Nr. 5/86 Mai
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

peeker

Fußball-Weltmeisterschaft 1986 in Mexiko

Luxus-Diskketteneditor

Bildschirmausgabe in Apple-Pascal

Kreiszahl Pi auf 15 000 Stellen

MS-File und MS-Word

Backup-Programm für Megaboard-Festplatte

Rückblick auf 10 Jahre Apple

Hüthig
PUBLIKATION

Wichtige Information für unsere peeker-Leser

Sie erhalten Ihren »**peeker**«

- an allen **Bahnhofsbuchhandlungen und -kiosken**
- bei allen **Apple-Händlern**
- und natürlich **beim Verlag.**

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

Hüthig PUBLIKATION

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

EDITORIAL

Sammeldisketten

Auf die nächsten Sammeldisketten werden wir neben den üblichen Programmen zu den Peeker-Aufsätzen zusätzlich eine Reihe von professionellen Großprogrammen aufnehmen. Nach dem Applesoft-Editor „Macroeditor" (Disk #16) geht es weiter mit einem kompletten ProDOS-Disketteneditor „Edit" von Arne Schäpers (Disk #17). Dann folgen das lang erwartete UCSD-Grafik-Druckprogramm „Superdump" für Epson- und Imagewriter von Jürgen Geiß, dann ein Programmpaket für Buchregister und Glossare, dann ein kompletter UCSD-Grafik-Editor „Pic-Edit" von Jürgen Geiß (Gegenstück zu „Turtle-Graphics-Libarary-Paket), dann ein kaufmännisches Kalkulationprogramm von mir, dann ein P-Code-Disassembler von Jürgen Geiß, und und und.
Jede der nachfolgenden Sammeldisketten wird deshalb zum Bersten voll sein, wobei es sich nicht wird vermeiden lassen, daß die Programmpaktete ggf. auf zwei Disketten verteilt werden. So enthält beispielsweise die Disk #17 den kompletten Quellcode zu dem Disketteneditor „Edit", während sich der Objektcode aus Platzgründen erst auf der Disk #18 befinden wird.

Kompaktkurse

Im letzten Jahr erschienen die auf gelbem Papier gedruckten, meist 16seitigen Kompaktkurse „6502 leicht gemacht" (Heft 7/85), „Z80-Assembler-Programmierung unter CP/M" von Dr. H. Kersten (11/85) und „Pascal, Teil 1" (Grundlagen von UCSD- und Turbo-Pascal) von mir (Heft 12/85). In den nächsten Heften werden wir diese „Minibuch"-Reihe fortsetzen mit den bereits vorliegenden Kompaktkursen „Einblicke in Logo" von Ernst F. Haas, „Pascal, Teil 2" (UCSD: Dateiverwaltung, Bibliothekskonzept u.a.) von Jürgen Geiß, „68000-Assembler auf dem Mac" von Pit Capitain. Weitere Kompaktkurse (z.B. „65816") sind geplant.

Kyan-Pascal

Die Version 2.0 wird am 25.4.86 ausgeliefert, doch haben wir aus produktionstechnischen Gründen zunächst nur 600 Exemplare von Kyan-Software erhalten, so daß ca. 40 Besteller leider etwa 14 Tage später beliefert werden. Kunden mit einer durch 50 teilbaren Kyan-Club-Nummer bekamen zusätzlich eine Buchprämie. Es sind dies die Club-Mitglieder:
1050: Rüdiger Okel, Lübeck
1100: Stefan Friske, Ganderkesee
1150: Thomas Quick, Mannheim
1200: Hans-Detlef Meier, Hannover
1250: Peter Holzwarth, St. Augustin
1300: Dieter Charchot, Hamburg
1350: Rudolf Willenbücher, Sinzheim
1400: Georg Foltin, Erkrath
1450: Karl-Heinz Ruppert, Würzburg
1500: Dr. Edi Stricker, CH-Olten
1550: Hans H. Meier, Germering
1600: Reinhard Müller, Mettingen

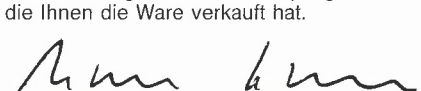
Die erste Club-Mitglieder-Liste wird Anfang Mai verschickt. Technische Einzelheiten zur Version 2.0 sowie zu den drei Kyan-Toolkits, die ebenfalls zu Club-Sonderpreisen erhältlich sind, folgen im nächsten Peeker.

Megaboard-Sonderangebot

In diesem Heft bieten wir zum ersten Mal eine Hardware im Rahmen des Hüthig Software Service an, und zwar eine Festplatte namens „Mobile Datenbox" mit Megaboard-Controller und fertig angepaßtem Dateiverwaltungsprogramm. Diese Komplettlösung ist sicherlich nicht nur für kaufmännische Apple-Anwender interessant.
Möglicherweise werden wir in Zukunft auch Hardware-Produkte amerikanischer Provenienz anbieten, denn zu unserer Enttäuschung müssen wir immer wieder feststellen, daß bestimmte Produkte wie etwa das nützliche „Fingerprint Plus" (siehe Heft 4/85) von keinem einzigen deutschen Apple-Händler vertrieben wird.

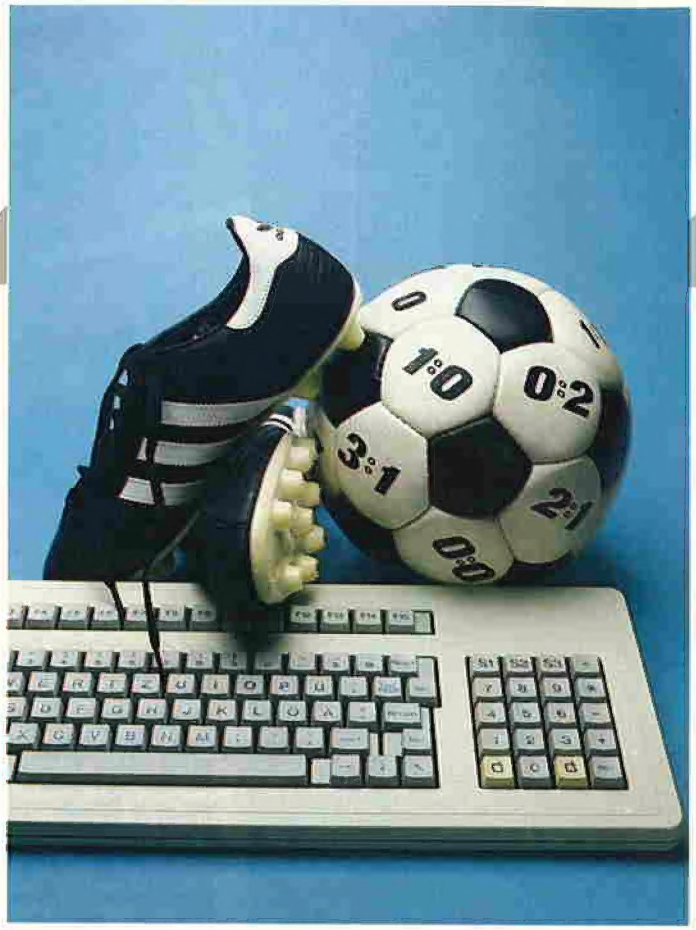
Leserbriefe

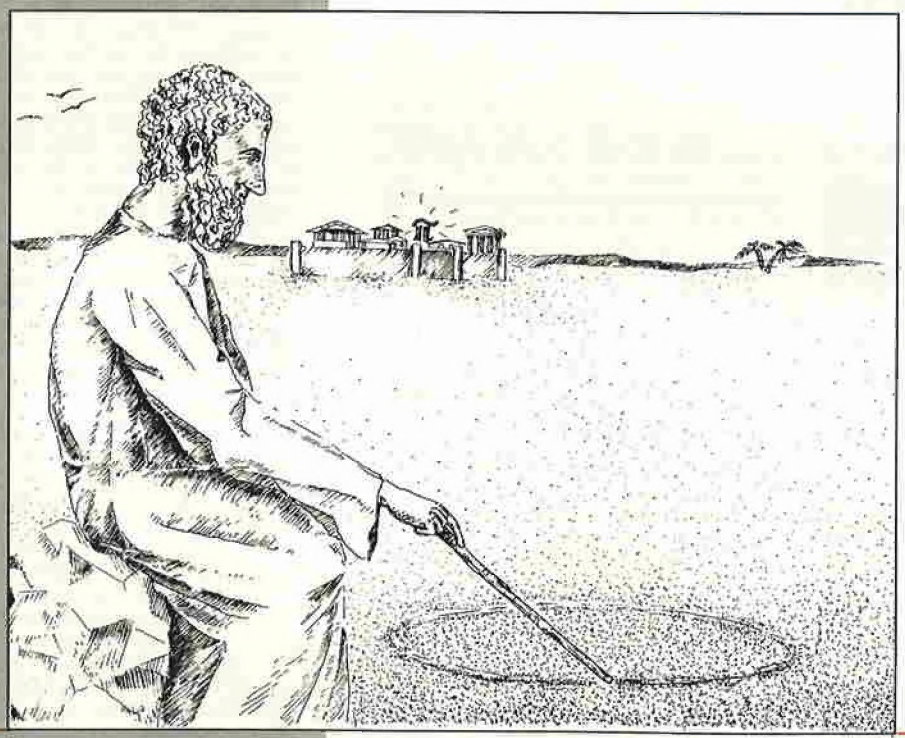
Nach dem Ausscheiden von Herrn Grumser muß ich vorübergehend die Peeker-Redaktion ganz allein betreuen. Deshalb hat sich das Erscheinen dieses Mai-Heftes verzögert, und auch das Juni-Heft wird verspätet erscheinen. Aus diesem Grunde bin ich gezwungen, die Leserbrief-Politik zu ändern. Sie werden es kaum glauben, aber seit Erscheinen des Peekers im September 1984 ist die Leserkorrespondenz auf 26 Leitz-Ordner angeschwollen. Ab sofort werde ich deshalb nur noch diejenigen Zuschriften beantworten, die sich konkret auf Peeker-Aufsätze oder -Software beziehen. Alle anderen Anfragen, die Sie eigentlich an den Produzenten oder Ihren Händler hätten richten müssen, werden entweder gar nicht mehr beantwortet oder in veröffentlichte Leserbriefe umgemünzt, wenn sie von allgemeiner Bedeutung sind. Bitte beachten Sie die einfache Faustregel: Zuständig für technische Anfragen ist immer diejenige Firma, die Ihnen die Ware verkauft hat.



Ulrich Stiehl

INHALT





Impressum

Peeker
3. Jahrgang 1986
ISSN 0176–9200

Verleger und Herausgeber:
Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften:
Heinz Melcher
Chefredakteur: Ulrich Stiehl (us)

Hüthig
PUBLIKATION

Telefonnummern:
Zentrale: 06221/489-1
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.

peeker

Inhalt 5/1986

Anschrift:
Dr. Alfred Hüthig Verlag GmbH
Im Weiher 10, Postfach 10 28 69
6900 Heidelberg
Telefon (0 62 21) 4 89-1
Telex 4-6 17 27 hued d
Telefax (0 62 21) 489 279
BTX * 51851 #

„Peeker" ist eine unabhängige Zeitschrift. Sie ist nicht verbunden mit der Firma Apple Computer, Inc. oder der Apple Computer GmbH. APPLE, das Apple-Zeichen und MAC sind Warenzeichen der Firma Apple Computer, Inc. und MACINTOSH ist ein Warenzeichen, in Lizenz vergeben von der Firma McIntosh Laboratory an die Firma Apple Computer, Inc.

