pascal-Leserbriefe. 12/85/68

NA: Nachträge
– Diversi-DOS 2-C. 8/85/68
– Druckfehler im Programm „Designer". 2/86/65
– Formatter. 9/85/69
– Fourier-Analyse. 9/85/69
– Korrekturen und Errata zu früheren Peeker-Heften. 5/86/73
– Nachtrag zum Pascal-RAM-Disk-Driver. 7/85/52
– PLOT 2.0. 9/85/69
– Schwierigkeiten mit den Graf-quattro-Cursoren. Nino G. Barbieri. 10/85/63
– SUPERDUMP und Apple IIc. 8/85/50
– Terminal-Programm für die Super-Serial-Card. Andreas Lecreux. 6/85/72
– Tips zur Konvertierung der Adreßverwaltungs-Dateien von DOS 3.3 nach CP/M. 10/85/21

PR: Produktberichte
– A/D-Wandlerkarte Softcard II. 5/86/77
– AFC-Tastatur mit Maus für Apple. 1/84/63
– Aktienanalyse mit Stockmaster. 9/86/69
– Akustikkoppler AK 300. 4/85/77
– Akustikkoppler dataphon s-21-d. 4/85/77
– Apple-Assembler. 9/86/69
– Atari-Joystick und Apple. 2/84/75
– BASF-Reinigungs- und Justagedisketten. 6/86/68
– Beliebte Apple-Spiele. 3/85/73
– Bidirektionaler Datenverkehr zwischen Apple und Brother-Typenradschreibmaschine. 1/84/64
– Bildschirm-Dump auf Tastendruck. Fingerprint Plus. Thomas Bühner. 4/86/66
– Branchenprogramm Reisebüro. 1-2/85/97
– 65C02 + Accelerator: Wer liefert was? 1/84/64
– CAD-System SYNCAD. 3/86/69
– Champion-Drucker-Karte. 5/86/76
– Computerunterstütztes Lernen. 1-2/85/99
– Copy-Killer jetzt in deutsch. 8/85/76
– Copy Killer (Kopierschutzsystem). 1/84/64
– CP/M für Apple IIe. 9/86/70
– CP/M-Karte für Apple IIc. Harald Grumser. 4/86/69
– CSW-EPROM-Karte für Apple II. 3/86/67
– CTS-Mineralöl. 2/84/75
– Das Bildverarbeitungssystem MAGIC. 8/85/74
– Daten-Diagnosesystem. 3/86/68
– Die bessere Maus. Maus für Apple-II-Rechner. Thomas Bühner. 4/86/67
– Diskettenkasten für Hängeregistratur. 3/86/66
– Disketten von Mitsubishi. 3/86/66
– Doppel-Parallelinterface mit RAM oder EPROM. 1-2/85/96
– Drucker-Treiber. 5/86/76
– DTACK-68000-Board. Ein Erfahrungsbericht. Wolfgang Trier. 6/86/58
– Ein Apple macht sich fein. Staubschutz-Abdeckungen. Thomas Bühner. 4/86/66
– Einbau der CP/M-3.0-Karte in den Apple IIc. 2/86/63
– Ein Heim für die Maus. Mouse Trap. Thomas Bühner. 8/86/61
– Erphi FSS 280. 2/86/67
– Europrint FT 80 X. 5/86/76
– Fakturierungsprogramm Faktufix. 5/86/76
– Flipper-Karte. 5/86/76
– Fußball-Bundesliga. 9/86/69
– Gepard-Computer. 5/86/78
– Grafiktablett für Mac. 5/86/76
– Grafik zu Papier bringen. Hires-Grafik-Druckprogramme. Imagewriter II, Printographer und Zoom Grafix. Thomas Bühner. 8/86/63
– High Fidelity Adaptor. Thomas Bühner. 3/86/61
– Imagewriter II und Unidisk im Kurztest. us. 8/86/56
– Individuelle Zeichensätze für Matrixdrucker (DMP-Charger). 1/84/63
– INTERTEXT, Internationale Textautomation mit dem Apple IIe. 4/85/76
– Jazz 1A für Mac Plus. 5/86/76
– Joystick einfacher anschließen. Game-Socket-Extender. Thomas Bühner. 4/86/68
– Juki-Typenraddrucker. Juki 6500. Juki 6300T. 9/86/69
– Keycom 6. Eine externe Tastatur für den Apple IIe. 7/86/65
– 64K-Karte für IIe. 1-2/85/96
– Kyan-Pascal. 12/85/77
– Laser 128. Ein Apple-IIc-Kompatibler. Hans-Martin Eng. 7/86/64
– Lohn- und Gehaltsabrechnung. 1-2/85/97
– Lotus – Jazz, ein multifunktionales Software-Paket. 1-2/85/97
– MacLabel. 2/84/75
– Mailmerger für AppleWorks. 1-2/85/99
– Microfloppy mit 2 x 1 MByte. 8/85/76
– Microline-Matrixdrucker. Microline 292/293. Microline 294. 9/86/70
– Microline 192 von Okidata. Harald Grumser. 3/86/64
– Mockingboard C. Roland Maisch. 3/86/62
– Modula-2 für den Macintosh. 1-2/85/96
– NEC-Matrixdrucker P6/P7. 9/86/70
– Optokoppler-Karte. 3/86/66
– OSP-2-Matrixdrucker. 3/86/67
– PAL-Programmer für Apple. 4/86/69
– Parallelportkarte für Apple II. 4/85/77
– PIA-Karte für Apple II. 5/86/77
– Plotter MP 2000. 3/86/69
– Plusworks, Appleworks auf dem Apple II+. Ein Erfahrungsbericht. Herwig Cuypers. 6/86/60
– Preissenkung für Apple-Computer. 3/86/68
– Prodos-Uhr für Apple IIc. 9/86/69
– 8086-Prozessorkarte. 3/86/69
– RAMWORKS und Z-RAM 3/86/67
– RGB-Farbinterfacekarte AI-80Z. Ein Erfahrungsbericht. Dirk Katzschke. 7/86/67
– Roboter-Plotter Penman. 3/86/68
– Roll-Maus. 4/86/69
– Satzprogramm Pagemaker. 3/86/66
– Sidekick. Macintosh-Büro-Manager. 6/86/66
– Sony-Laufwerke von Ueding. 1/84/64
– Star Gemini-15. Jürgen Geiß. 7/86/70
– STATEX für Bildschirme. 1-2/85/97
– Staub und Schmutz auf Computer und Tastatur? 1-2/85/97
– Synthesizersystem Juice digital. 3/86/66
– TempoHexe „SpeeDemon" 65C02C-Prozessorkarte. 1-2/85/97
– TRICARD – Multifunktionskarte für Apple IIe. 8/85/74
– Typenrad-Drucker Typeterm. 3/86/66
– UCSD p-System für den Macintosh. 1-2/85/97
– USV: Unterbrechungsfreie Stromversorgung. 4/86/70
– 220V-Steckdosenleiste. 3/86/69
– Z80+ Card. 4/86/70
– Z80+ Card. Hans-Martin Eng. 6/86/67
– Zeichnen mit der Maus. Maus-Grafiktablett – Mouse Tracer. Thomas Bühner. 8/86/61
– Z-RAM für den Apple IIc. Ein Erfahrungsbericht. Ulrich Tönnies. 6/86/62

TB: Testberichte
– Ampersoft. Eine Ampersand-Utility-Sammlung. Franz-Josef Hüskens. 1/86/68
– Apple II/IIe Assembler-Kurs. Dr. Jürgen B. Kehrel. 9/85/62
– BASF-Flexydisk 1X und 1D. us. 11/85/68
– Beagle Graphics. Rolf W. Becker. 8/85/77
– CP/M-Karte für den Apple IIc. Dr. Jürgen B. Kehrel. 3/85/75
– Dazzle Draw und Mousepaint. Dieter Charchot. 9/85/61
– Deutsche Sprachausgabe für den Apple. Harald Grumser. 9/85/63
– Die 16K-Akku-Karte LC-85. Harald Grumser. 10/85/72
– Die Microsoft-Premium-Softcard für den Apple IIe. us. 11/85/70
– Die OPERATOR IIe. Harald Grumser. 9/85/60
– Die Polaroid-Foto-Systeme. Thomas Bühner und Prof. Dr. Klaus Hausmann. 8/85/71
– Diskettenlaufwerke für ProDOS im Test. us. 2/84/74
– Diversi-DOS 4.1-C mit Garbage-Collection. Rudolf Rötering. 11/85/67
– Diversi-DOS 4-C. Rudolf Rötering. 6/85/72
– DMP-Charger – eigene Zeichensätze auf dem Matrixdrucker. Harald Grumser. 9/85/62 (Leserstellungnahme 5/86/73)
– Double-Hires-Grafik-Programme. Doublestuff Plus. Thomas Bühner. 3/85/70
– Double-Hires-Grafik-Programme. Superplot. Anmerkung von us. 3/85/70
– Druckerumschaltung per Knopfdruck. 5/85/71
– Editierhilfen für den Apple. Franz-Josef Hüskens. 6/85/77
– Erfahrungsbericht OPERATOR I. Rainer Hammerschmidt. 9/85/60
– Erphi-Controller. Eine kompatible Erweiterung. Harald Grumser. 4/85/75
– Exbasic Level II im Test. Eine Basic-Erweiterung bringt Sie ins Staunen, nicht nur positiv. Dr. Jürgen B. Kehrel. 1-2/85/88
– Extended Graphic System. Dr. Jürgen B. Kehrel. 6/85/71
– Gestochen scharf. Die Videokarte UltraTerm für den Apple II. Dieter und Jürgen Geiß. 2/84/60
– Hardbreaker und Softbreaker oder wie man Programme „entschützt". us. 3/85/72
– HOCO-Masterclock. Arne Schäpers. 3/85/76
– Kyan-Pascal-Programming Toolkits. System Utilities, Mouse Text und Turtle-Graphics. Matthias Meyer. 9/86/67
– Lasar II ze. Harald Grumser. 6/85/75
– Lernprogramme von INTUS. Dr. Jürgen B. Kehrel. 6/85/78
– Mailmerger und Appleworks. us. 4/85/73
– MEGACORE. Festplatte mit 10 MBytes. Harald Grumser. 7/85/76
– Memdos Junior. Ein neues Betriebssystem für den Apple II. Dr. Jürgen B. Kehrel. 3/85/75
– Microbuffer II. Dr. H. Kersten. 2/86/66
– MS-Basic 2.0 für den Mac. Pit Capitain. 12/85/76
– Pascal 1.2 Evolution statt Revolution. Claus Rautenstrauch. (Ergänzung 10/85/60) 3/85/65
– ProDOS-Debugger BUGBYTER. Dr. Jürgen B. Kehrel. 8/85/76
– Prometric-Motherboard. Dr. H. Vogel. 2/86/66
– S.A.M. The Software Automatic Mouth oder wie man den Apple zum Sprechen bringt. Ralf Augspurger. 2/84/66
– Slim-Line-Laufwerk von Chinon. 5/85/71
– Speedemon-Karte. us. 10/85/71
– Star Delta-10 und Grafstar-Interface. Karl-Walter Bott. 7/85/77
– The Last Disk-Utility. Ein umfangreicher Diskетteneditor. Harald Grumser. 5/85/72
– The Write Choice. Ein preiswertes Textverarbeitungsprogramm. Franz-Josef Hüskens. 1/86/66
– Triple-Dump. Ein Bildschirm-Druckprogramm. Rolf W. Becker. 11/85/66
– Typenradschreibmaschine Brother CE 50. Rainer Gerdes. 6/85/76
– Utilities von Beagle Brothers. Franz-Josef Hüskens. 11/85/65
– Video-Clips für jedermann. Die Video-1000-Karte. Norbert Man Brandl. 1/86/63
– Zwei neue Apple-IIe-Kompatible. Apple-IIe-Kompatibler SCSes. Harald Grumser. 10/85/74
– Zwei neue Apple-IIe-Kompatible. Apple-IIe-Kompatibler SCSe. Jürgen Geiß. 10/85/73

VE: Verschiedenes
– Apple-Diebe am Werk. 1-2/85/99
– Apple-Hotline. us. 3/85/35
– Apple User Club Austria. 1-2/85/99
– Diversi-DOS 2-C auf Peeker-Sammeldiskette #6, (Ergänzung 10/85/69) 6/85/74
– Große Peeker-Umfrage. 3/85/78
– Hinweise für Autoren. 2/84/72
– Hinweise für Autoren. 1/84/62
– Hinweise für Autoren. 1-2/85/92
– Hotline. 5/85/68
– Jahresinhaltsverzeichnis. Peeker 1/84 bis 12/85. 12/85/72
– Paul Lutus im ZDF. 9/85/68
– Peeker-Sammeldisketten #1-#11. 11/85/71
– Pyramid Pitty als Hex-Dump? 9/85/68
– Richtlinien für Autoren. 12/85/61

***

Stichwort-Sucher

Suchprogramm für kumuliertes Inhaltsverzeichnis

von Harald Grumser

Die Peeker-Sammeldiskette #23 enthält unter dem Dateinamen JAHRESINHALT ein 134 Sektoren umfassendes, kumuliertes Inhaltsverzeichnis aller Peeker-Ausgaben von Heft 1/1984 bis einschließlich Heft 9/1986 sowie ein neues Stichwort-Suchprogramm. Mit diesem neuen WORT.SUCHER läßt sich sehr komfortabel nach Stichwörtern suchen, z.B. nach Rubriken, Autoren, Begriffen oder sogar Wortfragmenten. Zu diesem Zweck lege man die Sammeldiskette #23 ein und starte das Programm mit RUN WORT.SUCHER. Wenn man jetzt nur Return eingibt, wird automatisch die Datei JAHRESINHALT geladen. Im Gegensatz zum Inhaltsverzeichnis, das in diesem Heft abgedruckt ist, enthält die Jahresinhalt-Datei keine Kleinbuchstaben und keine Umlaute. Für „Menü..." gebe man also „MENUE...", für „Adreß.." „ADRESS..." usw. ein.

Wichtiger Hinweis: Verwenden Sie nicht den alten Wortsucher von der Sammeldiskette #12, denn dieses Programm war *nur für Binärfiles* gedacht!

Wie wird gesucht?

– Um eine Übersicht über *alle* Rubrik-Namen zu erhalten, gebe man ein Sternchen (*) ein. Die Jahresinhalt-Datei enthält nämlich als Vorspann eine entsprechende Rubrik-Aufstellung, deren Einträge mit „*" beginnen.

– Um *alle* Einträge *einer* bestimmten Rubrik anzusehen, gebe man das zweistellige Rubrik-Kürzel, gefolgt von einem Doppelpunkt ein, z.B. „AP:". Der Doppelpunkt ist hier wichtig, weil die zweistelligen Kürzel auch in anderen Aufsatztiteln enthalten sein könnten (z.B. „AP" in „APPLE").

– Um alle Aufsätze *eines* bestimmten Heftes zu listen, gebe man die Heftnummer in der Form „ 1/85/", „ 10/85/" usw. ein, also Leertaste + Heftnummer + „/" + Jahreszahl + „/". Durch die Leertaste vor der Heftnummer und den Schrägstrich nach der Jahreszahl wird vermieden, daß nach anderen Zahlenkombinationen gesucht wird.