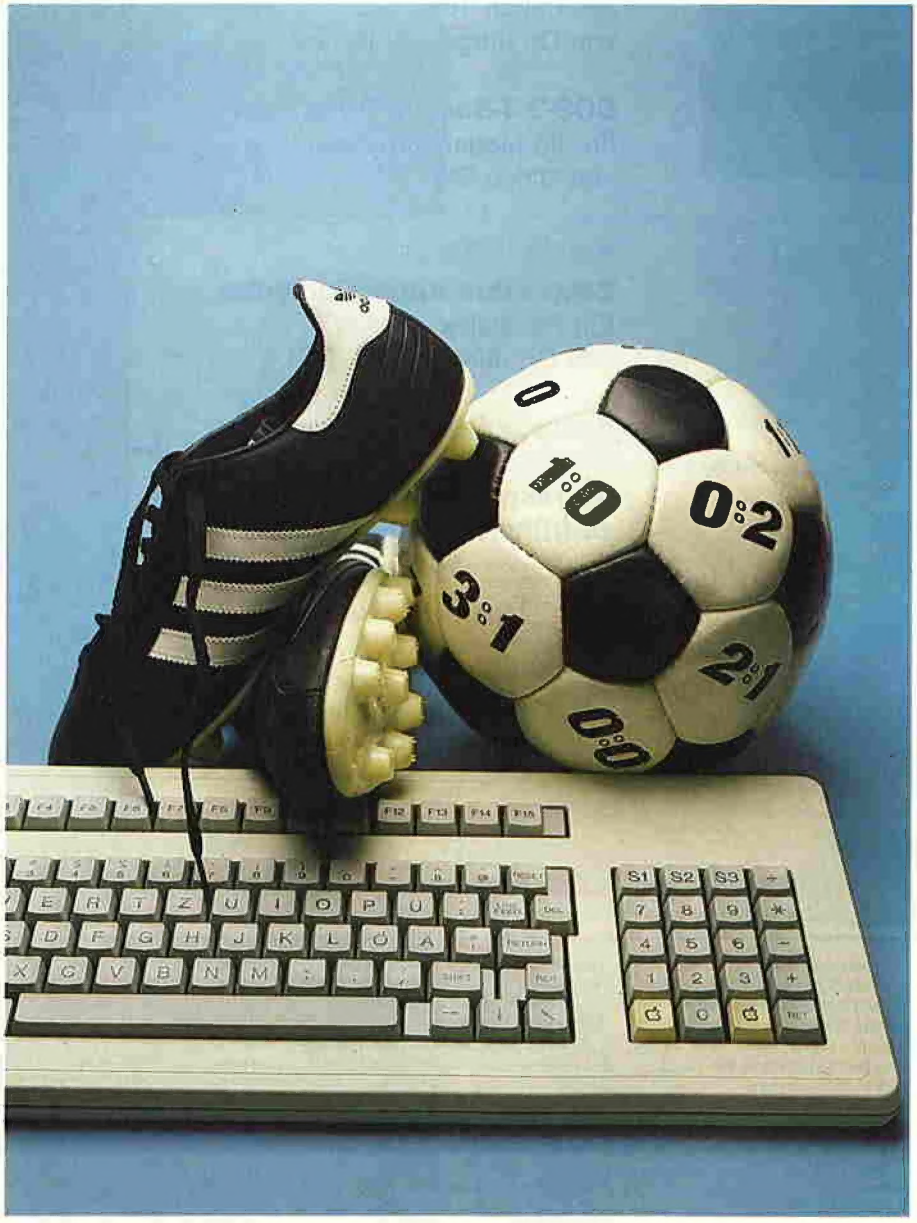
Vertrieb:
Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement Inland DM 72,–, einschl. MwSt und Versandkosten.
Jahresabonnement Ausland DM 72,– plus DM 18,– Versandkosten.
Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089/319 10 67, Telex 0 522 656
Vertriebsleitung:
Walter Menzel, Tel. (0 62 21) 48 92 80

Bankverbindungen:
Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postgirokonten: BRD: Ludwigshafen 4799-673, BLZ 545 100 67; Österreich: Wien 7555 88; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 5 968 92 08; Belgien: Brüssel 1084 1261; Dänemark: Kopenhagen 603 4969; Norwegen: Oslo 199 4243; Schweden: Stockholm 5477 76-5
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 02 65 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

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

Fußball-weltmeisterschaft 1986 in Mexiko

Ein Simulationsprogramm in Turbo-Pascal 3.0



von Manfred Stertenbrink

Das Programm simuliert die Fußballweltmeisterschaft 1986 in Mexiko. Dazu werden zunächst die sechs Vorrundengruppen und dann das Achtel-, Viertel-, Halb- und Finale durchlaufen. Bekanntlich spielt in den Vorrundengruppen jeder gegen jeden, wobei die beiden Gruppenersten auf jeden Fall, der Gruppendritte eventuell in die Endrunde (Achtelfinale) kommen. Im weiteren Verlauf wird im K.o.-System gespielt.

1. Toto-Verfahren

Zur Simulation stehen vier Alternativen zur Auswahl, die prinzipiell alle nach einem Verfahren arbeiten, welches bei der Auslosung der Totozahl für ausgefallene Spiele verwandt wird. Bei dieser Auslosung wird dem Zufall dadurch möglichst wirklichkeitsnah vorgebeugt, daß die Wahrscheinlichkeit für die 0, 1 oder 2 nicht etwa gleichgroß ist, sondern den Spielstärken der Mannschaften angepaßt wird. Deshalb setzen einige Experten eine „amtliche Tendenz" für jedes Spiel fest, die sich aus drei natürlichen Zahlen zusammensetzt, deren Summe zehn ergibt. So bedeutet beispielsweise 6-1-3, daß die Wahrscheinlichkeit für einen Sieg der 1. Mannschaft 0,6, die Wahrscheinlichkeit für einen Sieg der 2. Mannschaft 0,3 und die Wahrscheinlichkeit für ein Unentschieden 0,1 beträgt.

Man kann sich das Verfahren so vorstellen, daß in eine Urne sechs Kugeln mit der Ziffer 1, eine Kugel mit der Ziffer 0 und drei Kugeln mit der Ziffer 2 gelegt werden.

Dann wird eine Kugel gezogen, und die Ziffer auf dieser Kugel gibt die gesuchte Totozahl an. Grundsätzlich muß jede der drei Ziffern 0, 1, 2 in der Urne vorkommen (so ist beispielsweise 8-2-0 nicht erlaubt), da auch Siege von Fußballzwergen möglich sind und bei den letzten Weltmeisterschaften vorkamen; die bundesdeutsche Mannschaft kann davon berichten.

Im einzelnen verfahren die vier Alternativen wie folgt:

a) Totaler Zufall
Alle Spiele werden auf 3-4-3 gesetzt.

b) Zufall nach von einer Expertengruppe vorgegebenen Spielstärken
Es wurden 20 „Experten" um eine Einstufung der Mannschaften (verschiedene Wertigkeiten) gebeten. Im Programm wird diese Wertigkeit jeweils ins Totosystem übertragen. Zwei Beispiele:
Bulgarien (Wertigkeit 4) - Italien (Wertigkeit 6) wird 3-3-4 gesetzt,
Mexiko (Wertigkeit 7) - Irak (Wertigkeit 1) wird zu 7-2-1.

c) Zufall nach von Ihnen vorgebenen Spielstärken
Analog zu (b), nur wird die Einstufung der Mannschaften durch den Benutzer vorgenommen.

d) Zufall nach von Ihnen für jedes Spiel vorgenommener Toto-Vorhersage
Hier wird bei jedem einzelnen der 52 Spiele die Eingabe der Toto-Tendenz erwartet.

Nach diesen Vorgaben wird per Zufall ein Sieger bzw. ein Unentschieden ermittelt und zusätzlich ein geeignetes Torergebnis erzeugt. Bei Unentschieden in der Endrunde erfolgt eine neue Simulation, so daß auch Ergebnisse „nach Verlängerung" auftauchen. In den Vorrundengruppen wird jeweils eine Abschlußtabelle erstellt; für das Achtelfinale, für welches sich neben den jeweiligen beiden ersten Teams der sechs Gruppen noch die vier besten Gruppendritten qualifizieren, findet eine entsprechende Auswahl statt.

Zusätzlich zu diesen Alternativen bietet das Programm noch die Möglichkeit, vollkommen unabhängig vom Zufall die entsprechenden Spielergebnisse einzugeben, wobei dann ebenfalls das komplette Turnier mit Tabellen, Qualifikation für die nächste Runde usw. durchlaufen wird.

2. Beispiel

Zur Veranschaulichung sind zwei Auszüge tabellarisch dargestellt; diese sind unter der Wahl (b) entstanden und zeigen die Ergebnisse aus Gruppe 5 und dem Achtelfinale.

Achtelfinale

Belgien : Ungarn	2:0
Frankreich : Paraguay	2:1 n. Verl.
Bulgarien : Dänemark	1:0
Brasilien : Portugal	2:0
Italien : UdSSR	3:2 n. Verl.
Polen : Deutschland	4:3 n. Verl.
England : Mexiko	4:3 n. Verl.
Uruguay : Spanien	0:1

Gruppe 5

Uruguay : Deutschland	2:0
Schottland : Dänemark	1:0
Deutschland : Schottland	2:0
Dänemark : Uruguay	1:0
Dänemark : Deutschland	2:2
Schottland : Uruguay	0:2

Tabelle:

1. Uruguay	4:1	4:2
2. Deutschland	4:4	3:3
3. Dänemark	3:3	3:3
4. Schottland	1:4	2:4

Unter Wahl (b) sind die Mannschaften folgendermaßen gesetzt:

```
 1. Brasilien      9
 2. Argentinien    8
    Frankreich     8
    Daenemark      8
 5. Mexiko         7
    Deutschland    7
    Uruguay        7
    England        7
 9. Italien        6
    Belgien        6
    UdSSR          6
    Spanien        6
    Schottland     6
    Polen          6
    Portugal       6
16. Paraguay       5
    Ungarn         5
18. Bulgarien      4
    Nordirland     4
20. Algerien       3
21. Suedkorea      2
    Kanada         2
    Marokko        2
24. Irak           1
```

Eine 3000-fache Simulation ergab folgendes Ergebnis:
(Anzahl der gewonnenen Weltmeisterschaften)

```
 1. Brasilien     263
 2. Frankreich    254
 3. Argentinien   238
 4. Daenemark     208
 5. Mexiko        196
 6. England       176
 7. Uruguay       165
 8. Polen         162
 9. Belgien       156
10. Deutschland   153
11. UdSSR         143
12. Portugal      136
13. Italien       131
14. Paraguay      120
15. Schottland    110
16. Spanien       101
17. Ungarn         94
18. Nordirland     63
19. Bulgarien      47
20. Algerien       25
21. Irak           21
22. Marokko        19
23. Kanada         14
24. Suedkorea       5
```

3. Implementierung

Das Programm, das in Turbo-Pascal 3.0 geschrieben wurde und eine CP/M-Version mit mindestens 60K, also CP/M 2.23 oder 2.26, voraussetzt, besteht aus den drei sehr umfangreichen Modulen

WM86.PAS (Hauptprogramm)
VORSPANN.PAS (Include-Datei)
ENTSCHEI.PAS (Include-Datei)

Aus Platzgründen können diese Programme nicht im Peeker abgedruckt werden und befinden sich deshalb nur auf der Sammeldiskette.

Verfahren Sie Schritt für Schritt wie folgt:

1 Sammeldisk #17 mit den DOS-3.3-Textfiles WM86.PAS, VORSPANN.PAS und ENTSCHEI.PAS in Laufwerk 2 (= B) einlegen.

2 CP/M-Diskette (60K- oder 64K-Version), die Turbo-Pascal 3.0 und APDOS enthalten sollte, von Laufwerk 1 (= A) booten.

3 APDOS starten und nach dem Sternchen jeweils eingeben:

```
*WM86.PAS=WM86.PAS
*VORSPAN.PAS=VORSPANN.PAS
*ENTSCHE.PAS=ENTSCHEI.PAS
```

Beachten Sie, daß wegen eines Bugs in APDOS zunächst die letzten beiden Zielnamen um einen Buchstaben gekürzt werden müssen.

4 APDOS über Ctrl-C verlassen und nun die letzten beiden Namen wieder richtig benennen, also

```
>REN VORSPANN.PAS=VORSPAN.PAS
>REN ENTSCHEI.PAS=ENTSCHE.PAS
```

5 Jetzt Turbo-Pascal 3.0 starten und bei „Include Error Messages" je nach CP/M-Version ggf. mit N für Nein antworten.

6 O für Optionen wählen, C für COM-File tippen und wieder zurück über Q für Quit.

7 C für Compilieren tippen und dann WM86 für „Workfile name" eingeben. Der Rest der Compilierung erfolgt automatisch. Beachten Sie jedoch, daß noch genügend Platz auf der Diskette ist, da WM86.COM allein 23K einnimmt.

8 Nun Turbo-Pascal verlassen und Programm mit WM86 von CP/M aus starten. Alles Weitere können Sie dann dem Menü entnehmen.

EDIT

Ein Luxus-Disk-Editor für ProDOS

von Arne Schäpers

Teil 1: Befehlsauswertung und Diskettentreiber

Hinweise

1. Wegen des Umfangs mußte der Beitrag in 3 Teile aufgespalten werden. Die Peeker-Sammeldisk #17 enthält aber schon jetzt den *kompletten* Quellcode des Gesamtprogramms, während der komplette Objektcode auf die Sammeldisk #18 aufgenommen werden wird.
2. Da es sich um ein ProDOS-Programm handelt, bot es sich an, den ProDOS-„EDASM"-Assembler der Firma Apple zu verwenden. Deshalb ist der Quellcode auf der Sammeldisk ausnahmsweise *nicht* im Big-Mac- bzw. Merlin-Format.
3. Durch das gesonderte DISKIO-Modul läßt sich der für ProDOS gedachte Disk-Editor auch für DOS 3.3, CP/M und Pascal erweitern. Aus Platzgründen ist jedoch vorläufig nur die ProDOS-Version vorgesehen. Im Falle einer entsprechenden Leserresonanz werden noch die anderen DISKIO-Module entwickelt. Schreiben Sie an den Peeker!

1. Geschichte des Programms

Angefangen hat es mit der Feststellung, daß es noch keinen Disk-Editor für ProDOS gibt und daß man einen schreiben müßte, ganz im speziellen für Konfigurationen, die nicht ausschließlich mit der Disk II arbeiten – die meisten Disk-Utilities für DOS 3.3 haben ihre eigene RWTS (= Read-Write-Track-Sector-Routine) und sind deshalb in diesem Fall nicht einmal eingeschränkt brauchbar. Bei diesen Überlegungen blieb es vorerst.

Als mir dann tatsächlich eine ProDOS-Diskette durch ein wildgewordenes Programm „auf Nimmerwiedersehen" gesagt hatte, war erfreulicherweise „T.L.D.U." zur Hand. Dieses Programm ist im Peeker, Heft 6/85 bereits besprochen worden – es funktioniert für alle vier Betriebssysteme, benutzt die (u.U. gepatchte) RWTS bzw. RWTB des Betriebssystems und ist darüber hinaus noch unwiderstehlich billig. (Es soll allerdings nicht mehr mit den neuen IIe-ROMs uneingeschränkt funktionieren. us)
Warum dann noch ein eigenes Programm schreiben? T.L.D.U. hat dasselbe Problem wie fast alle Programme, die sehr viel können: Es ist mir zu kompliziert.
Eine der häufigsten Kritikpunkte an einem anderen „Cadillac"-Programm, nämlich

Leser werben Leser

»peeker« bietet Ihnen etwas Neues!
Wer jetzt schenkt hat mehr von seinem Apple.

Dafür schenkt »peeker« Ihnen etwas: Die praktische und formschöne Box, die Ihre Disketten übersichtlich ordnet.
Sie wissen ja, wie gut »peeker« Ihnen im täglichen Umgang mit Ihrem Apple behilflich ist. Und Sie brauchen Ihren »peeker« nicht mehr zu teilen oder auszuleihen.
Die Diskettenbox ist unser Geschenk an Sie für einen neuen »peeker«-Abonnenten. Denn wer einen Apple hat, der soll auch seinen »peeker« haben.

peeker 5+6/86

Bestellcoupon

Ich habe den neuen Abonnenten geworben und erhalte kostenlos die Diskettenbox.

Name, Vorname

Straße, Postfach

PLZ, Ort

X
Datum, Unterschrift

Ich bin der neue Abonnent. Bitte liefern Sie mir bis auf Widerruf, zumindest aber für 1 Jahr, »peeker« zum Jahresbezugspreis von z. Zt. DM 72,– (Ausland plus DM 18,– Porto) an folgende Anschrift:

Name, Vorname

Straße, Postfach

PLZ, Ort

X
Datum, 1. Unterschrift

Gewünschte Zahlungsweise
☐ gegen Rechnung
☐ bargeldlos durch Bankeinzug

Konto-Nr. | Bankleitzahl

Geldinstitut

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

X
2. Unterschrift

Coupon ausschneiden und einsenden an:

»peeker«
Abonnementservice
Im Weiher 10
6900 Heidelberg 1

Hüthig
PUBLIKATION

dem Wordstar, ist: „Wenn man damit alle zwei Wochen einen einzigen Brief schreiben soll, geht es mit der Schreibmaschine schneller." Sollten Sie einen Sektor-Editor wesentlich öfter als alle zwei Wochen brauchen, wäre es vielleicht angebrachter, einmal die Laufwerke zum Reparieren zu bringen...

Damit sind die Eckbedingungen für ein derartiges Programm festgelegt, ein typischer Programmablauf (z.B. „unDELETE" eines DOS-3.3-Files) sollte im Idealfall so aussehen:
– Disk-Editor starten;
– Zieldiskette einlegen;
– „R 11,F" eingeben: liest den ersten CATALOG-Sektor;
– „D" eingeben: Hex-ASCII DUMP;
– „Sxx: yy": setzt die entsprechende Tracknummer des T/S-Sektors;
– „Szz: A0": setzt den Filenamen wieder richtig;
– „W": schreibt den Sektor
– „CATALOG": da sollte der File jetzt wieder zu sehen sein;
– „Q": Ende des Sektor-Editors – ohne Reboot des Betriebssystems.

Programme, die so etwas können, sind schon des öfteren in diversen Computerzeitschriften veröffentlicht worden. Sie melden sich typischerweise mit
1) Lesen 2) Schreiben 3) Ende

In der Überschrift zu diesem Artikel steht ja bereits das Wort „Luxus" – betrachten Sie bitte einmal folgende Kommandozeilen:

```
§/TESTVOL/TEST!H!%A=0!%B=100!D
Y0!M!%C(%A!%D(%B!%D?0!J+2
!Y1!%C+100!%D-1!J-1
!Y2!%C?0!J+3!R%C!F'ZIEL!J-3!PD
!Y3!%A+1!%B+1!%B?200!J-0
```

Hier wird auf dem ProDOS-Volume TESTVOL der File TEST herausgesucht und danach jeder Block des Files nach der Zeichenfolge ZIEL abgesucht. Wird diese Zeichenfolge gefunden, folgt ein Hex/ASCII-Dump auf den Drucker. Die Suche endet mit dem letzten Block des Files.
Oder, etwas verständlicher ausgedrückt: *Das* kann das Programm auch, nur muß es der Benutzer nicht können, um „auf die schnelle" irgend etwas auf einer Diskette zu verändern.

Gehen wir gleich ans „Eingemachte". Das Programm leistet folgendes:
– Bearbeitung von ProDOS-, DOS-3.3-, Pascal- und CP/M-Disketten sowie entsprechender RAM-Disks;

– das Betriebssystem bleibt 100%ig intakt, d.h. vorher ausgeführte Patches für RAM-Karten, 160-Track-Drives usw. bleiben erhalten;
– das Programm selber ist lauffähig unter ProDOS und DOS 3.3;
– die Bildschirmanzeige unterstützt 40 Z/Z (nur Großbuchstaben), 40 Z/Z (Groß- und Kleinschreibung), videx-kompatible 80-Zeichenkarten sowie die 80-Zeichenkarte des IIc/e. Die Anzeigeart wird vom Programm automatisch bestimmt;
– unter ProDOS kein „SYSTEM"-Programm, d.h. nach Programmende muß nicht erneut das BASIC.SYSTEM geladen werden.

Wenn Sie jetzt ein 20K-Monstrum erwarten: Die Programmlänge beträgt je nach Ausbaustufe zwischen 4,5 und 6,8K. Durch entsprechenden Programmierstil kann man aus dem gesamten Programm mit Sicherheit noch einige hundert Bytes „herausholen" – aus Gründen der „Nachbausicherheit" habe ich allerdings davon Abstand genommen.

Das Programm besteht aus insgesamt fünf Source-Files:
ED.FRAME.TXT – Programmstart, Kommando-Einlese, -Verteilung und -Auswertung sowie alle Bildschirm-Funktionen (PRINT und Cursorsteuerung);
ED.DISKIO.TXT – betriebssystem-spezifischer (RAM-)Disk-Treiber;
ED.FUNCS1.TXT – die einzelnen Befehls-Routinen, erster Teil;
ED.FUNCS2.TXT – Befehls-Routinen, zweiter Teil;
ED.CMDS.TXT – Kommando-Tabelle und Puffer-Definitionen.

Da sich das gesamte Programm doch auf rund 2500 Zeilen summiert, werden in diesem Teil 1 nur ED.FRAME und ED.DISKIO zusammen mit ein paar Kommandos gelistet und besprochen. ED.FUNCS1, ED-.FUNCS2 sowie DISKIO folgen in den nächsten Ausgaben.

2. Kommando-Übersicht

Editier-Kommandos in der Kommandozeile

Linkspfeil: Cursor nach links, beendet INSERT und DELETE;
Rechtspfeil: Cursor nach rechts, beendet INSERT und DELETE;
<RETURN>: Abschluß der Kommandozeile, nur links vom Cursor stehende Zeichen werden berücksichtigt;
Ctrl-X: Löschen der gesamten Kommandozeile;
Ctrl-I: INSERT ab Cursorposition, rechts vom Cursor stehende Zeichen verschieben sich entsprechend;
Ctrl-D: DELETE ein Zeichen unter dem Cursor, rechts vom Cursor stehende Zeichen verschieben sich entsprechend;
<ESC>: das nächste getippte Zeichen wird auf Kleinschreibung übersetzt, sofern möglich.
Ctrl-E: (Re)EDIT, läßt die „alte" Kommandozeile auf dem Bildschirm, wenn als erstes Zeichen eingegeben.
Ctrl-C: bricht die Programmausführung innerhalb einer Kommandozeile ab.

Die Befehle in ED.DISKIO

a) Prefix- und File-Kommandos

§/Volname lädt den ersten Block des Volume-Directory in den Arbeitspuffer (= BUFFER);
§/Volname/Filename lädt den ersten Block eines Files (oder eines Subdirectory) ein;
RPx,y setzt Slot x, Drive y als READ-Drive;
RP/Volname sucht das entsprechende Volume und setzt damit den READ-Drive.

b) Block-READ und -WRITE

Rxxx liest Block xxx in den Arbeitspuffer;
R+ liest den nächsthöheren,
R- den nächstniedrigeren Block;
R liest den zuletzt gelesenen Block noch einmal.

WPx,y,
WP/Volname,
Wxxx,
W+,
W- und
W
sind entsprechende Kommandos für WRITE.

Zusätzliche Kommandos

X: Aufruf des Monitors, Return zu EDIT mit Ctrl-Y.
Q: Quit. Ende von EDIT via $03D3, d.h. BASIC-Warmstart.

Einige Anmerkungen

Ctrl-E: Da speziell bei längeren Kommandozeilen das Neutippen reichlich mühselig sein kann, gibt es dieses spezielle Control-Kommando, das nach Ausführung der Kommandozeile oder Abbruch durch Fehler als erstes getippt werden muß. Damit bleibt die fehlerhafte „alte" Kommandozeile stehen und kann nötigenfalls korrigiert werden.
Ctrl-C: Vor der Ausführung jedes neuen Befehls wird das Keyboard überprüft. Wenn Ctrl-C gedrückt worden ist, wird die Kommandozeile als beendet angesehen. Dieser Befehl ist nur im Zusammenhang mit Schleifen interessant.

§/Volname/Filename lädt immer den ersten Block eines Files in den Arbeitspuffer. Dieser erste Block ist nur bei Files mit Storage Type 1 und bei (Sub)directories mit dem ersten Datenblock identisch, ansonsten handelt es sich dabei um den Indexblock bzw. um den Masterblock!

RP setzt das WRITE-Prefix immer mit. Wenn von einem Drive gelesen und auf ein anderes geschrieben werden soll, muß zuerst mit „RP" der READ-Drive, danach mit „WP" („WPx,y" oder „WP/Volname") der WRITE-Drive gesetzt werden.

R setzt die Blocknummer für WRITE immer mit. Wenn von einem Block gelesen und auf einen anderen geschrieben werden soll, muß zuerst mit
R <xx,+,-,>
der READ-Block gelesen und danach mit
W <xx,+,-,>
der WRITE-Block geschrieben werden.

X: Der Sprung in den Monitor hängt zuerst das jeweilige DOS wieder an. Hauptanwendung für dieses Kommando sind CATALOG usw. sowie LIST des Arbeitspuffers. Der Arbeitspuffer geht von $2000 bis $21FF bzw. bis $20FF (für DOS 3.3 und CP/M).