– Um nach einem bestimmten Autor zu suchen, gebe man den vollen Zunamen ein.

– Um nach einem beliebigen Begriff zu suchen, gebe man diejenige Zeichenfolge ein, an die man sich noch erinnert. Beispielsweise würde „FUNKT" alle Aufsatztitel anzeigen, die Begriffe wie „Funktionen" usw. enthalten.

Eigene Suchdateien

Der WORT.SUCHER kann auch zum Durchsuchen *eigener* Textfiles verwendet werden, sofern sie aus Einzelstrings aufgebaut sind, die durch Returns abgegrenzt werden. Dateiaufbau:
– Gesamtlänge: maximal 135 Sektoren ($0B6C-$95FF; bei MAXFILES 1 noch einige Sektoren mehr).
– Länge je String: maximal 255 Zeichen.
– Endmarker: Ctrl-D oder $00 (Dies kann im Listing WORT.SUCHER.OBJ, Zeile 100, geändert werden.)

Kurzhinweise

1. Zweck:
Suchprogramm für Peeker-Jahresinhaltsverzeichnis oder ähnlich aufgebaute Textfiles.
2. Konfiguration:
II+/e/c; DOS 3.3 (kein ProDOS!)
3. Test:
RUN WORT.SUCHER
4. Sammeldisk:
WORT.SUCHER
T.WORT.SUCHER.OBJ
WORT.SUCHER
5. Sonstiges:
Vor dem Programmstart kann man bei Bedarf die 80-Zeichenkarte oder den Drucker einschalten, falls die Ausgabe nicht über den 40-Zeichenbildschirm erfolgen soll.

WORT.SUCHER

```
1Ø HOME : PRINT  CHR$ (4)"BLOAD WORT.SUCHER.OBJ"
2Ø PRINT "WORT.SUCHER": PRINT "═══════════"
3Ø PRINT : PRINT : PRINT "BITTE DEN NAMEN DER"
4Ø PRINT "TEXTDATEI EINGEBEN!"
5Ø PRINT : PRINT "(<RETURN> = JAHRESINHALT)"
6Ø PRINT : INPUT "";N$
7Ø IF LEN (N$) = Ø THEN N$ = "JAHRESINHALT"
8Ø PRINT  CHR$ (4)"OPEN"N$
9Ø CALL 25ØØ
```

WORT.SUCHER.OBJ

```
BSAVE WORT.SUCHER.OBJ, A25ØØ,L424

 1  *──────────────────*
 2  *                  *
 3  *  STICHWORT.SUCHER  *
 4  *                  *
 5  * Harald Grumser, 1985 *
 6  *──────────────────*
 7
 8           ORG  $Ø9C4      ;25ØØ
 1Ø
 11 PROMPT   EQU  $33
 12 MEMSIZE  EQU  $73
```

```
                        13  TEXT     EQU  $F8
                        14  LEN      EQU  $FA
                        15  POS      EQU  $FB
                        16  FMPL     EQU  $FD
                        17
                        18  IN       EQU  $200
                        19  DOSWARM  EQU  $3D0
                        20  FM       EQU  $3D6       ;File-Manager
                        21  GETFMPL  EQU  $3DC       ;Parameter-Liste
                        22
                        23  KBD      EQU  $C000
                        24  STROBE   EQU  $C010
                        25
                        26  GETLINE1 EQU  $FD6F
                        27  COUT     EQU  $FDED
                        28  HOME     EQU  $FC58
                        29
                        30  * Datei einlesen
                        31
                        32  * Datei muß durch Applesoft-Programm
                        33  * geöffnet werden!!!
                        34
09C4: 38                35          SEC
09C5: A5 73             36          LDA  MEMSIZE
09C7: E9 6C             37          SBC  #<BUFFER
09C9: 8D 68 0B          38          STA  LENGHT
09CC: A5 74             39          LDA  MEMSIZE+1
09CE: E9 0B             40          SBC  #>BUFFER
09D0: 8D 69 0B          41          STA  LENGHT+1
                        42
09D3: 20 DC 03          43          JSR  GETFMPL
09D6: 84 FD             44          STY  FMPL
09D8: 85 FE             45          STA  FMPL+1
09DA: A0 09             46          LDY  #9
09DC: B9 62 0B          47  SETFMPL LDA  FMPLIMGE,Y
09DF: 91 FD             48          STA  (FMPL),Y
09E1: 88                49          DEY
09E2: 10 F8             50          BPL  SETFMPL
09E4: 20 D6 03          51          JSR  FM
09E7: 90 12             52          BCC  TOOLONG
09E9: A0 0A             53          LDY  #10
09EB: B1 FD             54          LDA  (FMPL),Y    ;Return-Code
09ED: C9 05             55          CMP  #5          ;OUT OF DATA
09EF: F0 10             56          BEQ  START
09F1: A9 46             57          LDA  #<TEXT3
09F3: A2 0B             58          LDX  #>TEXT3
09F5: 20 86 0A          59  ERROR   JSR  TXTOUT
09F8: 4C D0 03          60  RETURN  JMP  DOSWARM
                        61
09FB: A9 52             62  TOOLONG LDA  #<TEXT4
09FD: A2 0B             63          LDX  #>TEXT4
09FF: D0 F4             64          BNE  ERROR       ;immer
                        65
                        66  * Such-String einlesen
                        67
0A01: 20 58 FC          68  START   JSR  HOME
0A04: A9 BA             69          LDA  #":"
0A06: 85 33             70          STA  PROMPT
0A08: A9 A7             71          LDA  #<TEXT1
0A0A: A2 0A             72          LDX  #>TEXT1
0A0C: 20 86 0A          73          JSR  TXTOUT
0A0F: 20 6F FD          74          JSR  GETLINE1
0A12: 8A                75          TXA
0A13: F0 EC             76          BEQ  START
0A15: 86 FA             77          STX  LEN
0A17: AD 00 02          78          LDA  IN
0A1A: C9 83             79          CMP  #$83        ;Ctrl-C
0A1C: F0 DA             80          BEQ  RETURN
0A1E: 20 2D 0A          81          JSR  SEARCH
0A21: A9 30             82          LDA  #<TEXT2
0A23: A2 0B             83          LDX  #>TEXT2
0A25: 20 86 0A          84          JSR  TXTOUT
0A28: 20 9B 0A          85          JSR  KEYWAIT
0A2B: 30 D4             86          BMI  START       ;immer
                        87
                        88  * Stichwort suchen
                        89
0A2D: A0 6C             90  SEARCH  LDY  #<BUFFER
0A2F: A9 0B             91          LDA  #>BUFFER
0A31: 84 F8             92          STY  TEXT
0A33: 85 F9             93          STA  TEXT+1
0A35: A2 00             94  SLOOP   LDX  #$00
0A37: 84 FB             95          STY  POS
0A39: E4 FA             96  SLOOP1  CPX  LEN
0A3B: F0 1A             97          BEQ  FOUND
0A3D: B1 F8             98          LDA  (TEXT),Y
0A3F: F0 41             99          BEQ  RETSRCH
0A41: C9 84            100          CMP  #$84        ;Ctrl-D
0A43: F0 3D            101          BEQ  RETSRCH
0A45: C9 8D            102          CMP  #$8D
```

```
0A47: F0 2B            103          BEQ  NOTFOUND
0A49: DD 00 02         104          CMP  IN,X
0A4C: D0 04            105          BNE  NEXTPOS
0A4E: E8               106          INX
0A4F: C8               107          INY
0A50: D0 E7            108          BNE  SLOOP1
0A52: A4 FB            109  NEXTPOS LDY  POS
0A54: C8               110          INY
0A55: D0 DE            111          BNE  SLOOP
                       112
0A57: A0 FF            113  FOUND   LDY  #-1
0A59: C8               114  PRINT   INY
0A5A: B1 F8            115          LDA  (TEXT),Y
0A5C: 20 ED FD         116          JSR  COUT
0A5F: C9 8D            117          CMP  #$8D
0A61: D0 F6            118          BNE  PRINT
0A63: 20 ED FD         119          JSR  COUT
0A66: AD 00 C0         120          LDA  KBD
0A69: C9 A0            121          CMP  #$A0
0A6B: D0 03            122          BNE  NOSPACE
0A6D: 20 9B 0A         123          JSR  KEYWAIT
0A70: C9 9B            124  NOSPACE CMP  #$9B        ;ESC
0A72: F0 0E            125          BEQ  RETSRCH
                       126
0A74: 98               127  NOTFOUND TYA
0A75: A0 00            128          LDY  #$00
0A77: 38               129          SEC
0A78: 65 F8            130          ADC  TEXT
0A7A: 85 F8            131          STA  TEXT
0A7C: 90 B7            132          BCC  SLOOP
0A7E: E6 F9            133          INC  TEXT+1
0A80: D0 B3            134          BNE  SLOOP
                       135
0A82: 8D 10 C0         136  RETSRCH STA  STROBE
0A85: 60               137          RTS
                       138
                       139  * Text ausgeben
                       140
0A86: 85 F8            141  TXTOUT  STA  TEXT
0A88: 86 F9            142          STX  TEXT+1
0A8A: A0 00            143          LDY  #$00
0A8C: B1 F8            144  TXTLOOP LDA  (TEXT),Y
0A8E: F0 0A            145          BEQ  RTNTO
0A90: 20 ED FD         146          JSR  COUT
0A93: C8               147          INY
0A94: D0 F6            148          BNE  TXTLOOP
0A96: E6 F9            149          INC  TEXT+1
0A98: D0 F2            150          BNE  TXTLOOP
0A9A: 60               151  RTNTO   RTS
                       152
                       153  * Tastendruck abwarten
                       154
0A9B: 8D 10 C0         155  KEYWAIT STA  STROBE
0A9E: AD 00 C0         156  KEYWAIT1 LDA  KBD
0AA1: 10 FB            157          BPL  KEYWAIT1
0AA3: 8D 10 C0         158          STA  STROBE
0AA6: 60               159          RTS
                       160
                       161  * Menütexte
                       162
0AA7: 8C               163  TEXT1   HEX  8C          ;Ctrl-L
0AA8: C2 C9 D4         164          ASC  "BITTE STICHWORT "
0AB8: C5 C9 CE         165          ASC  "EINGEBEN!"
0AC1: 8D 8D            166          HEX  8D8D
0AC3: A8 BC C3         167          ASC  "(<CTRL-C> <RETURN> =
                                         PROGRAMMENDE)"
0AE5: 8D 8D 8D         168          HEX  8D8D8D8D
0AE9: CC C9 D3         169          ASC  "LISTING ANHALTEN "
0AFA: CD C9 D4         170          ASC  "MIT <SPACE>"
0B05: 8D               171          HEX  8D
0B06: CC C9 D3         172          ASC  "LISTING BEENDEN "
0B16: A0 CD C9         173          ASC  " MIT <ESC>"
0B20: 8D 8D 8D         174          HEX  8D8D8D8D
0B24: D3 D4 C9         175          ASC  "STICHWORT: "
0B2F: 00               176          HEX  00
0B30: 8D 8D 87         177  TEXT2   HEX  8D8D87
0B33: AD A0 D4         178          ASC  "- TASTE DRUECKEN -"
0B45: 00               179          HEX  00
0B46: 8D 87            180  TEXT3   HEX  8D87
0B48: C9 AF CF         181          ASC  "I/O-ERROR"
0B51: 00               182          HEX  00
0B52: 8D 87            183  TEXT4   HEX  8D87
0B54: C4 C1 D4         184          ASC  "DATEI ZU LANG"
0B61: 00               185          HEX  00
                       186
0B62: 03 02 00         187  FMPLIMGE HEX  03020000000000
0B68: 00 00            188  LENGHT  HEX  0000
0B6A: 6C 0B            189          DA   BUFFER
                       190  BUFFER  EQU  *
```

GFA-BASIC

Ein schnelles BASIC im Pascal-Gewand

getestet von Ulrich Stiehl

1. Gibt es eine BASIC-Renaissance?

Bei den bisherigen 8-Bit-Rechnern mit 64K RAM war BASIC die einzige Hochsprache, die sich bequem im Speicher unterbringen ließ. Unter Berücksichtigung der Leistungsfähigkeit von 8-Bit-Prozessoren waren die alten BASIC-Interpreter echte Glanzleistungen, was die Kompaktheit und die Verarbeitungsgeschwindigkeit anbelangte, denn die Systemprogrammierer der alten Schule griffen tief in die Assembler-Trickkiste, um das Äußerste aus der jeweiligen CPU herauszuholen. Dafür mußten allerdings erhebliche Abstriche hinsichtlich der Anzahl der Befehle, des Programmierkomforts und der Strukturierung gemacht werden. Es verwundert deshalb nicht, daß inzwischen im schulischen und universitären Bereich meist die Sprache Pascal verwendet wird, die allerdings für 8-Bit-Rechner ohne RAM-Disk oder Festplatte zu umfangreich und damit wenig komfortabel ist.

Bei den neuen Rechnern mit 16-Bit-Prozessor und großem RAM-Speicher könnte BASIC eine Renaissance erleben, weil nunmehr einer Vermehrung des Befehlsvorrats, einer Erhöhung des Programmierkomforts sowie einer Erweiterung durch pascal-ähnliche Strukturen nichts mehr im Wege stünde. Damit ließen sich zwei Fliegen mit einer Klappe schlagen: Zum einen würde der Interpreter den Umgang mit der Programmiersprache sowie die Programmentwicklung ungemein erleichtern. Zum andern würden leistungsfähige und relativ schnelle Programme entstehen, deren Ausführungsgeschwindigkeit bei Bedarf mit Hilfe eines Compilers noch erheblich gesteigert werden könnte.

Die ersten BASIC-Interpreter für den Macintosh und den Atari ST, die teils in anderen Hochsprachen (C usw.) geschrieben worden sind, waren jedoch eine herbe Enttäuschung. So war beispielsweise das Microsoft-BASIC für den Macintosh oft sogar noch langsamer als das Applesoft-BASIC für den alten Apple II. Ähnlich Unerfreuliches erlebten wir mit dem Digital-Research-BASIC für den Atari ST, das bei einer Länge von rund 140K über einen gegenüber CP/M-MBASIC nur unwesentlich erweiterten Befehlsumfang verfügte und zudem enttäuschend langsam sowie völlig „verwanzt“ war. So führten bestimmte Befehle, z.B. X = 75.7, zu einem völligen Systemzusammenbruch, so daß der Griff zum Reset-Knopf zur Gewohnheit werden konnte. (Es ist jedoch soeben eine neue BASIC-Version von Metacomco angekündigt worden, die kostenlose an Atari-Besitzer abgegeben wird.)

Mit dem GFABASIC der Firma GfA Systemtechnik in Düsseldorf liegt jetzt jedoch ein Interpreter vor, der vielen Atari-Besitzern BASIC wieder für kleinere bis mittelgroße und vielleicht sogar für große Programmieraufgaben schmackhaft machen wird. Greifen wir als Beispiel den Komfort heraus. An anderer Stelle wird in diesem Heft der zweifellos sehr leistungsstarke Mark-Williams-C-Compiler gewürdigt, der für große bis sehr große Programmprojekte ausgelegt ist. Es verwundert deshalb nicht, daß dieser C-Compiler für einen Einzeiler, z.B. „printf soundso“, bei Verwendung von Diskettenlaufwerken 3 Minuten benötigt und dabei einen 8K langen Objektcode erzeugt. Der Wechsel zwischen Editor und Compiler wird deshalb bei kürzeren C-Programmen zu einer zermürbenden Angelegenheit.