3. ED.FRAME

3.1. Allgemeine Beschreibung

ED.FRAME ist der „Rumpf" des gesamten Programms und zerfällt in folgende Teile:
a) Initialisierung des Programms;
b) Installation (40/80 Z/Z, Groß-Kleinschreibung);
c) „Einsammeln" einer Kommandozeile;
d) Abarbeitung der Kommandozeile (Command Interpreter);
e) Bildschirm-Routinen (Druck eines Zeichens, HOME usw.);
f) Parameter-Auswertung.

ED.FRAME (und damit das gesamte Programm) hat zwei Einsprünge:
$803 (dezimal 2051): löscht den Arbeitspuffer und alle Flags;
$806 (dezimal 2054): läßt Puffer und Flags intakt.

Dieser Programmteil ist (genauso wie ED.DISKIO) vollständig und wird im Verlauf der folgenden Erweiterungen nicht mehr verändert.

3.2. Problemstellungen

Das „Einsammeln" der Kommandozeile und das Ausdrucken eines Zeichens sowie die Cursorsteuerung müssen sowohl für 40 Z/Z als auch für Videx- und IIc/e-80- Zeichenkarten funktionieren. Da der Monitor außer <CR> (Carriage Return; Returntaste) und <BS> (Backstep; Linkspfeiltaste) keine Steuerzeichen zur Cursorpositionierung kennt, muß bei den Löschroutinen Clear to End of Page (Löschung bis Seitenende) und Clear to End of Line (Löschung bis Zeilenende) zwischen 40 Z/Z, d.h. Monitoraufruf, und 80 Z/Z, d.h. Steuerzeichen unterschieden werden.
Das Neudrucken einer Kommandozeile nach INSERT oder DELETE eines Zeichens gestaltet sich deshalb geradezu abenteuerlich:
Zuerst wird der entsprechend verlängerte (verkürzte) Teil der Kommandozeile rechts vom Cursor neu gedruckt, danach folgt eine Löschung bis Seitenende. Um jetzt den Cursor wieder auf das Zeichen zu positionieren, das neu getippt wurde (bzw. auf das Zeichen rechts davon bei einem DELETE), wird zuerst die (bekannte) Position des ersten Zeichens in der Kommandozeile gesetzt, danach wird der String-Teil links von der ursprünglichen Cursorposition neu gedruckt. Als Ergebnis steht der Cursor dann tatsächlich da, wo er hingehört.

3.3. Ausgewählte Routinen

(Die Routinen werden in der Reihenfolge der Speicheradressen beschrieben. Siehe Z. = Zeilennummern des Listings)

INIT (Z. 36-52) löscht den Puffer und sämtliche Flags (Betriebsarten, 80-Z/Z-Erkennung usw).
INSTALL (Z. 55-148): zuerst wird das DOS „abgehängt", diese Routine (DISCONN) ist betriebssystem-spezifisch und deshalb in DISKIO enthalten.
Wenn das Typ-Byte des Apple ($FBB3) kleiner als 8 ist (IIe: 7, IIc:0, II/II+: $EA), dann ist Kleinschreibung auf dem Keyboard möglich, genauso wie für einen modifizierten II+, dessen AND-Maske in der Monitor-Routine KEYIN auf $FF (Standard: $DF) gesetzt ist. In diesem Fall wird angenommen, daß dann auch Kleinschreibung auf dem Bildschirm möglich ist. Das Flag SHIFTUP wird entsprechend gesetzt.
Wenn Slot 3 belegt ist (konstanter READ), wird angenommen, daß hier eine 80-Zeichenkarte steckt. HMAX wird in diesem Fall von 40 auf 80 erhöht, und das Flag IS40 (Unterscheidung in HOME, CLEOP, CLEOL) wird zurückgesetzt. Die Adresse der 80-Zeichenkarte ($C300) wird in den „Char Out"-Vektor ($36/37) eingesetzt. Beim ersten Aufruf via COUT wird die Karte dann ihre eigenen Vektoren einsetzen. Da die 80-Zeichenkarte für den IIe/c sich (natürlich?) nicht durch die Speicherstellen CH und CV ($24/$25) steuern läßt, was die Positionierung des Cursors angeht, muß hierbei noch einmal zwischen „80 Z/Z Videx" und „80 Z/Z IIc/e" unterschieden werden. Wenn die Karte des IIc/e erkannt wird, wird das Flag IIEC auf $FF gesetzt, ansonsten bleibt es auf dem Wert 00.
(Hinweis: Wenn man dies gelesen hat, wird man auch verstehen, warum der als Zugabe auf der Peeker-Sammeldisk #16 enthaltene Applesoft-Editor nur für 40 Z/Z geschrieben wurde. us)

GETLINE (Z. 156-312) ist aufgrund der bereits erwähnten Problematik reichlich lang ausgefallen: Nach einer Positionierung auf VTAB 22, HTAB 1 wird auf das erste Zeichen via GETKEY, also der Standard-Input-Routine, gewartet. Ist dieses Zeichen kein Ctrl-E, wird die Kommandozeile mittels der Ctrl-X vom Schirm gelöscht.
Danach werden solange Zeichen geholt und an CMDSTR angehängt bzw. in CMDSTR eingesetzt, bis entweder ein <RETURN> auftaucht oder CMDSTR seine maximale Länge erreicht hat. CMDSTR kann maximal 2 Zeilen lang werden, d.h. 78 Zeichen bei 40 Z/Z und 158 Zeichen bei 80 Z/Z.

GOTLINE (Z.320-359) fängt wieder mit dem ersten Zeichen von CMDSTR an. Nach einem Test des Keyboards auf Ctrl-C wird die Tabelle CMDS (im File EDCMDS) nach dem CMDSTR-Zeichen abgesucht. CMDS enthält jeweils ein Zeichen und danach eine Startadresse. Wird das Zeichen aus CMDSTR positiv verglichen, wird die Startadresse eingesetzt und die entsprechende Routine aufgerufen. Innerhalb der Routine wird dann der CMDSTR ggf. weiter ausgewertet und der CMDIDX (Index in CMDSTR) entsprechend weiter verschoben. Wenn der Befehl erfolgreich beendet wurde, geht es mit NXTCMD (Z. 327) solange weiter, bis CMDSTR zu Ende ist.

CMDERR (Z.362-385) und DSPERR werden von allen möglichen Punkten her angesprungen: wenn ein Parameter „out of Range" ist oder fehlt, bei einer Fehlermeldung des DOS usw. Das Spezielle an dieser Routine ist der Einsprung mit gesetztem X-Register: X enthält den CMDIDX-Wert, an dem der Fehler erkannt wurde.
Nach Ausdruck der Fehlermeldung auf VTAB 24 wird CMDSTR erneut gedruckt und zu GETLINE gesprungen, allerdings ohne vorherige Positionierung auf VTAB 22 / HTAB 1. Der Cursor bleibt damit auf dem Fehler in CMDSTR stehen, bis die erste Taste gedrückt wird.

Festplattenkomplettlösung für jedermann

Mit der Firma Frank & Britting GmbH, die auf Festplatten spezialisiert ist, konnten wir ein extrem günstiges Sonderangebot aushandeln, das eine Festplattenkomplettlösung selbst für Apple-Besitzer mit kleinerem Geldbeutel erschwinglich macht. Sie können unter zwei Varianten wählen:

Luxus-Lösung: 20-Megabyte-Festplatte MDB20 (MDB = Mobile Datenbox) + Megaboard-Controller + Handbuch + 3 Konfigurationsdisketten + Anschlußkabel + DB-Meister-Dateiverwaltungsprogramm + Handbuch + 2 Programmdisketten zum Gesamtpreis von nur DM 3199,- inkl. MwSt.
Standard-Lösung: 10-Megabyte-Festplatte MDB10 + Megaboard-Controller + Handbuch + 3 Konfigurationsdisketten + Anschlußkabel + DB-Meister-Dateiverwaltungsprogramm + Handbuch + 2 Programmdisketten zum Gesamtpreis von nur DM 2799,- inkl. MwSt.
(jeweils 6 Monate Garantie.)

Wozu eine Festplatte?

Für eine Festplatte sprechen zwei Argumente, nämlich erstens der bedeutend größere Datenspeicher und zweitens die sehr hohe Zugriffszeit.
● 80- und 160-Spur-Laufwerke haben zwar einen größeren Datenspeicher als die normalen 35-Spur-Laufwerke, doch ist die Zugriffszeit hier wie dort bescheiden.
● RAM-Karten haben zwar eine hohe Zugriffszeit, doch ist der Datenspeicher flüchtig: Strom aus – Daten weg!
Für unsere Festplatten-Standard-Lösung (MDB10) gilt demgegenüber:
– Die Nettospeicherkapazität beträgt hier ca. 9.650K = ca. 9.880.000 Zeichen = ca. 15 Disketten mit 160 Spuren = ca. 69 Disketten mit 35 Spuren.
– Die Datenübertragungsrate beträgt auf unterster Ebene ca. 44K/s = ca. 45.000 Zeichen/s. Zum Vergleich beläuft sich die Übertragungsrate bei normalen Diskettenlaufwerken auf ca. 7,5K/s und bei RAM-Karten auf ca. 44K/s, d.h. RAM-Karten sind also keineswegs schneller, sondern meist sogar langsamer als die MDB-Festplatte.
Sie erhalten deshalb mit unserer Festplattenlösung einen großen externen Massenspeicher bei sehr hoher Datenübertragungsrate zu einem äußerst niedrigen Preis.

Unkomplizierte Installation

Die eigentliche Festplatte, also die „Mobile Datenbox" MDB, ist ein kleines rechtekkiges Kästchen mit der Länge einer Handfläche. Das eine Ende des Flachbandkabels wird in die MDB, das andere Ende in den Megaboard-Controller gesteckt, der wie eine normale Erweiterungskarte aussieht und in jeden Slot von 1-7 paßt. Die MDB hat ein eigenes Netzteil. Deshalb brauchen Sie jetzt nur noch die MDB und danach den Apple einzuschalten, und fertig ist die Installation. Im Gegensatz zur Megacore-Festplatte, die wegen des Apple-Netzteil-Wechsels von Apple-Händlern eingebaut werden sollte, ist die Installation der MDB von jedermann problemlos zu bewerkstelligen.

Problemlose Konfigurierung

Danach booten Sie die mitgelieferte Systemdiskette DOS-CONFIG von einem 35-Spur-Laufwerk, tippen „I" für Initialisieren, geben die Festplattenparameter ein (bei MDB10: 4 Köpfe, 306 Zylinder, Präkompensation ab dem 256. Zylinder und Schneller Zugriff Ja), und nach zwei weiteren Kontrollbestätigungen beginnt die Initialisierung. Dies dauert ca. 14 Minuten für die MDB10, und nunmehr kann die Festplatte mit den Betriebssystemen beschrieben werden.

Als Betriebssysteme können Sie im einzelnen verwenden:
– DOS 3.3,
– Diversi-DOS 2C und 4C,
– Apple-Pascal 1.1 und 1.2,
– ProDOS 1.0.1, 1.0.2, 1.1.1,
– CP/M 2.2 56K (z.Zt nur diese CP/M-Version).

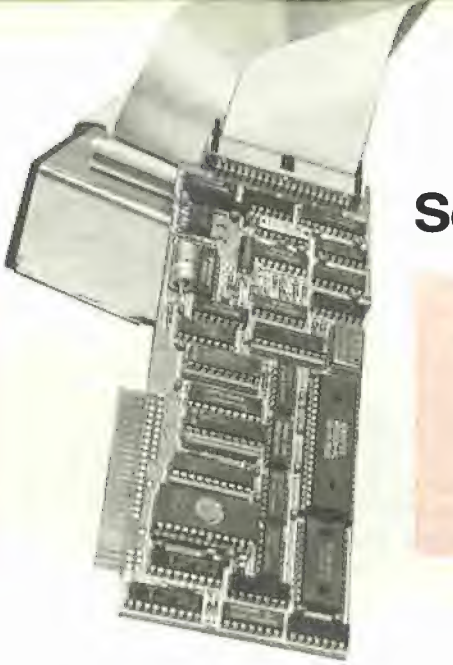
Die Konfigurierung, d.h. die Speicherverteilung auf die verschiedenen Betriebssysteme, sollte mit Bedacht überlegt werden. Nicht-benötigte Betriebssysteme können Sie natürlich weglassen. Nehmen wir an, Sie brauchen zunächst nur DOS 3.3. Wie wird nun konfiguriert?

Sie booten nach der obigen Initialisierung die Festplatte mit PR#7 und wählen aus dem automatisch erscheinenden Bildschirm-Menü „C" für „Configuration". Dann Tippen Sie „D" für DOS 3.3, danach „V" für Volumes, danach z.B. „10" für 10 Volumes = 2 x 10 x 140 = 2800K (1 Volume entspricht einem 35-Spur-Doppellaufwerk). Nun können Sie nach zwei weiteren Kontrollbestätigungen die neue Konfiguration auf die Diskette schreiben, was nur wenige Sekunden dauert. So einfach ist das! Später können Sie dann, sobald Sie mit der Festplatte besser vertraut sind, bei Bedarf die Konfigurationen für ProDOS, Apple-Pascal und CP/M nachholen.

Wie spricht man die MDB an?

Sie werden sich nun vielleicht fragen, wie Sie die Festplatte ansprechen müssen und ob Sie neue DOS-Befehle lernen müssen. Mitnichten! Booten Sie einfach die Festplatte mit PR#7 – übrigens heißt es immer PR#7, auch wenn Sie die Megaboard-Karte z.B. in Slot 4 gesteckt haben – und danach z.B.
INIT HELLO, S7, V3, D1
Damit wird das Volume in „Drive 1, Slot 7" formatiert. Nun können Sie mit
CATALOG
RUN usw.
auf dieses Volume wie üblich zugreifen.

er-Leser · Sonderangebot für Peeker-Leser ·



Wie schnell ist die MDB?

Die Datenübertragungsrate beträgt bei der MDB physikalisch 5 MBits/s. Dies ist ein theoretischer und niemals erreichbarer Wert. Auf Spur-Sektor- bzw. Blockebene beträgt die praktisch erzielbare Übertragungsrate ca. 44K/s bis 47,5K/s, je nach Betriebssystem. Einige anschauliche Beispiele: Auf Sektorebene läßt sich ein 140K-DOS-Volume mit dem Programm MDB.KOPY in 6,2s auf ein anderes 140K-Volume überspielen. Auf Dateiebene läßt sich eine 1-Megabyte-ProDOS-Datei mit dem Programm PROFID in 122s duplizieren. Dies sind umgerechnet 17.000 Bytes/s für Lesen bzw. Schreiben.

Was läuft mit der MDB?

Sie können normale Disk-II-Laufwerke und beispielsweise Erphi-160-Spur-Laufwerke im Mischbetrieb einsetzen. Die zusätzliche Verwendung von RAM-Karten ist bei der MDB überflüssig, da mit diesen in der Regel keine höheren Übertragungsraten erzielt werden können. Die Accelerator IIe funktioniert einwandfrei in Verbindung mit dem Megaboard-Controller (Slot-7-Schalter auf 1MHz einstellen). Die Speedemon funktioniert jedoch wegen des Cache-Speichers leider nicht.

DB-Meister

Speziell für die zukünftigen MDB-Besitzer wurde das Adreß- und Schemabriefprogramm DB-Meister, das in der normalen Diskettenversion DM 290,- kostet, zum Festplattenbetrieb umgeschrieben. Technische Daten:

– Datensatzlänge bis zu 230 Zeichen, aufteilbar in bis zu 25 Felder; maximal 1000 Datensätze pro Volume
– 2 variable Indexfelder und 1 zusätzliches, konstantes Suchfeld.
– Maskengenerator-Modul für die freie Definition von Bildschirmeingabe- und Ausdruck-Masken.
– Dateipflege-Modul zum Neueingeben, Ändern, Löschen, Ansehen, Bildschirm-Ausdruck usw.
– Sortier- und Filter-Module zum Sortieren und Untersortieren nach Indexfeldern und zum selektiven oder kumulierenden Filtern nach beliebigen Feldern.
– Druck-Modul zum Ausdrucken von Selbstklebe-Etiketten, tabellarischen Listen und Schemabriefen mit automatischem Einschub von Adressen und Anreden sowie weiteren Feldeinschüben im Briefkörper; Einzelblatt und Endlospapier verwendbar; Druckertreiber für Typenraddrucker mit anderer Typenradbelegung können eingebunden werden.
– Brief-Modul zum Schreiben der Schemabriefe; bei Bedarf separat verwendbar.
– Schnelles Backup-Programm (nur 10 Minuten für 2,5 Megabytes bzw. 18 DB-Meister-Dateien).
– Diverse Spezial-Module zur Indexfeld-Umdefinierung, zur Sortierung nach Kundennummern usw.

Alle Programm-Module laufen auf einem Apple IIe oder II+ mit MDB (Ausnahme: Das Brief-Modul funktioniert nur auf dem IIe mit 80-Zeichenkarte, doch können beliebige Fremdtexte eingelesen werden.).

Das DB-Meister-Programm eignet sich für normale Adreßverwaltung sowie für Dateien aller Art mit vielen, aber kleinen Datensätzen (max. 230 Zeichen). Das Programm ist nicht für das Schreiben von Rechnungen o.ä. gedacht.

Die Verarbeitungsgeschwindigkeit des „DB-Meisters" ist bei der MDB extrem schnell. Einige Sekundenangaben, bezogen auf eine Datei mit 500 Datensätzen zu je 220 Zeichen:

– Modul-Wechsel: 1s
– Suche nach Indexfelder: 0,3s
– Suche nach Nicht-Indexfeldern: 17s
– Filtern und Untersortieren: 25s
– Bildschirm-„Drucken": 38s

Noch ein etwas anschaulicheres Beispiel: Wenn Suchfelder neu vergeben werden sollen, muß die alte Indexdatei (ca. 50 Sektoren) und die ganze Hauptdatei (ca. 450 Sektoren) eingelesen werden und dann nach dem automatischen Heraussuchen der neuen Suchwörter die neue Indexdatei abgespeichert werden. Dieser normalerweise zeitraubende Prozeß dauert auf der MDB nur 14s.

Wie wird bestellt?

Sie senden Ihre Bestellung an den Hüthig-Software-Service. Sie erhalten dann von der Firma Frank & Britting eine Vorausrechnung, nach deren Überweisung Ihnen von dort die MDB10 bzw. MDB20, der Megaboard-Controller, das Handbuch und die Konfigurierungsdisketten mit 6 Monaten Garantie geliefert werden. Gleichzeitig erhalten Sie vom Hüthig-Software-Service das DB-Meister-Programm (2 Disketten und Handbuch) in der für die MDB bereits angepaßten Version. Nach einer geringfügigen Änderung im Hello-Programm können Sie diese Neuversion des DB-Meisters übrigens auch zusätzlich auf normalen 35-Spur-Laufwerken einsetzen.

Zur Bestellung können Sie eine der im Peeker eingehefteten Bestellkarten verwenden. Stichwort:

1 x MDB10-Sonderangebot für DM 2799,-
oder
1 x MDB20-Sonderangebot für DM 3199,-

Hüthig PUBLIKATION

SOFTWARE SERVICE
Im Weiher 10 · 6900 Heidelberg 1

Bildschirm- und PRINT-Routinen

SETUC (Z. 394-397) übersetzt von Klein- auf Großbuchstaben.
PRCHAR (Z. 399-415) druckt ein Zeichen. Bei 80 Z/Z ist SHIFTUP immer gelöscht, d.h. das Zeichen wird unverändert gedruckt. Bei 40 Z/Z wird das Bit 7 immer gesetzt. Wenn nur Großschreibung zulässig ist, wird bei Kleinbuchstaben vorher auf „Großbuchstabe" übersetzt und durch das Setzen von INVFLAG im Monitor der Kleinbuchstabe als inverser Großbuchstabe gedruckt.
PRADDR (Z. 417-422) druckt Akku und Y-Register als dreistellige Hexzahl, gefolgt von „- ".
PRTAY, PRTY, PRBYTE und **PRHEX** sind eine Kopie der Routinen aus dem Monitor; sie drucken A-Y als vierstellige Hexzahl bzw. über die jeweiligen anderen Einsprünge den Akku als zwei- und einstellige Hexzahl.
PRBITS (Z. 440-451) druckt den Akku-Inhalt als einzelne Bits aus und wird für die Darstellung der VBM (ProDOS) bzw. der VTOC (DOS 3.3) benötigt.

CLRDSP (Z. 472-481) löscht den Display-Teil des Bildschirms, nämlich die Zeilen 5 bis 20. Man hätte hier auch mit dem Setzen von WNDBTM ($23), also der Untergrenze des Textfensters, arbeiten können, aber es gibt tatsächlich 80-Zeichenkarten, die sich um diese Speicherstelle nicht kümmern und den gesamten Bildschirm löschen.

HOME, CLEOP, CLEOL, SETCV und **SETCH** (Z. 488-527): Bedingt durch die 80-Zeichenkarte des IIc/e, die fröhlich alle Versuche ignoriert, den Cursor auf „legale" Weise zu setzen, muß hier eine Unterscheidung mit Hilfe des Flags IIEC getroffen werden, das in INSTALL entsprechend gesetzt wurde. Für alle anderen Konfigurationen werden in gewohnter Weise die Speicherstellen CH ($24) und CV ($25) gesetzt, für den IIc/e sind es eben die Zwischenspeicherstellen des TEXT-Bildschirms, die von dieser Karte benutzt werden.
Diese verquere Art und Weise bedingt den Einsatz der Speicherstellen EDCH und EDCV, in denen die Cursor-Position immer intern mitgezählt wird. Allerdings trifft das für EDCH nur bedingt zu; diese Speicherstelle wird nur durch ein „POKE 36,xx", also ein direktes Setzen eines (H)TAB-Wertes verändert – glücklicherweise brauchen wir nicht jedes einzelne Zeichen mitzuzählen.
HOME arbeitet wie üblich: Zuerst wird VTAB 1 / HTAB 1 gesetzt, danach folgt ein Clear to End of Page, also eine Löschung bis Seitenende, bei dem zwischen „Monitor" und „80 Z/Z" unterschieden werden muß, weil der Monitor keine derartigen Steuerzeichen akzeptiert (auch das bekannte „ESC Shift-Ctrl P" funktioniert nur von der Tastatur aus). Erstaunlicherweise sind die Steuerzeichen für Videx und 80 Z/Z des IIc/e in diesen beiden Fällen tatsächlich dieselben...
Innerhalb von CLEOL (Clear to End of Line) haben wir leider einen echten Vorgriff auf den nächsten Teil dieser Aufsatzfolge: Alle Funktionen des Programms, die einen Output auf den Bildschirm erzeugen, lassen dasselbe auch für einen angeschlossenen Drucker zu – nur daß die meisten Drucker ein CLEOL-Steuerzeichen nicht spurlos verdauen würden. Folglich wird beim Modus „Drucker an" (PRINTER-Flag = $FF) dieses Steuerzeichen nicht gesendet.

Die Argument-Berechnung

GETPRM (Z. 533-576): Auch dies ist ein echter Vorgriff, weil im Moment noch keine einzige Funktion implementiert ist, die ein oder zwei Argumente benötigt. Diese Routine wird aber von mehreren späteren Kommandos benutzt und gehört deshalb in ED.FRAME.
Nehmen wir eine dieser Funktionen, z.B: „D"ump. Wenn „D" ohne folgende Parameter eingegeben wird, erfolgt ein DUMP von $0000 bis zum Ende des Puffers (MAXBUF) bzw. bis zum Ende des Display-Feldes.
Bei gegebener Startadresse, also z.B. „D30" wird ab $0030 geDUMPT, wiederum entweder bis zum Puffer-Ende oder bis zum Ende des Display-Feldes. GETPRM stellt hier fest, daß eine Startadresse angegeben wurde, und setzt nur die Default-Endadresse (MAXBUF).
Schlußendlich: Wenn sowohl Start- als auch Endadresse angegeben sind, (z.B. „D30,133", dann wird eben von $0030 bis $0133 geDUMPt.

EVALARG (Z. 580-593) benutzt zuerst EVALHEX und überprüft dann, ob das erhaltene Argument im zulässigen Bereich (0...MAXBUF) liegt. Falls nicht, wird ein gesetztes Carry returnt, genauso, wenn EVALHEX keine Hexzahl auswerten konnte.