2. Vor- und Nachteile

2.1. Vorteile von GFABASIC

- Der Interpreter nimmt nur ca. 56K ein und wird in weniger als 10s komplett von Diskette geladen, und zwar wahlweise vom GEM-Desktop oder auch aus einem DOS-Shell-Programm heraus (s. Peeker, 10/86, S. 73). Danach wird im Gegensatz zu anderen Programmiersprachen, zu denen neben C, Pascal usw. auch das Microsoft-BASIC für den Macintosh gehört, nicht mehr auf die Diskette zugegriffen, da das 56K-Programm alles enthält (Editor, DOS-Schnittstelle, Interpreter, Runtime-Routinen usw.).
- Der Editor verwendet im Gegensatz zum DR-BASIC keine Fenstertechnik und ist deshalb sehr schnell. Es wurde dabei an alles gedacht, was für die Programmentwicklung erforderlich ist (Überschreiben, Einfügen, Löschen, Verschieben von Programmblöcken, Suchen, Ersetzen usw.). Das Einrücken von logischen Programmblöcken (If-Then-Konstrukte usw.) erfolgt vollautomatisch. Wie beim Apple-II-BASIC (und im Gegensatz zum CP/M-MBASIC) kann man Programmzeilen, deren Syntax nach Eingabe von Return sofort überprüft wird, in der Regel ohne Leerzeichen eingeben. So wird beispielswese „?.7“ automatisch in „Print 0.7“ umgewandelt, so daß man sich im Gegensatz zu Pascal keine Gedanken machen muß, ob „.7“ als „0.7“ zu schreiben ist.
- Die Systembefehle (Load, Save, List usw.) können wahlweise über die Tastatur oder per Maus/Fenster erteilt werden. Auch hier denkt der Interpreter mit: Aus „savedemo“ wird automatisch „Save "Demo.Bas"“, aus „listdemo“ wird automatisch „List "Demo.Lst"“ usw.
- Die ab Mitte Oktober erhältliche neue

Version 2.0 des GFABASIC-Interpreters umfaßt etwa 300 Befehle, wenn man z.B. „Inp (x)", „Inp (#n)" und „Inp? (n)" als getrennte Kommandos ansieht. Hinzu kommen noch diverse Gemsys-, Vdisys- und Bios-Aufrufe.

● Programme können gepackt (Dateiname.Bas) oder als ASCII-Textfiles (Dateiname.Lst) gespeichert werden. Im Spätherbst wird noch ein GFABASIC-Compiler folgen, doch können im Moment noch keine Angaben über die Länge der auf diesem Wege erzeugten Objektcodes gemacht werden.

● Die Ausführungsgeschwindigkeit der Programme durch den Interpreter ist extrem hoch. Dies gilt namentlich für mathematische und grafische sowie in geringerem Maße für string-intensive Programme. Auch spielt es kaum eine Rolle, ob Unterprogramme am Anfang oder am Ende eines längeren Programms stehen, da für diesen Zweck offenbar Sprungtabellen angelegt werden. Trotzdem soll der in Arbeit befindliche Compiler eine Geschwindigkeitssteigerung um den Faktor 2 bis 30 bringen. Näheres zur Geschwindigkeit siehe unten.

● Das Zeilennummernkonzept wurden zugunsten von Namen für Labels und Prozeduren fallengelassen. Darüber hinaus wurden pascal-ähnliche Strukturen eingeführt, z.B. Repeat-Until, lokale Variablen und Zeiger, Prozeduren mit Wert- und Variablenparametern und anderes. Näheres hierzu siehe unten.

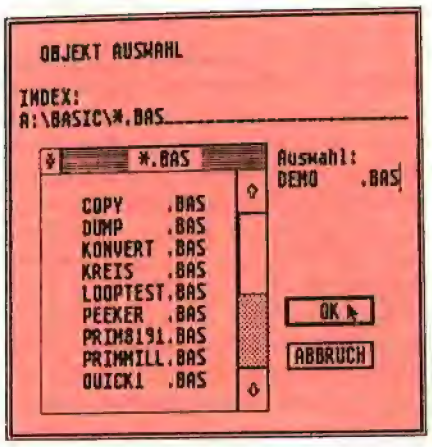


Abb. 1: File select-Befehl

● Neben den traditionellen BASIC-Befehlen ist auch an die GEM-Desktop-Programmierung gedacht worden, die mit GFABASIC teilweise spielend einfach ist. So erzeugt beispielsweise der Befehl

```
Fileselect "A:\Basic\*.bas", "Demo.bas", D$
```

eine Dateiauswahl-Fenster, in dem man wie auf dem normalen GEM-Desktop scrollen und auswählen kann, wobei der angeklickte Dateiname in diesem Fall der Stringvariablen D$ zugewiesen wird (vgl. **Abb. 1**).

● Für den Preis von DM 149,- (ab Version 2.0 DM 169,-) erhält man eine nicht-kopiergeschützte Programmdiskette sowie einen DIN-A5-Ordner, der im wesentlichen aus einer alphabetischen Befehlsbeschreibung besteht. Der Compiler wird separat zum gleichen Preis erhältlich sein. Damit ist das GFABASIC spürbar preiswerter als viele andere Programmiersprachen.

2.2. Nachteile von GFABASIC

● Obwohl es zur Version 1.0 bereits zwei Updates gab und nunmehr die Version 2.0 erscheint, die uns freundlicherweise vorab zu Testzwecken zugesandt worden ist, gibt es immer noch einige harmlose Fehler, mit denen man leben kann, sowie einige unerquickliche Bugs, vor denen man sich tunlichst hüten muß. Wenn beispielsweise ein laufendes Programm, das Strings in Unterprogrammen bzw. Prozeduren verwendet, nicht mit Clear beendet wird, darf man, wenn man in den Editor zurückgelangt, keine Programmzeile löschen. Tut man dies trotzdem und geht dann mit Esc in den Runtime-Bildschirm zurück, so bekommt man im günstigsten Fall die Fehlermeldung „Ungerade Wort-Adresse bei Peek", während im ungünstigsten Fall der Rechner hängt und nur noch durch Reset zu neuem Leben erweckt werden kann. Ähnlich wirre, aber harmlosere Fehlermeldungen können auftreten, wenn man ein laufendes Programm innerhalb einer geschachtelte Schleife unterbricht und dann in den Runtime-Bildschirm zurückkehrt.

● Die Befehle sind nicht immer konsequent genug durchgeführt und teils auch mangelhaft dokumentiert. Beispielsweise ist bereits bei der Version 1.0 der Lprint-Using-Befehl implementiert, aber nicht im Handbuch beschrieben worden. Der For-to-Befehl läßt eine Schrittweite (+Step) zu, während dies beim For-downto-Befehl nicht möglich ist; statt dessen muß man dann den For-to-Befehl mit negativer Schrittweite (-Step) verwenden. Dann hätte man sich den For-downto-Befehl auch ganz schenken können. Der Run-Befehl, mit dem man ein Programm direkt von Diskette starten kann, heißt skurrilerweise Chain, während der eigentliche MBASIC-Chain-Befehl mit Common nicht implementiert worden ist. Ferner vermißt man die Typdeklarationen Defint, Defstr usw. sowie einen Random-Befehl, der *dieselben* Zufallszahlenfolgen *wiederholt* erzeugen kann.

● Im Editor kann man immer nur einen einzigen Befehl pro Zeile eingeben, die allerdings 255 Zeichen lang sein kann. Der Doppelpunkt ist also nicht als Befehlsseparator verwendbar. Damit müssen Wertzuweisungen wie
A = 1: B = 2: C = 3
auf 3 Zeilen verteilt werden. Ab der Version 2.0 kann man wenigstens Rem-Kommentare in derselben Zeile hinzufügen. Als Separator dient dann das Ausrufezeichen.

Aus Platzgründen beschränken wir uns im nachfolgenden auf ausgewählte Programmbeispiele zu den Themen Geschwindigkeit und Strukturierung und bringen im Anschluß daran eine systematische Kurzübersicht. Darüber hinaus wird ab dem nächsten Peeker-Heft ein mehrteiliger Kurs für *Anfänger* veröffentlicht, der auch die Editor-Befehle behandeln wird. (Die folgenden Programmbeispiele sind nicht für Anfänger gedacht.)

3. Geschwindigkeit

Die Geschwindigkeit von GFABASIC ist enorm. Beispielsweise dauert der Aufruf einer „Leerprozedur", z.B.
Procedure Name
Return
selbst bei einem großen Programm nur 0,0002s (Sekunden). Durch geschickten Einsatz passender Befehle lassen sich beachtliche Geschwindigkeitssteigerungen erzielen. Dies gilt namentlich für die Grafik- und Stringbefehle.

Bsp. 1: Kreise (GFABASIC)

```
Rem Eigener Circle-Befehl
T=Timer
H=320
V=200
S=0.01 !Genauigkeit!
For R=5 To 195 Step 10
  For I=0 To 6.3 Step S
    Plot H+R*Cos(I),V+R*Sin(I)
  Next I
Next R
Print (Timer-T)/200 !44,6s
Input A$
Rem Implementierter Circle-Befehl
T=Timer
Cls
For R=5 To 195 Step 10
  Circle H,V,R
Next R
Print (Timer-T)/200 !0,75s
Input A$
```

Bsp. 2: Kreise (Applesoft)

```
10 HGR2 : HCOLOR= 7:
   H = 140:V = 90:S = 0.01
30 FOR R = 8 TO 84 STEP 4:
   FOR I = 0 TO 6.3 STEP S:
   HPLOT H + R *  COS (I),
   V + R *  SIN (I): NEXT I,R
40 GET X$: TEXT
50 REM 235s IIe mit 3,5MHz
```

Das obige **Beispiel 1** zeichnet 20 konzentrische Kreise, und zwar erst mit dem

Punkt-Plot-Befehl (ca. 44.6s) und dann mit dem Circle-Befehl (ca. 0,75s). Dies entspricht einem Verhältnis von 60 zu 1. Zum Vergleich wird als **Beispiel 2** ein Applesoft-Programm für den Apple II mit 3,5-MHz-Accelerator vorgestellt. Der hier ermittelte Wert (235s) verhält sich zum Punkt-Plot-Demo wie 5,3 zu 1 und zum Circle-Demo wie 313 zu 1. GFABASIC dürfte im Mittel 15mal schneller als Applesoft-BASIC auf einem Apple GS oder einem Apple IIe mit Accelerator-Karte sein.

Bsp. 3: Dateikopierprogramm

```
Line Input "Quelldatei";Q$
Line Input "Zieldatei ";Z$
Open "I",#1,Q$
Open "O",#2,Z$
R=Lof(#1)
L=32256
While R>L
  B$=Input$(L,#1)
  Print #2,B$;
  R=R-L
Wend
B$=Input$(R,#1)
Print #2,B$;
Close
```

Das obige **Beispiel 3**, das 2 Volumes (2 Drives oder 1 Drive + 1 RAM-Disk) voraussetzt, zeigt auf eindrucksvolle Weise, daß mit wenigen Programmzeilen ein vollständiges Dateikopierprogramm für Programm- und Datendateien aller Art geschrieben werden kann. Strings können nämlich eine Länge von bis zu 32767 Bytes einnehmen, die mit den Input/Print-Befehlen „auf einen Schlag" von/auf Diskette gelesen/geschrieben werden können.

Bsp. 4: Instring-Find-Test

```
Rem 1s für den 2000. String
N%=2000
Dim S$(N%)
Rem Erzeugen dauert 20s
For X%=1 To N%
  For Y%=1 To 10
    S$(X%)=S$(X%)+Chr$(Random(25)+65)
  Next Y%
Next X%
Do
  Print N%;". String: ";S$(N%)
  Input "Suchstring: ";Such$
  Exit If Such$=""
  T=Timer
  For X%=1 To N%
    If Instr(S$(X%),Such$)<>0 Then
      Print X%;". ";S$(X%)
    Endif
  Next X%
  Print (Timer-T)/200
Loop
```

Das obige **Beispiel 4** testet GFABASIC auf die Verwendbarkeit für Dateiverwaltungsprogramme, für die man üblicherweise Indexdateien zum Suchen von Datensätzen anlegt. Zunächst werden 2000 „Indexfelder" (= 2000 Strings mit je 10 Zufallsbuchstaben) angelegt. Dies dauert seine Zeit (20s) und zeigt, daß nicht alle String-Befehle schnell abarbeitet werden. Sehr leistungsfähig ist jedoch der Instring-Befehl, der im ungünstigsten Fall den 2000sten String in maximal 1s findet, nachdem zuvor alle 1999 vorangehenden Strings vergeblich überprüft worden sind. Wenn also beispielsweise bei einem Adreßverwaltungsprogramm für 2000 Adressen eine Suchdauer von 1s ausreicht, kann man auf intelligente Suchalgorithmen (binäres Suchen usw.) verzichten und statt dessen die leicht programmierbare „Holzhackermethode" (sequentielles Durchsuchen) anwenden.

Bsp. 5: Primzahlen bis 6 Mio.

```
T1=Timer
H%=6000000
Dim P!(H%)
S%=Int(Sqr(H%))+1
Arrayfill P%(),0
For X%=2 To S%
  If P!(X%)=0 And X%+X%<H% Then
    For Y%=X%+X% To H% Step X%
      P!(Y%)=-1
    Next Y%
  Endif
Next X%
T2=Timer
T3=(T2-T1)/200
Print Int(T3/60);"m, ";
Print Int(T3-Int(T3/60)*60);"s"
Rem Fast exakt 50 min. bis 6.000.000
A%=0
For X%=2 To H%
  If P!(X%)=0 Then
    Inc A%
  Endif
Next X%
Print A%;" Primzahlen von 2 bis ";H%
Rem 412.849 Primzahlen bis 6.000.000
```

Das obige **Beispiel 5** zeigt nicht nur die Geschwindigkeit, mit der der Eratosthenes-Primzahlenalgorithmus abgearbeitet wird (50 Minuten für 412.849 Primzahlen von 2 bis 6.000.000), sondern daß unter GFABASIC der Speicher vollständig genutzt werden kann. Beim Atari 1040 ST stehen nämlich unter GFABASIC bis zu 800K für Programm und Daten zur Verfügung (beim 520 ST entsprechend weniger). Zudem werden boolesche Arrays bitmäßig gepackt gespeichert. Für 6.000.000 boolesche Werte werden 6.000.000 : 8 Bits = 750.000 Bytes benötigt. Damit ist der Speicher dann weitgehend voll. Viele andere Programmiersprachen (z.B. UCSD-Pascal) sind nicht in der Lage, derart gigantische Arrays zu verwalten.