EVALHEX (Z. 587-612) ist eine generelle Auswertungsroutine für Hexadezimalstrings, deren Studium zwecks späterer Verwendung in eigenen Programmen sicher lohnend ist.
Der höherwertige Teil des Ergebnisregisters EVREG wird auf Null gesetzt, danach wird das nächste Zeichen aus CMDSTR geholt und via SETHEX von ASCII auf eine Zahl umgerechnet. SETHEX returnt mit gesetztem Carry, wenn dieses Zeichen keine gültige Hexzahl ist. Das führt, wenn es sich um das erste Zeichen innerhalb von EVALHEX handelt, zum Ende von EVALHEX mit gesetztem Carry, d.h. EVALHEX hat überhaupt keine Hexzahl gefunden. Ansonsten wird EVREG mit dem Ergebnis gesetzt. In EVLOOP wird das nächste Zeichen aus CMDSTR geholt. Falls es sich um eine weitere Hexzahl handelt, wird EVREG (und EVREG+1) mit 16 multipliziert und CMDIDX solange erhöht, bis keine weitere Hexzahl mehr in CMSTR folgt. EVALHEX returnt dann mit Carry clear („Hexzahl gefunden") und entsprechend berechnetem EVREG. (Es soll ja tatsächlich noch Programme geben, die sich bei der Eingabe von „E" anstelle von „$000E" mit einer Fehlermeldung beschweren...)

SETHEX (Z. 615-622) benutzt TSTHEX. Wenn TSTHEX mit Carry clear returnt, also positiv auf „Hexzahl" erkannt hat, wird umgerechnet, ansonsten returnt SETHEX ebenfalls mit gesetztem Carry.

TSTHEX (Z. 624-634) sieht etwas aufwendig für einen schlichten Test auf „Hexzahl" aus. Hier geht es aber darum, den Akku-Inhalt unverändert zu lassen und nur das Carry entsprechend zu setzen. Es erfolgt ein Return mit gesetztem Carry, wenn der Akku keine ASCII-Hexzahl enthält.

GLOBALS (Z. 642-694): Hier stehen zahlreiche Variablen, von denen die meisten in diesem Teil 1 noch überhaupt keine Funktion haben. Diese Anhäufung hat zwei Gründe:
a) Damit man die Variablen bei der Initialisierung des Programms alle auf einmal löschen kann, sollten Sie in einem zusammenhängenden Speicherbereich stehen.
b) Wenn dieser Speicherbereich wie etwa die Tabelle CMDS am Ende des Programms steht, verschiebt er sich bei den nächsten Erweiterungen laufend nach „oben". Für eine Tabelle wie CMDS ist das unerheblich, für Variablen wird der „Entwanzungs"prozeß damit erheblich erschwert, wenn z.B. CMDIDX plötzlich an einer ganz anderen Speicherstelle sitzt.

Erwähnenswert ist vielleicht noch die Zeile 694: Hier wird dafür gesorgt, daß die folgenden Speicheradressen konstant bleiben. Man kann zwischen den Labels GLOBAL und LGLBL bis zu 10 weitere Bytes einbauen oder eine beliebige Anzahl herauskürzen, ohne daß sich die Speicheradressen der nachfolgenden Routinen dabei ändern.
Hinter den Variablen stehen noch diverse Fehlermeldungen, die ebenfalls bereits vollständig sind, damit wir sie nicht jedesmal zusammen mit den späteren Erweiterungen erneut auflisten müssen.

4. ED.DISKIO

4.1. Allgemeine Beschreibung

ED.DISKIO enthält alle betriebssystemspezifischen Definitionen und Routinen zum Lesen und Beschreiben einer (RAM-) Disk. Durch diese Zusammenfassung ist das gesamte restliche Programm einschließlich der PRINT-Routinen theoretisch unter jedem Apple-Betriebssystem (DOS 3.3, ProDOS, UCSD-Pascal und CP/M) lauffähig. Da der Disk-Editor jedoch speziell für ProDOS gedacht ist, sind keine DISKIO-Routinen für die anderen Betriebssysteme vorgesehen.

4.2. Problemstellungen

Das Abhängen des BASIC.SYSTEM ist auf jeden Fall notwendig: ein Ctrl-D liefert sonst anstelle eines DELETE in der Kommandozeile einen SYNTAX ERROR, weil das BASIC.SYSTEM danach ein DOS-Kommando erwartet.
READ BLOCK und WRITE BLOCK sind zusammen mit dem MLI überhaupt kein Problem, weil die entsprechenden Befehle „user"-zugänglich implementiert sind. Wesentlich schwieriger gestaltet sich das Lesen des ersten Blocks eines Files:
ProDOS liefert nach einem OPEN und einem ersten READ den ersten Datenblock des betreffenden Files. Der eventuell vorhandene Indexblock bleibt dem Benutzer verborgen, denn ein Befehl „liefere Indexblock, wenn vorhanden" ist nicht implementiert. Aus diesem Grunde ist die Lösung zwangsläufig etwas unsauber ausgefallen.
Zuerst die einfachere Lösung: Der reservierte Dateipuffer von $0400 Byte wird von ProDOS in zwei Hälften geteilt: $0200 Bytes für Daten und $0200 Bytes für den Indexblock. Falls die Datei den Storage Type 1 hat, bleibt dieser Indexteil leer, während für eine Datei mit Storage Type 3 der „momentane" Indexblock eingeladen wird. Es wäre nun sehr einfach, vom ProDOS-Dateipuffer aus den Indexteil in den Arbeitspuffer zu kopieren – dieser Weg liefert aber auch für einen File mit Storage Type 3 den ersten Indexblock und nicht den Masterblock zurück.
Deshalb folgt hier die kompliziertere Lösung: Innerhalb des angegebenen Pathname wird nicht der File selber, sondern das Directory, in dem der File als Eintrag steht, gesucht. Dieses Directory wird via OPEN und READ gelesen; der Suchprozeß nach dem Filenamen findet innerhalb von ED.DISKIO statt. Die Struktur der Directory-Einträge ist bekannt und von Apple dokumentiert – im Gegensatz zur internen Benutzung der ProDOS-Dateipuffer.
Wenn der File im Directory gefunden wurde, wird aus dem File-Eintrag die Nummer des ersten Blocks herausgeholt, und dieser Block wird dann via READ BLOCK in den Arbeitspuffer gelesen.

4.3. Ausgewählte Routinen

MAXBUF legt die Größe eines Sektors (Blocks) auf der zu bearbeitenden Diskette und gleichzeitig die Größe des Arbeitspuffers fest. MAXBUF hat für ProDOS-Disketten den Wert 02 ($0200 Byte pro Block).

DISCONN (Z. 9-14) ist die Routine zum Abhängen des BASIC.SYSTEMs. Hier werden einfach die Vektoren, zu denen das BASIC.SYSTEM die Ein/Ausgabe leitet, nachdem auf „DOS-Befehl" überprüft wurde, direkt in die Speicherstellen auf der Zeropage eingesetzt. Als Folge davon hängt das BASIC.SYSTEM danach nicht mehr „dazwischen".
DOSCONN (Z. 16-23) ist das Gegenstück zu DISCONN: das BASIC.SYSTEM wird wieder „eingeklinkt". Die Vektoren von der Zeropage werden dabei in die Global Page zurückkopiert, denn sie könnten durch ein zwischenzeitliches „PR# 3" verändert worden sein.

EXIT (Z.25) ist das Ende von EDIT, das sich unter ProDOS recht einfach gestaltet: die Funktion „Q"uit springt hierher, und es erfolgt ein Kaltstart des BASIC.SYSTEM (Warmstart von Applesoft) über einen der Page-3-Vektoren. Dieses Verfahren ist unter DOS 3.3 exakt dasselbe.

READ BLOCK/PREFIX (Z. 31-35) ist syntaxmäßig innerhalb von EDIT exakt dasselbe wie WRITE BLOCK/PREFIX. Deshalb beschränkt sich die Ausführung auf das Setzen eines Flags und einen Sprung zu WRITE.

WRITE BLOCK/PREFIX (Z. 41-240): Das nächste Zeichen in CMDSTR wird überprüft. Wenn es sich dabei nicht um ein „P" handelt, folgt ein Sprung zu R/WBLOCK. Ansonsten muß das Kommando „RP" oder „WP" sein, also READ oder WRITE PREFIX. Wenn hier das nächste Zeichen in CMDSTR ein „/" ist, muß es sich um einen folgenden Pathname handeln, andernfalls um eine Angabe im Format „X,Y" für Slot und Drive.
SUNIT (Z. 52..) wertet Slot und Drive entsprechend aus und berechnet daraus die Unitnummer für ProDOS.
SETINFO (Z. 79...) trägt Slot und Drive in den String für PRINFO ein und endet mit einem Sprung zu PRINFO, wo die neue Information ausgedruckt wird.
SPREFIX (Z. 96...) holt den Volume-Namen via GETPATH aus CMDSTR und führt damit ein SET PREFIX durch. Die Unitnummer der zuletzt benutzten Unit auf der System Global Page („last recently used unit") wird in Slot und Drive zerlegt. SPREFIX endet mit einem Sprung zu SETINFO und damit mit dem Ausdrucken des gefundenen Slot und Drive.

GETPATH (Z. 141-151) kopiert den Path aus CMDSTR unter Benutzung von GNCHAR nach PATH. Der Speicherbereich PATH wird von verschiedenen Stellen innerhalb von ED.DISKIO für den Pathname für MLI-Kommandos benutzt. In GETPATH wird auch gleichzeitig noch die Position des zuletzt gefundenen „/" festgehalten (Speicherstelle LNAME). Diese Information wird jedoch von SPREFIX ignoriert, denn sie ist nur für die „§/Volname/Filename"-Funktion von Interesse.
GNCHAR (Z. 153-159) holt das nächste Zeichen aus CMDSTR und returnt mit gesetztem Z-Flag („EQ"), wenn CMDSTR zu Ende ist oder es sich bei diesem Zeichen um ein <Space> oder um ein „!" handelt. Diese beiden Zeichen werden als „Pathname-Ende" interpretiert, denn sie sind in Filenamen unter ProDOS ohnehin nicht zugelassen.

RWBLOCK (Z. 163-240) testet als erstes, ob hinter dem „R" oder „W" noch ein Argument folgt. Dieses Argument kann entweder ein „+", ein „-" oder die Nummer des betreffenden Blocks sein.
READ ändert dabei immer die Blocknummer von WRITE mit, WRITE ändert nur die WRITE-Blocknummer!
Die neue(n) Blocknummer(n) werden via PRINFO angezeigt (Z. 202). Danach folgt entweder ein WRITE des Arbeitspuffers oder ein READ des entsprechenden Blocks in den Arbeitspuffer hinein. Nach einem READ wird der neue Inhalt des Puffers über DUMP (s. nächster Teil) auf dem Bildschirm angezeigt. RWBLOCK ist damit beendet.

§FILENAME (Z. 248-363) setzt als erstes (intern) READ – die Blocknummer des gefundenen Blocks wird am Ende angezeigt und soll sowohl die READ- als auch die WRITE-Blocknummer sowie die entsprechenden Slot- und Drivenummern setzen.
Wenn der angegebene Path mit einem „/" beginnt, wird der (eventuell durch „RP" gesetzte) interne Path überschrieben, ansonsten wird dieser Path mit einbezogen. Ist kein interner Path gesetzt, verabschiedet sich §FILENAME mit „No Prefix?"
Der angegebene Pathname wird mittels GETPATH kopiert, wobei der Index zum letzten „/" festgehalten wird. Der erste Teil des Pathname (also der Teil vor dem letzten „/") wird als PATH gesetzt, der zweite Teil landet im Speicherbereich FNAME, wobei hier jeweils das Bit 7 der

ASCII-Zeichen ausgeblendet werden muß, denn ProDOS speichert die Filenamen mit „MSB off" im Directory.
Wenn innerhalb des angegebenen Path der letzte „/" auf der ersten Stelle steht, haben wir nur einen einzigen Filenamen. In diesem Fall wird die Volume Directory geladen (Z. 262 und 278 ff).