4. Strukturierung

Wer schon unter Pascal programmiert hat, wird an GFABASIC seine wahre Freude haben, denn es wurden mit Ausnahme des Case-Befehls alle pascaltypischen Strukturbefehle implementiert und darüber hinaus weitere Strukturen (Do-Loop) eingeführt. Ein Vergleich der wichtigsten Konstrukte (P = Pascal; B = GFABASIC):

For-Schleife
P:
For I := A To E Do Begin ... End
B:
For I = A To E ... Next I
Repeat-Schleife
P:
Repeat ... Until bedingung
B:
Repeat ... Until bedingung
While-Schleife
P:
While bedingung Do Begin ... End
B:
While bedingung ... Wend
If-Verzweigung
P:
If bedingung Then Begin ... End
B:
If bedingung Then ... Endif

Das nachfolgende **Beispiel 6** zeigt die Anwendung der Strukturbefehle unter GFABASIC. Darüber hinaus besteht, wie aus dem Demo ersichtlich ist, die Möglichkeit, Prozeduren mit Wert- und Variablenparametern zu definieren. Darauf werden wir zu einem späteren Zeitpunkt im Peeker näher eingehen.

Bsp. 6: Strukturbefehle

```
Print "Aufwärts-Step möglich"
For I=1 To 3 Step 0.5
  Print I
Next I
Rem ---
Print "Abwärts-Step nicht möglich"
For I=3 Downto 1
  Print I
Next I
Rem ------------------------------
Print "Statt Zeilennummern Labels"
Goto Label
Print "Wird ignoriert"
Label:
Print "Hier bei Label"
Rem ------------------------------
Print "Repeat-Vergleich nachher"
I=1
Repeat
  Print I
  I=I+1
Until I>3
Rem ---
Print "While-Vergleich vorher"
I=1
While I<=3
  Print I
  I=I+1
Wend
Rem ------------------------------
Print "Endlosschleife mit Exit"
I=1
Do
  Print I
  I=I+1
  Exit If I>3
Loop
Rem ------------------------------
Print "If-endif"
I=0
If I=0 Then
  Print 0
Endif
Rem ---
Print "If-then-else-endif"
If I=1 Then
  Print 1
```

```
Else
  If I=2 Then
    Print 2
  Else
    If I=3 Then
      Print 3
    Else
      Print "Nicht 1, 2, 3"
    Endif
  Endif
Endif
Rem ------------------------------
Print "Globale Variablen"
G=1
Gosub Global
Print G
Rem ---
Print "Lokale Variablen"
L=1
Gosub Local
Print L !weiterhin 1!
Rem ----
Print "Wertparameter"
W=1
Gosub Wertparameter(W)
Print W !weiterhin 1!
Rem ---
Print "Variablenparameter"
V=1
Gosub Variablenparameter(*V)
Print V !jetzt 2!
Rem ---
End
Rem ------------------------------
Procedure Global
  G=2
Return
Rem ---
Procedure Local
  Local L
  L=2
  Print L
Return
Rem ---
Procedure Wertparameter(Wert)
  Wert=Wert+1
  Print Wert
Return
Rem ---
Procedure Variablenparameter(Ptr)
  *Ptr=V+1 !entspricht V = V+1!
  Print V
Return
```

4.1. Quicksort-Beispiele

Im Peeker, Heft 1/86 sind zwei umfangreiche Aufsätze erschienen, die den Quicksort-Algorithmus in allen Einzelheiten anhand eines BASIC- und eines Assembler-Programms erläutern. Der dort abgedruckte BASIC-Algorithmus war für die bisherigen BASIC-Dialekte gedacht, die mit Zeilennummern arbeiten und weder über Repeat-Until-Konstrukte noch über rekursive Prozeduren verfügen. **Beispiel 7** (Quicksort mit Labels) ist eine exakte Übertragung des im Peeker, 1/86, S. 13, abgedruckten Applesoft-Programms in die analoge GFABASIC-Version. Obwohl dieses GFABASIC-Programm bereits eindrucksvoll schnell ist (ca. 11s für 1000 Elemente gegenüber 275s in Applesoft), brauchen und sollten Sie unter GFABASIC niemals derartigen Spaghetti-Code erzeugen. Trotz Anwendung desselben Algorithmus ist nämlich das **Beispiel 8** (Quicksort fast ohne Labels) bereits bedeutend übersichtlicher und auch etwas schneller (ca. 8.3s für 1000 Elemente), was mit der Verwendung der Dec-, Inc- und Swap-Befehle zusammenhängt. Es geht aber noch weiter: Eine spürbare algorithmische Verbesserung stellt **Beispiel 9** dar (Iteratives Quicksort), doch die Krönung wird in **Abb. 2** (Rekursives Quicksort, hier für Strings statt für Zahlen) gezeigt, denn GFABASIC läßt tatsächlich wie in Pascal rekursive Prozeduren zu. Die iterativen und rekursiven Programmversionen konnten deshalb auch unverändert einschlägigen Pascal- und Informatik-Lehrbüchern entnommen werden.

Bsp. 7: Quicksort mit Labels

```
Rem Quicksort für Reals: 11s
Sn%=1000      ! 1000 Elemente
Dim T(Sn%+1) ! T()-Real-Array
For X%=1 To Sn%
  T(X%)=Rnd*1000
Next X%
Print "Unsortiert"
Gosub Anzeige
Print "Start";Chr$(7)
U1=Timer
Gosub Sort
U2=Timer
Print "Ende";Chr$(7)
Print "Sortiert"
Gosub Anzeige
Print (U2-U1)/200;" Sekunden"
End
Procedure Anzeige
  For X%=1 To Sn%
    Print T(X%),
  Next X%
Return
Rem Quicksort-Unterroutine
Procedure Sort
  T(Sn%+1)=10000
  Sf%=1
  Sl%=Sn%
  Dim Ss%(30)
  Sc%=0
  Label1:
  If Sf%>=Sl% Then
    Goto Label7
  Endif
  Sd%=Sl%+1
  Si%=Sf%
  Sm%=Int((Sl%-Sf%)/2)+Sf%
  T=T(Sm%)
  T(Sm%)=T(Sf%)
  T(Sf%)=T
  Label2:
  Si%=Si%+1
  If T(Si%)<T Then
    Goto Label2
  Endif
  Label3:
  Sd%=Sd%-1
  If T(Sd%)>T Then
    Goto Label3
  Endif
  If Sd%<=Si% Then
    Goto Label4
  Endif
  Tt=T(Si%)
  T(Si%)=T(Sd%)
  T(Sd%)=Tt
  Goto Label2
  Label4:
  T(Sf%)=T(Sd%)
  T(Sd%)=T
  If Sd%-Sf%<Sl%-Sd% Then
    Goto Label5
  Endif
  Ss%(Sc%+1)=Sf%
  Ss%(Sc%+2)=Sd%-1
  Sf%=Sd%+1
  Goto Label6
  Label5:
  Ss%(Sc%+1)=Sd%+1
  Ss%(Sc%+2)=Sl%
  Sl%=Sd%-1
  Label6:
  Sc%=Sc%+2
  Goto Label1
  Label7:
  If Sc%>0 Then
    Sl%=Ss%(Sc%)
    Sf%=Ss%(Sc%-1)
    Sc%=Sc%-2
    Goto Label1
  Endif
Return
```

Bsp. 8: Quicksort fast ohne Labels

```
Rem Quicksort für Reals: 8.3s
Rem Main nicht gelistet
Rem Quicksort-Unterroutine
Procedure Sort
  T(Sn%+1)=10000
  Sf%=1
  Sl%=Sn%
  Dim Ss%(30)
  Sc%=0
  Goto Schleifentest
  Schleifenanfang:
  Sd%=Sl%+1
  Si%=Sf%
  Sm%=Int((Sl%-Sf%)/2)+Sf%
  T=T(Sm%)
  Swap T(Sm%),T(Sf%)
  Repeat
    Repeat
      Inc Si%
      Until T(Si%)>=T
      Repeat
        Dec Sd%
      Until T(Sd%)<=T
      If Sd%>Si% Then
        Swap T(Si%),T(Sd%)
      Endif
    Until Sd%<=Si%
    T(Sf%)=T(Sd%)
    T(Sd%)=T
    If Sd%-Sf%<Sl%-Sd% Then
      Ss%(Sc%+1)=Sd%+1
      Ss%(Sc%+2)=Sl%
      Sl%=Sd%-1
    Else
      Ss%(Sc%+1)=Sf%
      Ss%(Sc%+2)=Sd%-1
      Sf%=Sd%+1
    Endif
    Sc%=Sc%+2
    Schleifentest:
    If Sf%<Sl% Then
      Goto Schleifenanfang
    Endif
    If Sc%>0 Then
      Sl%=Ss%(Sc%)
      Sf%=Ss%(Sc%-1)
      Sc%=Sc%-2
      Goto Schleifentest
    Endif
  Return
```

Bsp. 9: Iteratives Quicksort

```
Rem Quicksort für Reals: 7.7s
Rem Main nicht gelistet
Rem Quicksort-Unterroutine
Procedure Sort
  Dim Ss%(15,1)
  Sc%=1
  Ss%(Sc%,0)=1
  Ss%(Sc%,1)=Sn%
  Repeat
    Sl%=Ss%(Sc%,0)
    Sr%=Ss%(Sc%,1)
    Sc%=Sc%-1
    Repeat
      Si%=Sl%
      Sd%=Sr%
      T=T(Int(Sl%+Sr%)/2)
      Repeat
        While T(Si%)<T
          Inc Si%
        Wend
        While T<T(Sd%)
          Dec Sd%
        Wend
```

Rekursives Quicksort

```
Rem Quicksort für Strings: 7,2s
Sortnumber%=1000
Dim Sortarray$(Sortnumber%)
For X%=1 To Sortnumber%
  For Y%=1 To 10
    Sortarray$(X%)=Sortarray$(X%)+Chr$(Random(25)+65)
  Next Y%
Next X%
Gosub Anzeige("Unsortiert")
Print "Start";Chr$(7)
Uhrzeit1=Timer
Gosub Sort(1,Sortnumber%)
Print "Ende";Chr$(7)
Uhrzeit2=Timer
Gosub Anzeige("Sortiert")
Print (Uhrzeit2-Uhrzeit1)/200;" Sekunden"
End
Procedure Anzeige(T$)
  Print T$
  For X%=1 To Sortnumber%
    Print Sortarray$(X%),
  Next X%
Return
Rem Quicksort-Unterroutine
Procedure Sort(Sortleft%,Sortright%)
  Sortincrement%=Sortleft%
  Sortdecrement%=Sortright%
  Sorttemporary$=Sortarray$(Int(Sortleft%+Sortright%)/2)
  Repeat
    While Sortarray$(Sortincrement%)<Sorttemporary$
      Inc Sortincrement%
    Wend
    While Sorttemporary$<Sortarray$(Sortdecrement%)
      Dec Sortdecrement%
    Wend
    If Sortincrement%<=Sortdecrement% Then
      Swap Sortarray$(Sortincrement%),Sortarray$(Sortdecrement%)
      Inc Sortincrement%
      Dec Sortdecrement%
    Endif
  Until Sortincrement%>Sortdecrement%
  If Sortincrement%<Sortright% Then
    Gosub Sort(Sortleft%,Sortdecrement%)
  Endif
  If Sortincrement%<Sortright% Then
    Gosub Sort(Sortincrement%,Sortright%)
  Endif
Return
```

Abb. 2: Rekursives Quicksort für String-Arrays

```
      If Si%<=Sd% Then
        Swap T(Si%),T(Sd%)
        Inc Si%
        Dec Sd%
      Endif
    Until Si%>Sd%
    If Si%<Sr% Then
      Inc Sc%
      Ss%(Sc%,0)=Si%
      Ss%(Sc%,1)=Sr%
    Endif
    Sr%=Sd%
  Until Si%>=Sr%
Until Sc%=0
Return
```

5. Befehlsvorrat

Die nachfolgende Übersicht berücksichtigt zwar bereits die neuen Befehle von GFA-BASIC Version 2.0, doch könnten sich noch kleine syntaktische Korrekturen ergeben, weil die Neuversion kurz vor Redaktionsschluß eintraf und deshalb noch nicht alle Befehlserweiterungen im Detail erprobt werden konnten. Wir werden deshalb ggf. zu einem späteren Zeitpunkt eine ergänzte Übersicht veröffentlichen.

Allgemeines

Länge des BASIC-Interpreters:
ca. 60K
Länge für Programm + Daten:
max. 800K bei Atari 1040 ST
Länge einer Programmzeile:
max. 255 Zeichen
Befehlsseparator:
Nur Return-Zeichen (nicht „:"!) und somit nur 1 Befehl pro Programmzeile
Länge eines Variablennamens:
max. 255 Zeichen

Variablenkennung

Varname = Real-Variable
Varname% = Integer-Variable
Varname! = Boolesche Variable
Varname$ = String-Variable
*Varname = Real-Zeiger
*Varname% = Integer-Zeiger
*Varname! = Boolescher Zeiger
*Varname$ = String-Zeiger
(Real-Variablen haben keine Typkennung)

Datentypen

Fließkommazahl:
Bereich: +/-9.9999999999E+/-154
11stellige Mantisse, 3stelliger Exponent
intern 6 Bytes

Ganzzahl (%):
Bereich: +/-2147483647
intern 4 Bytes

Wahrheitswert (!):
Bereich: -1 = wahr, 0 = falsch
intern 2 Bytes; bei Arrays 1 Bit

Zeichenkette ($):
Stringlänge: max. 32767 Zeichen
Zeichenbereich: ASCII 0-255
intern 1 Byte pro Zeichen

Zeiger (* vorangestellt):
Ganzzahl

Felder:
Arrays, z.B. A(10,20), können mehr als 65535 Feldelemente umfassen und zudem mehrdimensional sein. Boolesche Felder sind immer gepackt.

Operatoren

() (Klammern)
↑ (Expoment)
+ (Vorzeichen)
- (Vorzeichen)
* (Fließkomma-Multiplikation)
/ (Fließkomma-Division)
DIV (Ganzzahl-Division)
MOD (Ganzzahl-Divisionsrest)
+ (plus: Real oder Integer)
- (minus: Real oder Integer)
= (gleich: Zahl oder String)
(„=" ist neben „LET =" auch Wertzuweisung)
< (kleiner: Zahl oder String)
> (größer: Zahl oder String)
<= (kleiner/gleich: Zahl oder String)
>= (größer/gleich: Zahl oder String)
<> (ungleich: Zahl oder String)
== (etwa gleich: nur Zahl)
+ (String-Verkettung)
NOT (logische Negation)
AND (logische Konjunktion)
OR (logische Disjunktion)
XOR (logische Kontravalenz)
IMP (logische Implikation)
EQV (logische Äquivalenz)

Konstanten

PI = 3.1415926536
TRUE = -1
FALSE = 0

BASEPAGE
VDIBASE
(vom Speicherausbau abhängig)

Systembefehle

```
NEW
CLEAR
CLR var [,var...]
RUN
STOP
END
CONT
QUIT
SYSTEM
REM (oder ' oder !)
EDIT
LIST
DEFLIST aexpr
TRON
TROFF
```

Programmfluß

```
FOR var = a [DOWN]TO e [STEP s]
...
NEXT var

GOTO label (oder @ label)
...
label:
...

DO
...
LOOP

REPEAT
...
UNTIL bedingung

WHILE bedingung
...
WEND

IF bedingung [THEN]
...
[ELSE ...]
ENDIF

EXIT IF bedingung

GOSUB name[(expr [,expr])]
...
ON expr GOSUB name [,name...]
...
PROCEDURE name [(var [,var...])]
LOCAL var [,var...]
...
RETURN

DEFFN name[(varliste)] = expr
FN name[(expr [,expr])]
```

Mathematik

```
SQR(aexpr)
EXP(aexpr)
ATN(aexpr)
COS(aexpr)
SIN(aexpr)
TAN(aexpr)
LOG(aexpr)
LOG10(aexpr)
INT(aexpr)
FIX(aexpr)
FRAC(aexpr)
TRUNC(aexpr)
ABS(aexpr)
SGN(aexpr)
EVEN(aexpr)
ODD(aexpr)
RANDOM(aexpr)
RND[(aexpr)]
ADD var,n
SUB var,n
MUL var,n
DIV var,n
DEC var
INC var
ARRAYFILL feld(),n
```

Stringbefehle

```
ASC(sexpr)
LEN(sexpr)
VAL(sexpr)
VAL?(sexpr)

LEFT$(svar[,n])
RIGHT$(svar[,n])
MID$(svar,a[,n])
MID$(svar,a[,n])=sexpr
INSTR([n,]a$,b$)
INSTR(a$,b$[,n])
SPACE$(aexpr)
STRING$(n,svar)
STRING$(n,c)
CHR$(aexpr)
STR$(aexpr)
UPPER$(svar)
FRE(aexpr)

HEX$(aexpr)
OCT$(aexpr)
BIN$(aexpr)
&Hexpr → Hex
&Oexpr → Oktal
&Xexpr → Binär

LSET svar=string
RSET svar=string
CVI(sexpr)
CVL(sexpr)
CVS(sexpr)
CVF(sexpr)
CVD(sexpr)

MKI$(n)
MKL$(n)
MKS$(n)
MKF$(n)
MKD$(n)
```

Felder und Zeiger

```
DIM var(indizes) [,var(indizes),...]
DIM? (feld())
ERASE feld()
TYPE (ptr)
SWAP var1,var2
MAX(expr [,expr...])
MIN(expr [,expr...])
ARRPTR(var)
VARPTR(var)
OPTION BASE 1
```

Zuweisung

```
[LET] var = expr
DATA [const[,const]...]
READ var [,var...]
RESTORE label ... label:
```

Eingabe

```
INKEY$
INPUT ["text";] var [,var...]
LINE INPUT ["text";] var [,var...]
FORM INPUT n, svar
FORM INPUT n AS svar
INP(n)
INP?(n)
```

Ausgabe

```
PRINT [AT(s,z)] [;] [expr [,] [;] ['']...]
PRINT USING "format", liste [;]
DEFNUM n
WRITE [expr [,expr...] [;]
POS(n)
TAB(n)
CRSCOL
CRSLIN

LLIST
LPOS(n)
LPRINT [expr [,] [;] ['']]
LPRINT USING "format", liste [;]
OUT aexpr,a
OUT?(n)
HARDCOPY
```

Directory-Befehle

```
DIR ["filespec" [TO "file"]]
FILES ["filespec" [TO "file"]]
DIR$(n)
MKDIR "directory"
RMDIR "directory"
CHDIR "directory"
CHDRIVE n
DFREE(n)
```

Allgemeine Dateibefehle

```
LOAD "filename"
SAVE "filename"
PSAVE "filename"
LIST "filename"
BLOAD "filename" [,adresse]
```

BSAVE "filename", adresse, len
BGET [#]i,adr,cnt
BPUT [#]i,adr,cnt
CHAIN "filename"
EXEC [(]flag,nam,cmd,env[)]
KILL "filespec"
EXIST("filespec")
NAME "oldfile" AS "newfile"

Sequentielle und Random-Dateien

OPEN "modus", [#]n, "filename" [,len]
FIELD [#]n,expr AS svar ...
INPUT #n, var [,var...]
LINE INPUT #n, var [,var...]
INPUT$(x[,#n])
PRINT #n [,expr [,] [;] ['']...]
PRINT #n, USING "format", liste [;]
WRITE #n [,expr [,expr...] [;]
INP(#n)
OUT #n,a
GET [#]n[,i]
PUT [#]n[,i]
SEEK [#]n,aexpr
RELSEEK [#]n,aexpr
LOC([#]n)
LOF([#]n)
EOF([#]n)
CLOSE [[#]n]

Grafikbefehle

CLS [#i]
COLOR c
SETCOLOR i,r,g,b
SETCOLOR i,n
GRAPHMODE n
DEFLINE [s], [b], [a], [e]
LINE x0,y0,x1,y1
DRAW [TO] x0,y0
DRAW x0,y0 TO x1,y1 [TO x2,y2...]
PLOT x,y
POINT(x,y)

BOX x0,y0,x1,y1
PBOX x0,y0,x1,y1
RBOX x0,y0,x1,y1
PRBOX x0,y0,x1,y1

CIRCLE x,y,r [,phi0,phi1]
PCIRCLE x,y,r [,phi0,phi1]

ELLIPSE x,y,rx,ry [,phi0,phi1]
PELLIPSE x,y,rx,ry [,phi0,phi1]

DEFMARK [c], [a], [g]
POLYLINE n,x(),y() [OFFSET x0,y0]
POLYFILL n,x(),y() [OFFSET x0,y0]
POLYMARK n,x(),y() [OFFSET x0,y0]

FILL x,y
DEFFILL [c], [a], [b]
DEFFILL [c], a$
DEFTEXT [c], [s], [r], [g]
TEXT x,y, [!,] string
GET x0,y0,x1,y1,a$
PUT x0,y0,a$ [,modus]
SGET svar
SPUT svar
SPRITE a$ [,x,y]

Fensterbefehle

MENU feld()
MENU KILL
MENU OFF
MENU(n)
MENU n,x
ON MENU
ON MENU GOSUB name
ON MENU KEY GOSUB name
ON MENU MESSAGE GOSUB name
ON MENU IBOX a,x,y,b,h GOSUB name
ON MENU OBOX a,x,y,b,h GOSUB name
OPENW n [,x,y]
OPENW 0
INFOW n, "info"
TITLEW n, "titel"
FULLW n
CLEARW n
CLOSEW n
CLOSEW 0

ALERT a,mstring, b, bstring, var
FILESELECT "filespec", "filename", sexpr

Mausbefehle

DEFMOUSE n
DEFMOUSE a$
MOUSE x,y,k
MOUSEX
MOUSEY
MOUSEK
HIDEM
SHOWM
ON MENU BUTTON c,m,s GOSUB name

Tonbefehle

SOUND stim,laut,note,octa [,verz]
SOUND stim, laut, #period, [,verz]
WAVE stim,huell,form,dauer,verz

Zeitbefehle

DATE$
TIME$
SETTIME timestring,datestring
TIMER
PAUSE aexpr

Maschinenbefehle

PEEK(aexpr)
DPEEK(aexpr)
LPEEK(aexpr)
POKE aexpr,n
SPOKE aexpr,n
DPOKE aexpr,n
SDPOKE aexpr,n
LPOKE aexpr,n
SLPOKE aexpr,n
C:var (parameterliste)
CALL var
CALL var (parameterliste)
MONITOR [n]
HIMEM
RESERVE n
BMOVE scr,dst,cnt

Unterbrechnung und Fehler

ON BREAK
ON BREAK CONT
ON BREAK GOSUB name

ERR
ERROR n
FATAL
ON ERROR
ON ERROR GOSUB name
RESUME
RESUME NEXT
RESUME label

Mark-Williams-C-Compiler

Ein professionelles Entwicklungssystem für den Atari ST

getestet von Dieter Geiß

Beim Auspacken des Programmpakets, das uns von dem Importeur Alverdes Softline in Oberkirch zu Testzwecken zur Verfügung gestellt wurde, bekommt man zuerst einen Schock. Zwei randvolle, doppelseitig formatierte Disketten mit 86 Dateien sowie ein Handbuch von knapp 650 Seiten erwecken den Eindruck, als ob es alles andere als einfach wäre, ein C- oder Assemblerprogramm für den Atari zu schreiben. Doch nach kurzer Eingewöhnungszeit sieht alles nur noch halb so schlimm aus.

1. Handbuch

Da es sich um ein amerikanisches Produkt handelt, ist es durchweg in Englisch gehalten, was aber heutzutage keinen Programmierer mehr abschrecken dürfte. Es umfaßt drei Teile, wobei sich der erste Teil (543 Seiten) nach der Einführung vor allem mit der Bedienung der verschiedenen Programmteile befaßt. Bei ihnen handelt es sich um eine Mikro-Shell, den C-Compiler/Linker, den symbolischen Debugger und den Assembler. Ebenfalls im ersten Teil befindet sich hinter der Übersicht über die Fehlermeldungen ein alphabetisches Lexikon, das kaum Fragen offenläßt. Hier findet man schnell alles, wonach man beim Programmieren immer sucht. Dazu gehören:

– alle C-Funktionen wie abort, abs, acos, ... unlink, write
– alle BIOS-, XBIOS- und GEMDOS-Funktionen
– eine Kurzbeschreibung mit allen Optionen für die verschiedenen Programme
– allgemeine C-Begriffe wie Register, Struct, Extern usw.
– Atari-spezifische Begriffe wie Keyboard-Layout usw.

Der zweite Teil (25 Seiten) behandelt ein spezielles Programm mit Namen „Make". Mit dessen Hilfe ist es möglich, viel Zeit beim Schreiben komplexer Programme zu sparen. Darauf soll hier jedoch nicht weiter eingegangen werden. Nur soviel sei gesagt, daß Programme und Module automatisch übersetzt werden, wenn sich z.B. das Datum bzw. die Uhrzeit einer Header-Datei geändert hat. Dann werden alle Module älteren Datums, die diese Header-Datei enthalten, durch das Make-Kommando neu übersetzt.
Teil 3 (75 Seiten) schließlich beschreibt den Editor. Es handelt sich dabei um eine Version des zumindest in Amerika populären EMACS-Editors namens Micro-EMACS. Auf ihn soll später eingegangen werden.

2. Installation

Bevor man effektiv mit dem System arbeiten kann, sollte man sich die Originaldisketten kopieren. Dann folgt eine Prozedur, die sich eine halbe Stunde hinziehen kann – die Installation auf Festplatte bzw. auf Disketten. Dabei wird deutlich, daß das gesamte System ziemlich voluminös ist. Es wird im Handbuch ausdrücklich darauf hingewiesen, daß man entweder eine Festplatte oder zwei doppelseitige Laufwerke haben muß, um überhaupt arbeiten zu können. Besitzer *eines* doppelseitigen Laufwerks werden nun den Kopf schütteln und denken: Kommt für mich nicht in Frage! Dies ist jedoch zum Glück nicht zutreffend, weil die Entwickler des Systems maßlos übertreiben, was die benötigte Hardware anbelangt. Bei meinem ersten Compilierungslauf überspielte ich einfach sämtliche Dateien, die wirklich wichtig sind, auf eine RAM-Disk, die ich mit 500K anlegte. Es lief dann alles einwandfrei! Wenn man sich eine eigene Batch-Datei schreibt, welche die Programme optimal verteilt, kommt man bequem mit einer RAM-Disk und einem doppelseitigen Laufwerk aus.

3. Programme

3.1. Mikro-Shell

Zwei Umgebungen werden dem Programmierer angeboten: Er kann seine Programme unter der GEM-Oberfläche übersetzen lassen oder die mitgelieferte Mikro-Shell benutzen. Es handelt sich um eine UNIX-ähnliche Umgebung, die in dieser Version schon 37 Kommandos kennt. Die bekanntesten sind
date (Datum und Uhrzeit einstellen),
mv (Datei umbenennen),
cp (Dateien kopieren),
rm (Dateien löschen),
ls (Directory-Listing),
cat (Dateien anzeigen) usw.
UNIX-Kenner werden ihre wahre Freude damit haben. Einige der komplexeren Befehle sind allerdings Programme, die erst geladen werden müssen. Dieser Umstand bringt es mit sich, daß man bei Verwendung von Diskettenlaufwerken durchaus einige Sekunden (8.4 s) warten muß, bevor die lakonische Antwort „lk: not found" erscheint, die anzeigt, daß der Befehl nicht gefunden wurde. Dies tritt auch immer dann auf, wenn man sich vertippt (lk statt ls). Arbeitet man dagegen mit einer RAM-Disk, so ist die Wartezeit erträglich (0.2 s). Der Vorteil einer DOS-Shell wurde schon in einem anderen Artikel (Peeker 10/86) erwähnt. Wichtig ist vor allem das automatische Abarbeiten von Kommando-Dateien (Batch-Dateien) und der schnelle Wechsel zwischen Editor, Compiler/Assembler sowie Linker oder Library Utility. Arbeitet man unter GEM, so muß jedesmal der

Bildschirm neu aufgebaut werden, wenn man den Editor verläßt oder der Compiler eine Fehlermeldung bringt.

3.2. Editor

Er stellt mehr einen Programm- denn einen Texteditor dar. Er arbeitet vollständig unter TOS, kommt also ohne Maus und Menüs aus. Man kann dennoch in maximal 11 verschiedenen (Text-)Fenstern gleichzeitig editieren – eine der hervorstechendsten Eigenschaften. Des weiteren ist es möglich, eigene Tastatur-Makros zu definieren. In einem speziellen Modus kann man sogar Kommandos an die umgebende Shell abgeben. So ist es ohne weiteres möglich, z.B. 10 verschiedene Module gleichzeitig zu editieren und jeweils das fertig editierte Modul übersetzen zu lassen.
Der Clou ist allerdings das Vorhandensein des kompletten Source-Listings des Editors auf einer der zwei System-Disketten. Er befindet sich in einer Text-Library, aus der die einzelnen Text-Dateien mit Hilfe der Archiv-Utility extrahiert werden können.
Die Scroll-Geschwindigkeit des Editors läßt allerdings sehr zu wünschen übrig. Das liegt daran, daß er so allgemein geschrieben wurde, daß er auch für andere Computer schnell anzupassen ist. Da allerdings nicht jeder Computer Befehle kennt, um einen Teil des Textbildschirms zu scrollen, wurde das Problem gelöst, indem jeweils der gesamte Bildschirminhalt neu gezeichnet wird, was einen enormen Geschwindigkeitsverlust zur Folge hat. Man kann aber auch einen anderen Editor verwenden, wie z.B. den Metacomco Screen Editor oder den ST Editor von Omikron Software.