BADFNAME (Z. 285-291) wird angesprungen, wenn ein MLI-Fehler aufgetreten ist oder ein ungültiger Pathname angegeben wurde. Die Meldung „Illegal Pathname/Volume/File not found" wird ausgegeben.
SETDIR (Z. 299-315) führt ein GET FILE INFO durch, um zu verifizieren, daß es sich beim „letzten Directory vor dem File" wirklich um ein Directory handelt, und liest danach den ersten Block dieses Directory über OPEN und READ des MLI ein. Der erste Eintrag in diesem ersten Block enthält die Länge der jeweiligen File-Einträge (ELEN) und die Anzahl der File-Einträge pro Directory-Block (ECOUNT).
NXTBLOCK (Z. 317-343) sucht diesen Block nach dem Filenamen ab und springt zu GOTFILE, wenn der File-Eintrag gefunden wurde.
BLOCKEND (Z. 345-350) lädt den nächsten Directory-Block. Falls das Directory zu Ende ist, meldet das MLI den Fehler „EOF encountered". Damit ist die Suche beendet, und es folgt ein Sprung zu BADFNAME, ansonsten geht es weiter mit NXTBLOCK.

GOTFILE (Z. 352-357) holt aus dem gefundenen File-Eintrag die Nummer des ersten Blocks heraus (Bytes $11 und $12 des File-Eintrags).
GOTVDIR (Z. 358-363) ist der Einsprung zum Lesen des Volume-Directory: Hier ist die Blocknummer bereits bekannt ($0002). Sie wird in RBLOCK *und* WBLOCK gesetzt. GETUNIT berechnet aus der Unitnummer Slot und Drive und setzt sie für PRINFO. Danach folgt ein Sprung zu DORW – hier wird der entsprechende Block in den Arbeitspuffer gelesen und das INFO neu gedruckt.

Die Zeilen 371-405 enthalten die entsprechenden MLI-Aufrufe für GET FILE INFO, OPEN und READ der Directory.

PRINFO (Z. 409-444) steht ebenfalls in ED.DISKIO: Da je nach (softwaremäßigem) Format der Zieldiskette entweder mit Blocknummern (ProDOS), Tracks/Records (CP/M) oder mit Tracks und Sektoren (DOS 3.3) gearbeitet wird, ist das Format der Anzeige ebenfalls leicht unterschiedlich. PRINFO druckt vier Leerzeichen, wenn der Ausdruck auf den Bildschirm geht, für einen Ausdruck auf den Drucker werden diese Leerzeichen weggelassen. Die Länge des INFO selber ist so abgestimmt, daß bei 40 Z/Z zwei untereinanderstehende Zeilen gedruckt werden; bei 80 Z/Z ist es nur eine Zeile. Dementsprechend wird vorher VTAB 2 (40Z) oder VTAB 3 (80Z) gesetzt.

5. ED.FUNCS1

5.1. Allgemeine Beschreibung

ED.FUNCS1 (und ED.FUNCS2) enthalten die nicht-betriebssystem-spezifischen Funktionen von EDIT, wie z.B. Bytes ändern, darstellen, suchen, vergleichen usw. In dieser ersten Ausbaustufe von EDIT sind nur zwei Funktionen definiert:
X – Aufruf des Monitors und Anhängen des BASIC.SYSTEM
Q – Ende von EDIT.

5.2. Problemstellungen

Innerhalb des Monitors kann man alles Mögliche anstellen – von einem einfachen LIST des Puffer-Inhaltes über ein CATALOG der Ziel-Diskette bis hin zu einem RESET. Aus diesem Grund muß auf jeden Fall die Returnadresse innerhalb des Command Interpreters von EDIT sicher zwischengespeichert werden. Des weiteren wird mit absoluter Sicherheit der Bildschirmaufbau von EDIT demoliert – nach einem Return aus dem Monitor muß der Bildschirm neu gedruckt werden.

5.3. Ausgewählte Routinen

XMON (Z. 6-48) speichert zuerst die EDIT-interne Returnadresse, kopiert danach CMDSTR auf einen sicheren Platz (CMDSTR befindet sich ab $0200 im normalen Input-Buffer) und setzt danach die Ctrl-Y-Funktion des Monitors auf MONRTS. Nach Anhängen des BASIC.SYSTEM via DOSCONN wird in den Monitor gesprungen.
MONRTS wird vom Monitor her durch die Ctrl-Y-Funktion her angesprungen. Hier wird das BASIC.SYSTEM wieder abgehängt, CMDSTR wieder nach $0200 kopiert und die interne Returnadresse wieder auf den Stack gebracht.
Es folgt REPRINT: TITLE wird aufgerufen. Hier wird der Schirm gelöscht, „PRODOS SECTOR EDITOR..." und zwei Zeilen mit „----" geschrieben. PRINFO druckt danach „Source: Slot X,D Y..." aus. SETDUMP ist momentan ein RTS und wird später den Puffer-Inhalt neu ausdrucken. Das letzte, was uns jetzt noch fehlt, ist die Kommandozeile. Sie muß natürlich ebenfalls neu gedruckt werden, und zwar vollständig.

RTYP2 – normalerweise benutzt, um den Cursor nach einem INSERT oder DELETE beim Eintippen der Kommandozeile wieder auf die richtige Stelle zu bringen – druckt jetzt den ersten Teil von CMDSTR, der bereits abgearbeitet ist (also links vom Cursor). Danach springt das Programm zu RETYPE, wo zuerst der rechte (noch nicht abgearbeitete) Teil von CMDSTR neu gedruckt wird, danach noch einmal der linke Teil: Damit steht sogar der Cursor wieder an der richtigen Stelle.
QUIT (Z. 56-58) sollte selbsterklärend sein.

6. ED.FUNCS2 und ED.CMDS

ED.FUNCS2 enthält momentan nur die nicht definierten Routinen SETDUMP und DUMP, die beide mit einem RTS enden. Die Funktion GETVAR endet mit einem Sprung zu CMDERR – die Benutzung von Variablen anstelle von Argumenten ist ebenfalls noch nicht definiert.

ED.CMDS umfaßt die Tabelle CMDS mit den Kommando-Zeichen und jeweils folgenden Startadressen sowie Definitionen der verschiedenen von EDIT belegten Pufferbereiche, von denen in dieser Ausbaustufe die meisten noch nicht verwendet werden.
Die Startadresse des Arbeitspuffers sollte wegen der Erweiterungen unverändert bleiben und ist deshalb gleich auf $2000 gesetzt. An dieser Stelle wird EDIT im „Vollausbau" enden.

Hinweis:

Es folgen noch die Teile 2 und 3 dieses dreiteiligen Beitrages. Im Teil 2 wird u.a. auch die komplette Befehlssyntax vorgestellt, so daß Sie mit der Benutzung des Editors noch bis zum nächsten Peeker-Heft warten sollten.
Die Sammeldisk # 17 enthält den kompletten Quellcode, die Disk # 18 den kompletten Objektcode.

ED.FRAME.TXT

```
0000:             1         SBTL 'DISK EDITOR'
0000:             2 PTR:    EQU $00         ;diverse Zwecke
0000:             3 *
0000:             4 CH:     EQU $24         ;Cursor horizontal
0000:             5 CV:     EQU $25         ;Cursor vertikal
0000:             6 INVFLAG: EQU $32        ;für COUT
0000:             7 COUTV:  EQU $36         ;O/I-Vektoren
0000:             8 *
0000:             9 CMDSTR: EQU $200        ;Buffer für Commands
0000:            10 PATH:   EQU $2A0        ;interner Pathname
0000:            11 YVEC:   EQU $3F9        ;Ctrl-Y Funktion
0000:            12 *
0000:            13 KEYBOARD: EQU $C000
0000:            14 KBSTROBE: EQU $C010
0000:            15 *
0000:            16 BELL:   EQU $FF3A       ;Piep!
0000:            17 VCALC:  EQU $FC22       ;Cursorpositionierung
0000:            18 GETKEY: EQU $FD0C       ;Input 1 Zeichen
0000:            19 COUT:   EQU $FDED       ;PRINT Zeichen
0000:            20 MONITOR: EQU $FF69      ;Monitor-Einsprung
0000:            21 *
0000:            22         MSB ON
0000:            23 *
00A1:            24 STOP:   EQU '!'         ;Kommando-Trennung
0000:            25 *
----- NEXT OB FILE NAME IS EDIT1
0803:            26         ORG $803
0803:            27 *
0803:20 0F 08    28 START: JSR INIT         ;clear BUFFER&FLAGS
0806:A2 FF       29 RESTART: LDX #$FF
0808:9A          30         TXS             ;Stackpointer setzen
0809:20 2F 08    31         JSR INSTALL     ;UC/lc, 80 Z
080C:4C 44 09    32         JMP GETLINE
080F:            33 *
080F:            34 ************************
080F:            35 *
080F:A9 20       36 INIT:  LDA #<BUFFER     ;Adresse hi
0811:8D 1A 08    37         STA CLRBUF+2
0814:A2 00       38         LDX #0
0816:8A          39         TXA
0817:A8          40         TAY
0818:9D 00 20    41 CLRBUF: STA BUFFER,X
081B:E8          42         INX
081C:D0 FA       43         BNE CLRBUF
081E:EE 1A 08    44         INC CLRBUF+2    ;Ziel+$100
0821:C8          45         INY
0822:C0 02       46         CPY #MAXBUF
0824:90 F2       47         BCC CLRBUF
0826:A2 31       48         LDX #LGLBL-GLOBALS-1
0828:9D 94 0C    49 NOFLAGS: STA GLOBALS,X
082B:CA          50         DEX             ;GLOBALS auf 00
082C:D0 FA       51         BNE NOFLAGS
082E:60          52         RTS
082F:            53 *
082F:            54 *
082F:20 7D 0D    55 INSTALL: JSR DISCONN ;DOS/BASIC.SYSTEM ab
0832:A9 28       56         LDA #40
0834:8D 97 0C    57         STA HMAX        ;Annahme: 40 Zeichen
0837:A9 FF       58         LDA #$FF
0839:8D 9A 0C    59         STA IS40        ;Flag für HOME etc.
083C:A9 00       60         LDA #0
083E:8D A0 02    61         STA PATH        ;interner Path auf ""
0841:AE B3 FB    62         LDX $FBB3
0844:E0 08       63         CPX #$08        ;Test: IIe / IIc?
0846:90 09       64         BCC STCASE      ;hat Lowercase
0848:AE 83 FD    65         LDX $FD83
084B:E0 FF       66         CPX #$FF        ;modifizierter II+?
084D:F0 02       67         BEQ STCASE
084F:A9 FF       68         LDA #$FF        ;nur UPPERCASE
0851:8D 98 0C    69 STCASE: STA SHIFTUP
0854:            70 *
0854:20 9D 08    71         JSR TESTS3      ;Test Slot 3
0857:8D 94 0C    72         STA TEMP
085A:20 9D 08    73         JSR TESTS3      ;zweiter Test
085D:4D 94 0C    74         EOR TEMP
0860:D0 45       75         BNE STARTZ      ;Read nicht konstant
0862:8D 98 0C    76         STA SHIFTUP     ;= 00
0865:8D 9A 0C    77         STA IS40        ;40 Z-Flag zurück
0868:85 36       78         STA COUTV
086A:AA          79         TAX             ;= 00
086B:A9 50       80         LDA #$50        ;80 Zeichen
086D:8D 97 0C    81         STA HMAX
0870:A9 C3       82         LDA #$C3
0872:85 37       83         STA COUTV+1     ;COUT auf $C300
0874:AD 05 C3    84         LDA $C305       ;Test auf 80Col
0877:49 38       85         EOR #$38        ;$Cx05: $38
0879:D0 11       86         BNE INIT80
087B:AD B3 FB    87         LDA $FBB3       ;MACHID für IIc/e
087E:C9 08       88         CMP #$08
0880:B0 0A       89         BCS INIT80      ;nicht IIe/c
0882:A9 FF       90         LDA #$FF
0884:8D 99 0C    91         STA IIEC        ;Flag: 80COL IIc/e
0887:8D 0F C0    92         STA $C00F       ;IIe/c: ALTCHARSET
088A:D0 1B       93         BNE STARTZ      ;"always"
088C:            94 *
088C:BD 99 08    95 INIT80: LDA SINIT,X  ;z.B. Ctrl-Z "3"
088F:F0 16       96         BEQ STARTZ      ;für VIDEX
0891:20 ED FD    97         JSR COUT
0894:E8          98         INX
0895:D0 F5       99         BNE INIT80
0897:F0 0E      100         BEQ STARTZ      ;"always"
0899:           101 *
0899:9A B3 00   102 SINIT: DFB $9A,$B3,00,00 ;Ctrl-Z "3"
089D:           103 *
089D:A2 00      104 TESTS3: LDX #0          ;Prüfung auf
089F:8A         105         TXA             ;konstanten Read
08A0:5D 00 C3   106 TSTS31: EOR $C300,X  ;von Slot 3
08A3:E8         107         INX
08A4:D0 FA      108         BNE TSTS31
08A6:60         109         RTS
08A7:           110 *
08A7:           111 *
08A7:AD 97 0C   112 STARTZ: LDA HMAX        ;max. Länge von
08AA:38         113         SEC             ;CMDSTR: 2 Zeilen
08AB:E9 01      114         SBC #01
08AD:0A         115         ASL A
08AE:8D 9D 0C   116         STA MAXCMD
08B1:           117 *
08B1:20 BD 08   118         JSR TITLE       ;"SECTOR EDITOR..."
08B4:20 75 10   119         JSR PRINF0      ;"Source, Target..."
08B7:A9 00      120         LDA #00
08B9:8D 9F 0C   121         STA CMDEND      ;Backup-CMDSTR auf ""
08BC:60         122         RTS
08BD:           123 *
08BD:20 97 0B   124 TITLE: JSR HOME
08C0:AD 97 0C   125         LDA HMAX        ;Zeichenzahl minus
08C3:38         126         SEC             ;String-Länge
08C4:E9 19      127         SBC #IDEND-PGMID
08C6:4A         128         LSR A           ;Mittenzentrierung
08C7:20 89 0B   129         JSR SETCH
08CA:A2 00      130         LDX #00
08CC:BD ED 08   131 TTL1:  LDA PGMID,X   ;"PRODOS SECTOR..."
08CF:F0 06      132         BEQ TTL2        ;<00> = Stringende
08D1:20 ED FD   133         JSR COUT
08D4:E8         134         INX
08D5:D0 F5      135         BNE TTL1
08D7:A9 04      136 TTL2:  LDA #4          ;VTAB 4
08D9:20 DE 08   137         JSR TTL3        ;"--------"
08DC:A9 15      138         LDA #21         ;VTAB 21
08DE:20 55 0B   139 TTL3:  JSR SETVTB
08E1:AE 97 0C   140         LDX HMAX
08E4:A9 AD      141         LDA #'-'
08E6:20 ED FD   142 TTL4:  JSR COUT        ;"----"...
08E9:CA         143         DEX
08EA:D0 FA      144         BNE TTL4
08EC:60         145         RTS
08ED:           146 *
08ED:D0 D2 CF   147 PGMID: ASC 'PRODOS   SECTOR EDITOR V1.0'
0906:00         148 IDEND: DFB 00
0907:           149 *
0907:           150 *
0907:           151 *
0907:           152 *********************
0907:           153 *-----MAINLOOP-----*
0907:           154 *********************
0907:           155 *
0907:20 47 0B   156 NEWCMD: JSR SETV221  ;VTAB 22, HTAB 1
090A:20 9C 0B   157         JSR CLEOP       ;Clear ab Cursor
090D:A9 00      158         LDA #00
090F:8D 9E 0C   159         STA CMDIDX      ;=> CMDSTR = ""
0912:8D 9F 0C   160         STA CMDEND      ;=> BACKUP = ""
0915:8D FE 09   161         STA IFLAG       ;kein INSERT
0918:8D FF 09   162         STA CAPSFLG     ;kein LC-Shift
091B:60         163         RTS
091C:           164 *
091C:AC 9E 0C   165 RETYPE: LDY CMDIDX      ;druckt CMDSTR nach
091F:CC 9F 0C   166 RTYP1: CPY CMDEND      ;DELETE oder INSERT
0922:F0 09      167         BEQ RTYP2       ;neu und setzt
0924:B9 00 02   168         LDA CMDSTR,Y    ;den Cursor
0927:20 E6 0A   169         JSR PRCHAR      ;nächstes Zeichen
092A:C8         170         INY
092B:D0 F2      171         BNE RTYP1       ;"always"
092D:20 9C 0B   172 RTYP2: JSR CLEOP
0930:20 47 0B   173         JSR SETV221     ;VTAB 22, HTAB 1
0933:A0 00      174         LDY #00
0935:CC 9E 0C   175 RTYP3: CPY CMDIDX      ;CMDSTR zu Ende?
0938:F0 09      176         BEQ RTYP4       ;OK- Fertig
093A:B9 00 02   177         LDA CMDSTR,Y
```