3.3. Compiler/Linker

Der Compiler kennt den vollen Kernighan-Ritchie-C-Standard mit einigen Erweiterungen. So erlaubt er unter anderem die Zuweisungen von Structs, die Übergabe von Structs als Wertparameter und ein Struct als Funktionswert zu definieren. Der Compiler selbst besteht aus maximal 5 (!) Pässen. Eine erstaunlich große Zahl, die daher rührt, daß er ursprünglich für andere Betriebssysteme entwickelt wurde, die früher nicht sehr viel RAM-Speicher zur Verfügung hatten, aber eine ausreichende Kapazität an externem Speicher. So belegt der Compiler nur einen kleinen Bereich des Hauptspeichers, erzeugt beim Compilieren aber eine Menge von Zwischendateien, die sich auf einem schnellen Datenträger befinden sollten, also einer Festplatte oder RAM-Disk. Die fünf Pässe sind:

- cpp: Präprozessor
- cc0: Parser
- cc1: Code Generator
- cc2: Optimizer/Object Generator
- cc3: Disassembler

Der **Präprozessor** behandelt die entsprechenden Anweisungen wie „#include“ oder „#ifdef“ usw. Es ist möglich, die erzeugte Zwischendatei resident zu halten, was bei der Fehlersuche wertvolle Hilfe leisten kann.
Ein Problem haben die Compilerbauer aber nicht bedacht. Offensichtlich haben sie noch nie ein GEM-Programm übersetzen lassen, das Resource-Dateien verwendet. Das Resource-Construction-Set erzeugt bekanntlich eine Header-Datei (Suffix „.H“), in welcher sich die Konstanten für die Resourcen befinden. Bindet man diese in sein Programm mittels eines „#include“-Befehls ein, so versagt der Compiler seinen Dienst. Grund: Das Resource-Construction-Set hängt an die Header-Datei als letztes Zeichen ein Ctrl-Z (ASCII 26) als EOF-Marke an, was durchaus nicht unüblich ist. Dieses Zeichen allerdings verkraftet der Präprozessor nicht. Der Omikron-Editor hängt sogar bei allen Text-Dateien, die er erzeugt, ein Ctrl-Z an. Dieser Fehler des Präprozessors sollte unbedingt behoben werden, da man sonst gezwungen ist, mittels eines Diskette neditors das Ctrl-Z z.B. in ein Leerzeichen umzuwandeln. Keine sehr schöne Methode!
Der **Parser** benutzt den rekursiven Abstieg, um einen Zwischencode zu erzeugen, der sowohl von der Sprache (C) als auch vom Mikroprozessor, für den der Code zu erzeugen ist (68000), unabhängig ist.
Der **Code Generator** verwandelt den erzeugten Parser-Baum in Maschinencode um. Die Erzeugung selbst ist tabellengesteuert, wobei es Einträge für jeden Operanden und jede Adressierungsart gibt. Er soll recht guten Code erzeugen (siehe unten).
Der **Peephole Optimizer** eliminiert Sprünge zu Sprüngen, Sprünge zum nächsten Befehl usw., bis alle Adressen aufgelöst sind. Dabei wird der fertige Objektcode erzeugt.
Der **Disassembler** ist optional. Er erzeugt ein Assembler Source Listing, das dazu benutzt werden kann, den erzeugten Code zu untersuchen.

Trotz der vielen Pässe ist die Bedienung des Compilers recht einfach. Dazu existiert ein Compiler-Treiber, ein Programm namens „cc“. Es ruft nicht nur die einzelnen Pässe des Compilers, sondern auch den Linker auf. Dabei kann eine Unzahl von Parametern (etwa 30) übergeben werden, die den Ablauf des Compilierens steuern.
Der Linker übernimmt die erzeugte Codedatei sowie weitere Objektmodule, die schon übersetzt worden sind. Diese kann man auch in Bibliotheken einbinden. Als Standardbibliotheken werden eine C-, eine Mathematik- sowie die VDI- und AES-Bibliotheken mitgeliefert. Mit dem Programm „ar“ können sowohl Objekt- als auch Text-Bibliotheken (siehe Micro-EMACS) erzeugt werden.

3.4. Debugger

Mit Hilfe des Debuggers ist es möglich, seine Programme auf Maschinenebene zu testen. Die Unzahl von Kommandos, die der Debugger kennt, läßt seine Bedienung etwas schwierig und unübersichtlich erscheinen. Man kann Breakpoints setzen, Befehle in Einzelschritten abarbeiten und mit Symbolen adressieren. Die Symbole befinden sich jeweils noch in der Code-Datei; sie können aber später auch entfernt werden. Zu den weiteren Merkmalen des Debuggers gehören Stack Backtracing und die Möglichkeit, alle auftretenden Daten fast beliebig zu formatieren, also z.B. als Fließkommazahl zu interpretieren.

3.5. Assembler

Obwohl C eine Sprache ist, bei deren Benutzung man fast immer auf Assembler verzichten kann, kommt es manchmal vor, daß man kleine Assemblerroutinen schreiben muß. Diese können mit anderen Objekt-Modulen gelinkt werden. Sowohl der Compiler als auch der Assembler erzeugen das gleiche Objekt-Format. Es gibt zwar keine bedingte Assemblierung und keine Makros, aber z.B. lokale Labels.

4. Geschwindigkeitstests

Die beiden wohl wichtigsten Kriterien bei der Anschaffung eines Compilers sind die Compilier-, Link- und Laufzeiten der erzeugten Programme. Es gibt Compiler, die bis zu zehnmal schneller sind als andere oder die einen Code erzeugen, der dreimal so schnell ist. Anhand von vier verschiedenen Compilern sollen hier die Zeiten zum Übersetzen und Linken sowie die Laufzeit der Programme unter die Lupe genommen werden. Bei den Compilern handelt es sich um den Alcyon-C-Compiler, der aus dem Entwicklungspaket von Atari bekannt sein dürfte, den Megamax-C-Compiler, der von Application Systems in Heidelberg vertrieben wird, den Lattice-C-Compiler von Metacomco und natürlich den Mark-Williams-C-Compiler. Der Preis ist der Vollständigkeit halber mit angegeben. Dabei handelt es sich jeweils um den Preis für das ganze Entwicklungssystem.

Zwei verschiedene Programme wurden gewählt, um die Systeme zu testen. Das erste Programm (siehe **Tabelle 1**) ist ein Kartenspiel-Simulationsprogramm, bei dem vor allem viel gerechnet wird. Es errechnet für das Spiel „Blackjack", das einzige Spiel, bei dem der Spieler der Spielbank überlegen sein kann, die optimale Stragegie zum Ziehen der Karten. Die Laufzeiten beziehen sich dabei auf 1000 gespielte Partien. Das Programm hat eine Länge von 730 Zeilen, wobei 189 Zeilen aus inkludierten Header-Dateien stammen. Die Laufzeiten sind nicht so typisch für die Compiler, da die Rechenoperationen teilweise aus den Bibliotheken stammen.

Bei dem zweiten Programm (siehe **Tabelle 2**) handelt es sich um den Dhrystone-Benchmark, also einen Benchmark-Test, der praktisch alle verschiedenen C-Konstrukte berücksichtigt (Zeitangaben in Sekunden).

Das Fehlen des Benchmarks für den Alcyon-C-Compiler ist dadurch zu erklären, daß der Test mit einem Bombeninferno endete. Wie mittlerweile bekannt sein dürfte, hat der Compiler des Entwicklungspakets etliche Macken, die schon vielen Programmierern Kopfzerbrechen gemacht haben.

Die Zahlen der Tests sprechen wohl eine deutliche Sprache. Der Megamax ist allen anderen Compilern um eine oder mehrere Nasenlängen voraus. Hätte er nicht eine böse Einschränkung und mindestens vier gravierende Fehler, so könnte man in dieser Hinsicht alle anderen Compiler vergessen.

5. Fazit

Der Mark Williams ist wohl der zweitbeste zur Zeit auf dem Markt befindliche Compiler. Das Handbuch, bei dem vor allem das Lexikon hervorsticht, ist recht brauchbar. Allerdings entbehrt es der Dokumentation der GEM-Bibliotheken, die man erst in anderen Büchern wie z.B. „Software-Entwicklung auf dem Atari ST" (Hüthig-Verlag) nachlesen muß.

Ein Leckerbissen dürfte das Paket für UNIX-Fans sein. Die ganze Komplexität des UNIX-Betriebssystems mit seinen vielen Funktionen und Optionen findet sich teilweise bei Mark Williams. Die Bibliotheken sind ebenfalls UNIX-kompatibel, womit es ein leichtes sein dürfte, auf dem Atari Software für UNIX-Maschinen zu entwickeln. Die Fehlerlosigkeit, die vor allem bei Compilern sehr wichtig ist, zeigt sich erst in sehr langen Programmen. Dabei schneiden vor allem der Alcyon-, aber auch der Megamax-Compiler schlecht ab. Die professionelle Programmierung des Mark Williams zeigt, daß man auch Compiler schreiben kann, die nicht bei einfachsten C-Konstrukten falschen Code erzeugen, der Programmierern schlaflose Nächte bereitet. Für die Erzeugung korrekten Codes bezahlt man dann gerne mit der längeren Wartezeit beim Übersetzen und Linken.

Tabelle 1: Übersetzungs- und Linkzeiten mit "Blackjack" (730 Zeilen)

Name	Preis	Comp.	Link.	Gesamt	Länge	Laufzeit
Megamax	595.–	8 s	13 s	21 s	12949	15.39 s
Alcyon	969.–	68 s	46 s	114 s	18506	12.19 s
Mark Williams	600.–	33 s	66 s	99 s	17238	14.36 s
Lattice	348.–	64 s	40 s	104 s	33694	36.15 s

Die Einheit der Länge ist Bytes.

Tabelle 2: Ausführungszeiten mit Dhrystone-Benchmark

Name	Sekunden	Anzahl Benchmarks
Megamax	44	1121
Alcyon	—	—
Mark Williams	55	896
Lattice	73	684

Drucker-Produkte

Drucker-Zwischenspeicher

Mit der CIRTECH Puffer-Box der Fa. Semjan Computer Systeme kann an jeden Drucker ein zusätzlicher Drucker-Zwischenspeicher mit 256K oder 512K RAM angeschlossen werden. Die „Cachebox", die bei fast allen Computern eingesetzt werden kann, speichert die Daten, die vom Computer an den Drucker geschickt werden, und gibt sie schrittweise an den Drucker weiter. Diese Speichertechnik ermöglicht es, bis zu 200 bzw. 400 DIN-A4-Seiten automatisch zwischenzuspeichern und an den Drucker weiterzugeben. Die Puffer-Box verarbeitet bis zu 20.000 Zeichen/s und ermöglicht bis zu 65.000 Kopien der Originaldaten. Die Cachebox ist zu allen gängigen Computern (Apple, Atari, IBM, HP, Tandy etc.) und fast allen Druckern (Epson, NEC, Brother, ITOH etc.) kompatibel. Sie erlaubt ein ungehindertes Arbeiten am Computer ohne langwieriges Warten auf das Ende eines Druckvorganges. Die wertvolle Rechenzeit des Computers wird besser genützt, die Rechnerkapazität kann unabhängig von der geringeren Druckerkapazität ausgeschöpft werden.
Die Puffer-Box wird mit 256 oder 512K-RAM in drei Ausführungen geliefert: seriell, parallel, seriell und parallel.
Die Preise incl. Mwst. und Handbuch betragen für die serielle oder parallele Ausführung ca. DM 807,– (256K) bzw. DM 1215,– (512K), für die seriell/parallele Ausführung ca. DM 989,– (256K) bzw. DM 1397,– (512K).
Bezugsquelle: Semjan Computer Systeme, Frankfurt

Epson-Nachrüstsatz

Für die weit verbreiteten Epson-Matrixdrucker MX 80/100, RX 80/100, FX 80/100 und JX 80 bietet die Fa. Mayer Computersysteme einen Nachrüstsatz an, der diese Drucker NLQ-fähig (NLQ = Near Letter Quality) und leichter bedienbar macht. Rund 160 verschiedene Schriftarten können über 3 Tasten direkt am Drucker angesteuert werden. Einzelblatteinzug ist als Zusatzausrüstung lieferbar. Interessant ist diese Lösung für alle Druckerbesitzer, die sich keinen neuen Drucker leisten können oder wollen, aber ein verbessertes Schriftbild wünschen.
Der Umbausatz kostet ca. DM 249,–, die Umrüstung incl. Versandkosten ca. DM 295,–.
Bezugsquelle: Friedhelm Mayer Computersysteme, Berlin

Traktor für Okidata-Drucker

Okidata bietet als Option für die OKI-Drucker der Serien Microline 192, 193, 292, 293 und 294 einen neuartigen Traktor-Aufsatz an, der speziell für diese Geräte entwickelt wurde. Der Traktor ermöglicht einen äußerst exakten Zeilenvorschub in beide Richtungen. Er ist v.a. für Anwendungen konzipiert, bei denen es auf präzise Druckergebnisse ankommt, z.B. bei Grafik oder Plotter-Simulationen. Neu ist die Einspann-Vorrichtung, die es ermöglicht, das Papier von der Frontseite des Geräts in beide Führungswalzen einzuspannen.
Der Preis beträgt für die schmale Version ca. DM 489,–, für die breite Version ca. DM 614,–.
Bezugsquelle: Okidata GmbH, Düsseldorf

Comtest-Schnittstellentester

Der Comtest-Schnittstellentester, der von der schwedischen Hardwarefirma Jet-Computer hergestellt wird, ist ein nur handtellergroßes Gerät zum Testen der V24/RS232C-Schnittstelle und Stromschleifen. Das Gerät ist unentbehrlich für alle, die Datenkommunikationsgeräte wie Modems, Terminals, Drucker etc. installieren, reparieren und benutzen. Es eignet sich zur Prüfung der Signalpegel, zur Kreuzkopplung und Verbindung zweier DTEs, zum Gerätetest durch digitalen Loopback und zum Aufspüren von Differenzspannungen. Alle 25 Leitungen werden laufend überwacht und können jederzeit unterbrochen, überbrückt und getauscht werden. 52 LEDs zeigen den Pegel der 23 Signalleitungen an. Das Gerät ermöglicht 2-Stufen-Differenzspannungstest zwischen DTE und DEE und testet bidirektional Stromschleifen in 4 Stufen (10, 20, 40, 60mA); da das Gerät aus der Schnittstelle mit Strom versorgt wird, ist keine Batterie nötig. Der Tester besitzt Direktanschluß an DTE/DEE und Kabel; Einschleifen in Centronics-Druckerkabel für IBM-PC ist möglich. Comtest wird mit 8 Steckkabeln, Schutztasche und deutscher Gebrauchsanleitung geliefert.
Der Preis beträgt ca. DM 388,–.
Bezugsquelle: Ing.Büro Sauer, Aachen

Apple-Produkte

Macintosh und VAX-Super-Mini

Mit dem Datenübertragungs-Programm MaxVax von der MKV GmbH, Mannheim, können EDV-Laien jetzt auch die Rechnerleistung eines VAX-Super-Minis nutzen. MacVax ermöglicht die Datenübertragung zwischen den bedienerfreundlichen Macintosh Personal Computern und VAX/VMS-Systemen von Digital Equipment. Über das AppleTalk-Netzwerk wird der Macintosh an einen VAX-Rechner angeschlossen. Durch Übertragungs- und Konvertierungsmöglichkeiten können Daten aus dem Super-Mini an den Macintosh weitergegeben werden und stehen somit auch EDV-Laien jederzeit zur Information und zur Bearbeitung zur Verfügung.
Der VAX-Rechner bietet für den Macintosh folgende ergänzende Funktionen: Mailbox, Bibliothek, Archivierung, Print-Server. Um die Datenübertragung zu ermöglichen, verwendet der Macintosh Personal Computer die Terminalemulation und der VAX-Rechner das Datenübertragungsprogramm MacVax. MacVax ist genau auf MacTerminal angestimmt. Es erlaubt damit die Datenübertragung im Macintosh-Format und bietet Konvertierungsmöglichkeiten zwischen dem Macintosh und dem VMS-Format. Alle wichtigen Funktionen einschließlich Hilfestellungen sind über einfache Kommandos zu bedienen.
Bezugsquelle: MKV GmbH, Mannheim

Festplatte für den Apple GS

Am 15.9.86 stellte die Firma Apple weltweit den Apple GS, ihren neuen 16-Bit-Rechner, vor. Der Computer mit dem 65C816-Prozessor verfügt über 8 Steckplätze und wurde als offener Rechner konzipiert. Damit können auch im Apple-II-Nachfolgemodell Erweiterungssteckkarten zum Einsatz kommen.
Die Fa. Frank & Britting entwickelt v.a. Festplatten-Controller und ist auf diesem Gebiet zum Harddisk-Controller-Spezialist in Europa geworden. Schon für den Apple II/IIe/II+ bietet sie die Festplatten-Speichererweiterung MEGA-CORE und die mobile Datenbox MDB, jeweils mit 10 und 20 MByte Kapazität, an. Auch für den neuen Apple GS hat Frank & Britting bereits eine Festplatten-Speichererweiterung entwickelt, die MEGA-CORE GS. Beim Apple GS wird dazu das Original-Apple-Netzteil durch ein Einbau-Modul ersetzt, das aus einem verstärkten Netzteil mit Lüfter, Controller und einer 10- oder 20-MByte-Festplatte besteht. Die MEGA-CORE GS wird komplett mit Handbuch und der Software zur Anpassung der Betriebssysteme DOS, CP/M, ProDOS und Pascal geliefert.
Bezugsquelle: Frank & Britting, Forst
(Ggf. können Peeker-Leser auch diese Festplatte – wie schon die MDB 10 und 20 für den Apple IIe/II+ – zum Sonderpreis über den Hüthig-Software-Service beziehen.)

Statistik für den Apple

Die Statistik-Programme „APPSTAT" und „STATFAST" der amerikanischen Fa. STATSOFT ermöglichen es Apple-II- und Macintosh-Besitzern, auf ihren Computern einfache und höhere statistische Methoden anzuwenden. Das Spektrum der Programme reicht von deskriptiven Statistiken wie Mittelwert, Standardabweichung, T-Test, Korrelationen usw. bis hin zur multiplen Regressionsanalyse und mehrfaktoriellen Varianzanalyse. Eine analyseunterstützende grafische Darstellung von Daten in Form von Histogrammen und Streuungsdiagrammen ist ebenfalls möglich. Neben der direkten Dateneingabe über den programmeigenen Editor ist die Übernahme von Daten anderer Programme realisierbar. Das Programm ist durch Menüsteuerung sehr benutzerfreundlich. Es kostet für den Apple II ca. DM 269,–, für den Macintosh ca. DM 319,–.
Bezugsquelle: Loll + Nielsen, Hamburg

Super-Grafik-Karte APGRADE

Für den Apple II+, IIe und Kompatible hat die Fa. Computer Informations Systeme die Grafik-Karte APGRADE mit eigenem Grafikprozessor für hochauflösende Grafik (512 * 512 Punkte) entwickelt, die in jedem Slot (1-7) des Apple läuft. Die Karte wird in 4 Versionen mit verschiedenen Arbeitsspeichern geliefert: Die Grafikgrundplatinen APGRADE MB mit 64K Arbeitsspeicher (2/4 autonome Grafikseiten im Schwarz-Weiß-Modus) und APGRADE MBX mit 256K (8/16 Seiten S/W) und die beiden Farberweiterungsplatinen APGRADE CO mit 128K (6/12 Seiten S/W bzw. 2/4 Seiten in Farbe) und APGRADE COX mit 512K Arbeitsspeicher (24/48 Seiten S/W bzw. 8/16 autonome Grafikseiten in Farbe). Programmgesteuert sind zwei Auflösungen auswählbar: Im „interlaced scan" (Zeilensprungverfahren) 512 * 512 Punkte, im „non-interlanced scan" 256 * 512 Punkte. Die Schreib- und Anzeigeseiten sind unabhängig voneinander anwählbar, Schrift kann in verschiedener Größe und Ausrichtung dargestellt werden. Video- und Synchronisationssignale sind getrennt vorhanden (normal und invertiert). Durch Video-Softswitches kann auf Ein- oder Zwei-Monitorbetrieb umgeschaltet werden; der Anschluß eines Lichtgriffels ist möglich. Durch eine spezielle Farbmisch-Logik bietet APGRADE 8 Grundfarben bzw. Graustufen und 36 Mischfarben bzw. Graustufen an.
Die Anwendungsbereiche der Platine reichen von dem Erstellen von Kurven, Schaubildern, Diagrammen und Masken über Kontroll- und Meßwertanzeigen bei Steuer- und Regelungsprozessen bis zur Darstellung dreidimensionaler Körper, digitalisierter Bilder und CAD. Die getrennten Grafikseiten und die hohe Zeichengeschwindigkeit von 900.000 Punkte/s erlauben Echtzeitbewegungen.
Wird keine Grafik benötigt, kann die Karte als schnelle RAM-Disk mit 64K bzw. 256K (MB bzw. MBX) oder 192K bzw. 768K (CO bzw. COX) verwendet werden.
Die Installation ist problemlos: Die Karte wird einfach in einen Steckplatz (1-7) im Computer gesteckt. Monitore ab 15 MHz sind verwendbar.
Für Anwender und Programmierer wird eine ausführliche Dokumentation mitgeliefert: Anwenderhandbuch, Programmierhandbuch mit Programmlistings und Programmierbeispielen, v.a. in Assembler, und ein Servicehandbuch mit Hardware-Informationen. Ebenfalls im Lieferumfang enthalten sind Demo- und Hilfsprogramme, RAM-Disk-Treiberprogramme, Patches zur Applesoft-Grafikerweiterung als Basis-Grafik-Betriebssystem, nachladbare Zusatzmodule für diverse Highlevel-Grafikanweisungen, Dump-Modul für verschiedene Drucker und einfache Editoren zum Entwurf von Symbolen und Diagrammen.
Die Karten arbeiten mit verschiedenen Peripheriegeräten wie Drucker, Plotter, Grafik-Tableaus, Maus, Joystick zusammen. Ein umfangreiches Angebot an Zusatz-Software wird angeboten: Pascal-Unterstützung, Plotteranpassung, Grafikeditor usw.
Preise incl. Mwst.: APGRADE MB (64K) ca. DM 598,–, MBX (256K) ca. DM 910,–;
Farberweiterungsplatine CO (128K) ca. DM 885,-, COX (512K) ca. DM 1365,–.
Bezugsquelle: Computer Informations Systeme, Frankfurt

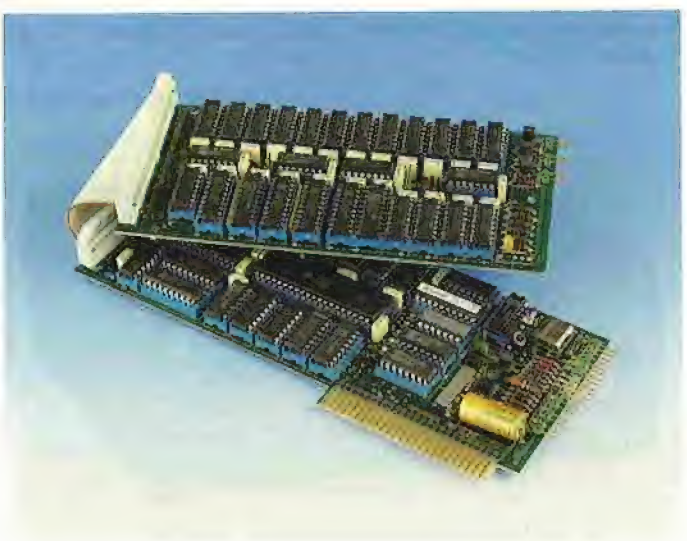
Apple-Schnittstelle für Citizen-Drucker

Citizen Europe bietet jetzt eine 7-Bit-Centronics-Schnittstelle an, mit der alle Citizen-Matrixdrucker und der Typenraddrucker Premiere 35 an den Apple II angeschlossen werden können. Nach Herstellerangaben wurde das Interface ausführlich mit dem Apple-Betriebssystem DOS 3.3 getestet. Es ist bei allen europäischen Citizen-Händlern zum Preis von ca. DM 260,– erhältlich.
Bezugsquelle: Citizen-Händler oder Burson-Marteller GmbH, Frankfurt

Apple-Free-Software

Ab 1.10.86 werden alle Programme für den Apple IIe/c/+ aus dem Haus der Buchhaltung zur Free-Software, d.h. die Programme sind frei kopierbar. Insgesamt führt das HDB etwa 120 Programmversionen für den Apple II, die jetzt als Free-Software gelten. Außerdem

▼



beinhalten die Preise für alle Softwareprodukte ab 1.10.86 bereits die Mehrwertsteuer, was gegenüber der bisherigen Berechnung eine Preissenkung um 14% bedeutet. Die Programmdisketten kosten jeweils DM 39,50.
Bezugsquelle: Haus der Buchhaltung, Duisburg

Firmenmitteilungen

Neuer Geschäftsführer bei Apple

Die Firma Apple Computer berichtet, daß mit Wirkung ab 1.10.86 Gerhard Jörg zum alleinvertretungsberechtigen Geschäftsführer der Apple Computer GmbH, München berufen wurde. Der 41jährige ist gleichzeitig Mitglied des Management-Teams Apple Computer Europe mit Sitz in Paris. Zuvor war Jörg beim EDV-Hersteller Burroughs GmbH in Frankfurt Mitglied der Geschäftsführung im Bereich Marketing und Vertrieb. „Mit der Berufung von Gerhard Jörg wollen wir die Bedeutung des deutschen Computermarktes für Apple unterstreichen", erklärte Roger Kermisch, General Manager Apple Europe. „In den kommenden drei Jahren wird Apple die Position insbesondere im kommerziellen Bereich weiter ausbauen. Deshalb hat sich Apple für einen Manager aus der Kommunikationsbranche entschieden", bekräftigte Kermisch das Apple-Engagement in Deutschland. Jörg bringt aus den letzten 12 Jahren als Geschäftsführer in der Branche umfangreiches Know-how im Aufbau und der Reorganisation von Marketing- und Vertriebsgesellschaften mit, insbesondere auch im Bereich des Computerfachhandels.
(Anm. d. Red.: Von dem neuen Apple-Geschäftsführer sollte man Flexibilität erwarten können, hat er doch allein in den letzten 4 Jahren gleich viermal seine Stellung gewechselt. 1983 mußte die HPR Computer GmbH, eine Tochter des Schweizer Büromaschinenherstellers Hermes Precisa, unter seiner Führung Konkurs anmelden. Jörg wechselte als Geschäftsführer zum Stuttgarter Büromaschinenhersteller Häussler, den er schon nach zwei Jahren aufgrund von Schwierigkeiten mit dem Unternehmensleiter wieder verließ. Beim Eschborner EDV-Hersteller Burroughs Deutschland GmbH war er in den letzten 10 Monaten für Marketing und Vertrieb zuständig. Es bleibt abzuwarten, ob Jörg der Mann ist, der in der Apple-Geschäftsführung endlich die nötige Kontinuität gewährleistet. Als Branchenkenner und Fachmann für Marketing und Vertrieb könnte er bei Apple sicher für einige Verbesserungen sorgen.)

Saturn-128K-RAM-Karte und erphi-Controller AFDC

Nach eingehenden Untersuchungen der Saturn-RAM-Disk-Software hat die Fa. erphi electronic festgestellt, daß die RAM-Disk unter CP/M 2.20 mit der Firmware des erphi-Controllers AFDC kompatibel ist, wenn bei der Installation richtig vorgegangen wird. Wichtig ist, daß bei der Anpassung der Programme INIT.COM bzw. INIT-2.COM durch das MBASIC-Programm PSEUDO.BAS bereits alle angeschlossenen Drives mit der Kapazität installiert sein müssen, wie sie später zusammen mit der Saturn-Karte betrieben werden sollen. Es werden also zunächst alle Saturn-Programme auf die künftige Boot-Diskette (z.B. 2 x 80 Tracks) kopiert. Es sind dies die Programme PSEUDO.BAS, PSEUDO.COM, PARAM.DAT, INIT.COM, INIT2.COM. Anschließend wird diese Diskette im Autopatch gebootet. Dann müssen alle vorhandenen Drives einmal angesprochen werden (z.B. B:, C: etc.). Man kann sich jetzt mit STAT DSK: noch einmal überzeugen, daß alle Drives im richtigen Format „eingehängt" sind. Erst jetzt wird MBASIC aufgerufen und RUN „PSEUDEO" eingegeben. Im weiteren kann so vorgegangen werden, wie es in der Saturn-Beschreibung angegeben ist, um INIT und INIT2 anzupassen. Die so erstellten RAM-Disk-Treiber arbeiten mit der AFDC einwandfrei zusammen. Auf dem Markt befindet sich auch eine von R. Hergert an CP/M 2.23 angepaßte Version der Saturn-Software. Diese ist mit der AFDC-Firmware nicht verträglich, weil der RAM-Disk-Treiber zu lang ist und die Disk-Parametertabellen des AFDC teilweise überschreibt (s. AFDC-Handbuch).

Frei-Programme von EcoSoft

EcoSoft Economy Software AG, eine neugegründete Schwestergesellschaft von Intus Software übernahm am 1.8.86 von Intus den Vertrieb von Frei-Programmen und Shareware. In Vertretung verschiedener amerikanischer Shareware-Lieferanten und Autoren übernimmt EcoSoft auch die Registrierung von Anwendern und liefert ihnen Handbücher und Neufassungen der Programme. EcoSoft bietet derzeit über 20.000 Programme auf über 1.200 Disketten für Apple II, Macintosh, Atari ST, IBM und Kompatible, C64, C128 und Amiga. Programmübersichten können von EcoSoft kostenlos angefordert werden.
Bezugsquelle: EcoSoft, Waldshut-Tiengen

Neues von der Basis-User-Group

Das Rundschreiben Nr. 3 der Basis-User-Group vom September 1986 befaßt sich schwerpunktmäßig mit Fragen zur Wordstar-Anwendung unter Basis-Computern. Als Fortsetzung zum ersten Teil, der im Rundschreiben Nr. 1 veröffentlicht wurde, werden die Patches und Labeladressen von Wordstar 3.0 beschrieben. Der zweite Wordstar-Artikel befaßt sich mit dem Finden und Patchen der User-Patch-Area von Wordstar, Version 3.0.
Andere Themengebiete werden mit folgenden Artikeln behandelt:
– Fehlerbereinigung von CP/M 3 Version C.A1
– CP/M-3.0-Patches
– Turbo-Pascal-2.0-Patches.
Für Interessenten an den Nachrichten der Basis-User-Group nochmal die Kontaktadresse der BUG:
Rolf Gachnang
Neue Jonasstr. 81
CH- 8640 Rapperswil

Bühler-Elektronik-Filiale

Die Firma Bühler-Elektronik eröffnet am 10.11.86 in Mannheim, M7, 9a-10 einen weiteren Filialbetrieb neben ihrer Filiale in der Karlsruher Waldstraße. Auf ca. 250qm Verkaufsfläche werden in Mannheim elektronische Bauteile, Meßgeräte und Werkzeuge, Computer und Computer-Peripherie sowie HIFI-Geräte, Lautsprecher und Alarmanlagen angeboten.

Atari-ST-Produkte

Datenbank für den Atari ST

Dieter und Jürgen Geiß vom Softwarehaus ADI (Peeker-Lesern längst als fachkundige Autoren bekannt) haben in Zusammenarbeit mit ATARI die echt relationale Datenbank ADIMENS ST entwikkelt. Die Datenbank besitzt eine Schnittstelle zu den Textverarbeitungsprogrammen 1ST WORD und 1ST WORD PLUS und schließt damit die Lücke zwischen Dateiverwaltung und Textverarbeitung. Die Einbettung in die GEM-Oberfläche des Atari ST erlaubt eine unproblematische Bedienung des Systems. ADIMENS ST überzeugt durch Leistungsfähigkeit und Lieferumfang:
Die Datenbank ist echt relational aufgebaut, d.h. eine *physikalische* Datei kann mehrere *logische* Dateien umfassen; dadurch sind Verzweigungen möglich.
ADIMENS ST zeichnet sich auch bei größeren Datensätzen (z.B. 2000 Datensätze) durch extrem kurze Such-, Zugriffs- und Sortierzeiten aus. Beim Einsatz einer RAM-Disk beträgt die Suchzeit für den ersten Datensatz ca. 0.04 - 0.3 Sekunden, die Zugriffszeit für jeden weiteren Datensatz nur 0.01 - 0.1 Sekunden!
Daten- und Textverarbeitung werden mit der neuen Datenbank über Shell integriert. Ein direkter Wechsel zwischen Textverarbeitung und der aktuellen Datenbank ist daher problemlos möglich.

Die integrierte Druckeranpassung und Druckersteuerung erlaubt die Direktausgabe von Listen auf Atari-, Epson- oder IBM-kompatiblen Matrixdruckern.

Vor jeder Option kann ein Hilfetext mit erklärenden Informationen angewählt werden.

Für Programmierer, die aufbauend auf der durch ADIMENS ST erzeugten Datenbasis eigene Programme erstellen möchten, ist eine Programmierschnittstelle (C) vorgesehen.

Dank der bedienerfreundlichen GEM-Oberfläche des Atari ST läßt sich der mächtige Funktionsumfang des Software-Pakets leicht beherrschen.

Die umfassende Konzeption der Datenbank eröffnet dem Anwender vielfältige Einsatzmöglichkeiten:
– Integrierte Daten- und Textverarbeitung (Listen, Serienbriefe, Formulare)
– Karteiaufbau und -verwaltung
– Adreßverwaltung mit komplexen Selektionsmöglichkeiten
– Katalogerstellung und -verwaltung
– Bibliotheksanwendungen
– Auftrags-, Projekt-, Terminplanung
– Verwaltung, Pflege und Organisation von diversen Daten, Listen, Plänen usw.
ADIMENS ST kostet komplett ca. DM 499,–
Bezugsquelle: ATARI Corp., Raunheim und Atari-Händler

Amateurfunk-Fernschreib-Programm für Atari ST

Das Programm RADIO-WRITER ST ermöglicht den Empfang und die Aussendung von Texten über Fernschreibkanäle im Code CCITT Nr. 2 (Baudot) und im ISO-7-Bit-Code (ASCII). Damit ist ein Zugriff auf Datenbanken über Telefonmodems oder Fernschreiben im Amateurfunk möglich. Das Programm bietet frei wählbare Schnittstellenparameter, Selektivruf-Auswerter und Selbsttester, geteilte Bildschirmdarstellung, vollautomatische Steuerung des Empfängers und hat bereits im QSO änderbare Standardtexte eingebaut. Die Einstellungen der GEM-Bildschirmmenüs sind mit der Maus programmierbar, Aufzeichnung und Wiedergabe der Textdateien sowie die Druckausgabe während des Empfangs erfolgen ohne Datenverluste. RADIO-WRITER ST arbeitet mit folgender Gerätekonfiguration: Atari 260ST, 520ST/ST+, 1040ST, Monitor SM124, mindestens 1 Diskettenlaufwerk SF354/SF314 oder Harddisk, Konverter mit AFSK (bzw. Modem) mit Anschluß an die RS232-Schnittstelle des Computers, Texteditor zum Erstellen der Texte (ED,1ST WORD).
RADIO-WRITER ST kostet komplett mit Diskette und Handbuch ca. DM 98,–.
Bezugsquelle: AFUSOFT – Buchhandlung Franke, Eisingen

Buchhaltung auf dem Atari ST

Das Haus der Buchhaltung bietet für den Atari ST ein Administrationspaket an, das v.a für Benutzer aus dem Bereich des Einzelhandels interessant sein dürfte. Es umfaßt Programme zu den Bereichen Buchhaltung, Faktura, Lohnbuchhaltung, Karteiverwaltung und Scheckschreibung. Insgesamt werden 7 Disketten mitgeliefert, von denen eine spezielle Texte für die Implementierung auf dem Atari ST enthält; von den übrigen Disketten sind 3 Programmdisketten und 3 vorbereitete Datendisketten. Ein 300seitiges, deutschsprachiges Handbuch wird mitgeliefert.
Im Einzelbezug kosten die Programme ca. DM 3900,–, als Gesamtpaket ca. DM 2260,– (Preise incl. Mwst.)
Bezugsquelle: Haus der Buchhaltung, Duisburg

Verlangen Sie mehr

Computerbücher aus dem Fachbuchverlag Hüthig:
Präzise Informationen, aktuelle Themen,
gut lesbarer gesetzter Text.

Arne Schäpers

ProDOS-Analyse

Versionen 1.0.1, 1.0.2, 1.1.1

Arne Schäpers
ProDOS-Analyse
Versionen 1.0.1, 1.0.2, 1.1.1
1985, 470 S., kart., DM 68,—
ISBN 3-7785-1134-3
Dies ist die umfangreichste und detaillierteste Darstellung, die jemals ein Apple-Betriebssystem erfahren hat. So etwas gibt es nicht einmal in Amerika!

Ulrich Stiehl
Apple Assembler
1984, 200 S., 3 Abb., kart., DM 34,—
ISBN 3-7785-1047-9
Alle wichtigen ROM-Routinen sowie eine Vielzahl von Hilfsprogrammen werden in diesem Buch für den Programmierer mit Anfängerkenntnissen vorgestellt. Insgesamt über 40 Utilities mit mehreren völlig neuartigen Programmen sind gelistet.

Arne Schäpers

Bewegte Apple-Grafik

DOS Toolkit-Erweiterungen

Arne Schäpers
Bewegte Apple-Grafik
1985, 305 S., 6 Abb., kart., DM 58,—
ISBN 3-7785-1150-5
Ein Kursus für alle, die auf dem Apple hochaufgelöste und bewegte Grafiken in Maschinensprache programmieren wollen. Schrittweise wird ein Arcade-Spiel entworfen, das käuflichen Action-Spielen in der meisterhaften Grafik als Vorbild dienen kann.

Ulrich Stiehl
Apple DOS 3.3
Tips und Tricks
2., durchges. Auflage 1984, 216 S.,
kart., DM 28,—
ISBN 3-7785-1049-5
Das Standardwerk zum DOS 3.3 sowohl für BASIC- als auch für Assemblerprogrammierer. Enthüllt einige bislang noch niemals publizierte Techniken und viele Tricks aus der langjährigen Praxis des Autors. Dieses Buch ist der unentbehrliche Begleiter für jeden Apple-Programmierer.

Frank Bühler

Applesoft BASIC

Frank Bühler
Applesoft BASIC
Tips und Tricks
1985, 241 S., 40 Abb., kart., DM 38,—
ISBN 3-7785-1094-0
Das Buch enthält eine komplette Beschreibung aller möglichen Applesoft-Befehle und zeigt an einem Beispiel die erforderliche Syntax auf. Ausgearbeitete Unterroutinen können leicht in eigene Programme übernommen werden.

Ulrich Stiehl
ProDOS für Aufsteiger
Band 1
2., geänderte Auflage 1985, 208 S.,
kart., DM 28,—
ISBN 3-7785-1098-3
Das Schwergewicht des ersten Bandes liegt auf der Darstellung der Pro-DOS-internen Systemadressen und der Assemblerprogrammierung unter diesem neuen Betriebssystem. Das Werk enthält auf über 70 Seiten eigens hierfür entwickelte Programme.

Jürgen Kehrel

Apple-Assembler lernen

Band 1: Einführung in die Assembler-Programmierung

Jürgen Kehrel
Apple-Assembler lernen
Band 1: Einführung in die Assembler-Programmierung des 6502
1985, 235 S., kart., DM 38,—
Noch nie war es so einfach, die Assembler-Programmierung auf dem Apple zu erlernen, wobei auch der neue 65C02-Prozessor behandelt wird.

Carl Ulrich Wassermann

Apple IIc

Handbuch für Anwender und Programmierer

Carl-Ulrich Wassermann
Apple IIc
Handbuch für Anwender und Programmierer
1985, 324 S., kart., DM 35 —
ISBN 3-7785-1157-2
Ein Handbuch für alle, die die besonderen Eigenschaften der Apple IIc voll ausschöpfen wollen.

Benutzen Sie die Bestellkarte im PEEKER!

Weitere Titel und Informationen finden Sie in unserem Computerbuch-Katalog:
Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1

Hüthig

Warum wollen Sie ein teures Textverarbeitungsprogramm kaufen, wenn es ein billiges und besseres gibt?

Fast-Writer

von Harald Grumser
Programmdiskette und Handbuch
Gerätevoraussetzung: Apple IIe oder IIc (nicht II+)
DOS-3.3-Version.
Normalpreis DM 128,– (ISBN 3-7785-1419-9)
Sonderpreis für Peeker-Abonnenten DM 98,–

ProDOS-Version.
Normalpreis DM 128,– (ISBN 3-7785-1421-0)
Sonderpreis für Peeker-Abonnenten DM 98,–
Kombinationspreis für Bezieher der DOS-3.3-Version DM 28,–

Fast-Writer läuft auch auf neuem Apple GS!

Der Fast-Writer von Harald Grumser ist in zahlreichen Funktionen wie Scrollen, Suchen und Ersetzen mit Abstand das schnellste und damit angenehmste Textverarbeitungsprogramm für den Apple IIe oder IIc.

Flexibilität

Viele Textverarbeitungsprogramme sind geschützt und laufen deshalb nur in Verbindung mit normalen Disk-II-Laufwerken. Nicht so der Fast-Writer!

– Der Fast-Writer modifiziert weder DOS 3.3 (oder Diversi-DOS) noch ProDOS und kann deshalb mit BRUN FAST.WRITER gestartet werden. Unter Diversi-DOS ist der Fast-Writer dann in 3 Sekunden im Speicher. Vergleichen Sie einmal, wie lange es dauert, bis andere Textprogramme im Speicher sind!

– Da der DOS-3.3-Fast-Writer in den oberen 16K (= Language Card) liegt, kann man ihn vorübergehend verlassen und mit einem einfachen Befehl wieder starten. Mit anderen Worten: Der Fast-Writer ist permanent verfügbar, auch wenn Sie zwischendurch beispielsweise mit FID Dateien kopiert haben.

– Der Fast-Writer läuft mit allen externen Datenspeichern, die für DOS 3.3 oder ProDOS gedacht sind, z.B. mit dem Erphi-160-Spur-Subsystem, mit der Megaboard-MDB-Festplatte, mit RAM-Karten usw. Spezielle Anpassungen sind nicht erforderlich. Suchen Sie einmal ein Textprogramm, das mit diesen Datenspeichern auf Anhieb funktioniert!

– Der Fast-Writer kann mühelos über ein Menü für Ihre speziellen Aufgaben konfiguriert werden. Sie können z.B. per Knopfdruck die Zeilenbreite (normal 80 Zeichen) am Bildschirm einstellen, wobei ab einer Breite von weniger als 41 Zeichen automatisch auf die größere Bildschirmschrift umgestellt wird. Ferner können Sie die Größe des Arbeitsspeichers (insgesamt ca. 35 500 Zeichen) beliebig in Textspeicher und Hilfspuffer (für Löschen und Blockverschieben) aufteilen. Wenn Sie z.B. große Textblöcke im Speicher zu verschieben haben, so können Sie einen entsprechend großen Hilfspuffer von z.B. 10 000 Zeichen einrichten. Damit entfällt das zeitraubende Zwischenspeichern und Einlesen von Diskette.

Befehlsvorrat

Der Fast-Writer verfügt über eine große Zahl von Befehlen, von denen Sie in der Praxis jedoch nur wenige benötigen. Fünf Befehlsübersichten sind durch eingebaute Hilfsübersichten immer abrufbar, so daß Sie schon nach einer mehrstündigen Einarbeitung auf das Handbuch verzichten können. Eine Auswahl der wichtigsten Befehle:

– Freie Cursorbewegung in allen vier Richtungen mit eingebauter Schnell-Scroll-Routine.

– Diverse, per Knopfdruck ein- und ausschaltbare Optionen, z.B. Wortumbruch/kein Wortumbruch, Return sichtbar/unsichtbar, Kopfzeile (Statuszeile) mit Speicherbelegung, Cursorposition usw. eingeblendet/ausgeblendet, Bildschirm geteilt/ungeteilt, Tabulatorleiste sichtbar/unsichtbar, überschreibmodus (statt normalen Einfügmodus) ein/aus usw.

– Eingabe von Kontrollbuchstaben (einschließlich Ctrl-V!) möglich. Automatische Konvertierung in Groß- oder Kleinschreibung (unter Berücksichtigung der Umlaute und ß!)

– Extrem schnelles Suchen und Ersetzen von Zeichenketten (vorwärts und rückwärts).

– Makros frei definierbar und per Knopfdruck abrufbar. Makros können nicht nur stereotype Wortfolgen sein (z.B. „Sehr geehrte Herren"), sondern auch alle Befehlsfolgen, die man beim Fast-Writer sonst über die Tastatur eingeben würde. So läßt sich beispielsweise ein Text automatisch von Laufwerk 1 laden und auf Laufwerk 2 speichern.

– DOS-Kommandos wie Catalog, Delete, Rename usw. immer verfügbar (bei DOS 3.3 zusätzlich Init, bei ProDOS zusätzlich Online und Datum)

– Ausdruck auf Matrixdrucker (normal endlos), Schreibmaschine (normal mit Einzelblatt) und zu Kontrollzwekken auf Bildschirm; links- und rechtsbündig, zentriert und Blocksatz; einstellbarer linker, rechter und oberer Rand (im Text änderbar), bei Bedarf mit Kopfzeile und Paginierung usw. Der Ausdruck kann über eigene Druckertreiber umgelenkt werden, um z.B. Probleme mit Typenrädern, Steuerzeichen usw. zu beheben.

– Makros, Druckparameter, Druckertreiber und Tabulatoren können auf Diskette gespeichert werden.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg 1