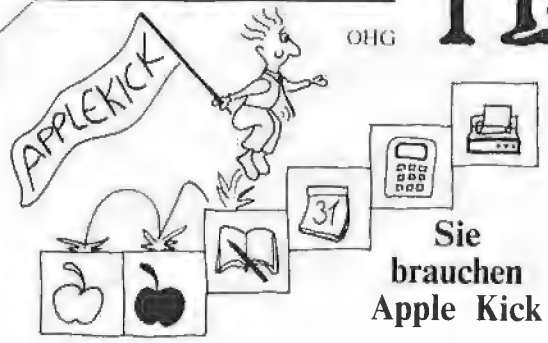
```
093D:20 E6 0A  178         JSR PRCHAR     ;nächstes Zeichen
0940:C8        179         INY
0941:D0 F2     180         BNE RTYP3      ;"always"
0943:60        181 RTYP4:  RTS
0944:          182 *
0944:          183 ******************
0944:          184 *----GETLINE----*
0944:          185 ******************
0944:          186 *
0944:20 4F 0B  187 GETLINE: JSR SETV22   ;VTAB 22, HTAB 1
0947:20 0C FD  188 GETLNZ:  JSR GETKEY
094A:A9 00     189         LDA #00
094C:8D 9E 0C  190         STA CMDIDX     ;Index in CMDSTR
094F:20 E6 0A  191         JSR PRCHAR     ;Cursorpos. Videx
0952:AD 00 C0  192         LDA KEYBOARD   ;wg. 80Z-//e
0955:09 80     193         ORA  #$80
0957:C9 85     194         CMP  #$85      ;Ctrl-E: (Re)-EDIT?
0959:F0 07     195         BEQ EDLINE     ;ja, stehenlassen!
095B:48        196         PHA            ;nein, erstes Zeichen
095C:20 07 09  197         JSR NEWCMD     ;löscht CMDSTR
095F:68        198         PLA
0960:D0 0F     199         BNE GTL1       ;"always"
0962:          200*
0962:20 53 0B  201 EDLINE  SETV24         ;eventuelle Fehler-
0965:20 B2 0B  202         JSR CLEOL      ;meldungen löschen
0968:20 47 0B  203         JSR SETV221    ;VTAB 22, HTAB 1
096B:          204 *
096B:          205 *
096B:20 0C FD  206 NXTCHAR: JSR GETKEY
096E:AD 00 C0  207         LDA KEYBOARD   ;wg. 80Z-//e
0971:09 80     208 GTL1:   ORA #$80
0973:C9 A0     209         CMP #$A0       ;kleiner <Space>?
0975:90 03     210         BCC ISCTRL
0977:4C 00 0A  211         JMP ISCHAR     ;druckbares Zeichen
097A:          212 *
097A:4E FE 09  213 ISCTRL: LSR IFLAG      ;alle Controls
097D:4E FF 09  214         LSR CAPSFLG    ;löschen die Flags
0980:          215 *
0980:C9 84     216 TSTDEL: CMP #$84       ;Ctrl-D: DELETE?
0982:D0 1F     217         BNE TSTBS
0984:AC 9E 0C  218         LDY CMDIDX     ;beide Strings
0987:CC 9F 0C  219         CPY CMDEND     ;gleich lang?
098A:F0 DF     220         BEQ NXTCHAR    ;=> kein DELETE
098C:B9 01 02  221 DEL1:   LDA CMDSTR+1,Y ;sonst wird das
098F:99 00 02  222         STA CMDSTR,Y   ;Backup um ein
0992:CC 9F 0C  223         CPY CMDEND     ;Zeichen verkürzt
0995:F0 03     224         BEQ DEL2       ;danach RETYPE
0997:C8        225         INY
0998:D0 F2     226         BNE DEL1
099A:CE 9F 0C  227 DEL2:   DEC CMDEND     ;Gesamtlänge minus 1
099D:20 1C 09  228         JSR RETYPE
09A0:4C 6B 09  229         JMP NXTCHAR
09A3:          230 *
09A3:C9 88     231 TSTBS:  CMP #$88       ;BACKSPACE?
09A5:D0 0E     232         BNE TSTINS
09A7:AE 9E 0C  233         LDX CMDIDX
09AA:F0 BF     234         BEQ NXTCHAR    ;CMDSTR = ""
09AC:20 ED FD  235         JSR COUT       ;Cursor nach links
09AF:CE 9E 0C  236         DEC CMDIDX     ;Index nach links
09B2:4C 6B 09  237         JMP NXTCHAR
09B5:          238 *
09B5:C9 89     239 TSTINS: CMP #$89       ;Ctrl-I: INSERT?
09B7:D0 0D     240         BNE TSTCR
09B9:AE 9F 0C  241         LDX CMDEND     ;beide Strings
09BC:EC 9E 0C  242         CPX CMDIDX     ;gleich lang?
09BF:F0 AA     243         BEQ NXTCHAR    ;=> kein INSERT
09C1:8D FE 09  244         STA IFLAG      ;sonst IFLAG setzen
09C4:D0 A5     245         BNE NXTCHAR    ;"always"
09C6:          246 *
09C6:C9 8D     247 TSTCR:  CMP #$8D       ;<CR>?
09C8:D0 06     248         BNE TSTCU
09CA:20 9C 0B  249         JSR CLEOP
09CD:4C 51 0A  250         JMP GOTLINE    ;*** ZEILE BEENDET
09D0:          251 *
09D0:C9 95     252 TSTCU:  CMP #$95       ;RECHTSPFEIL?
09D2:D0 0D     253         BNE TSTCX
09D4:AE 9E 0C  254         LDX CMDIDX     ;Backup > CMDSTR?
09D7:EC 9F 0C  255         CPX CMDEND
09DA:F0 8F     256         BEQ NXTCHAR    ;nein
09DC:BD 00 02  257         LDA CMDSTR,X   ;sonst ein Zeichen
09DF:D0 32     258         BNE GOTCHAR    ;Zeichen vom Backup
09E1:          259 *
09E1:C9 98     260 TSTCX:  CMP #$98       ;Ctrl-X?
09E3:D0 06     261         BNE TSTESC
09E5:20 07 09  262         JSR NEWCMD     ;alles neu
09E8:4C 6B 09  263         JMP NXTCHAR    ;macht der Mai...
09EB:          264 *
09EB:C9 9B     265 TSTESC: CMP #$9B       ;<ESC>?
09ED:D0 09     266         BNE BADCHAR
09EF:4D FF 09  267         EOR CAPSFLG    ;UC <=> LC
09F2:8D FF 09  268         STA CAPSFLG
09F5:4C 6B 09  269         JMP NXTCHAR
09F8:          270 *
09F8:          271 *
09F8:20 3A FF  272 BADCHAR: JSR BELL      ;Piep!
09FB:4C 6B 09  273         JMP NXTCHAR
09FE:          274 *
09FE:          275 *
09FE:          276 IFLAG:  DS  1          ;B7 set: INSERT
09FF:          277 CAPSFLG: DS 1          ;B7 set: UC => LC
0A00:          278 *
0A00:AE 9E 0C  279 ISCHAR: LDX CMDIDX     ;nächste freie
0A03:EC 9D 0C  280         CPX MAXCMD     ;Stelle in CMDSTR
0A06:F0 EE     281         BEQ BADCHAR    ;CMDSTR "voll"
0A08:2C FF 09  282         BIT CAPSFLG    ;LC-Übersetzung?
0A0B:10 06     283         BPL GOTCHAR    ;nein
0A0D:C9 C1     284         CMP #'A'
0A0F:90 02     285         BCC GOTCHAR
0A11:09 20     286         ORA #$20       ;Shift nach LC
0A13:4E FF 09  287 GOTCHAR: LSR CAPSFLG   ;immer zurück
0A16:2C FE 09  288         BIT IFLAG
0A19:10 1A     289         BPL APPEND     ;kein INSERT
0A1B:AC 9F 0C  290         LDY CMDEND
0A1E:C8        291         INY            ;Backup nach
0A1F:CC 9D 0C  292         CPY MAXCMD     ;INSERT zu lang?
0A22:F0 D2     293         BEQ BADCHAR
0A24:48        294         PHA            ;neues Zeichen
0A25:88        295 INS1:   DEY
0A26:B9 00 02  296         LDA CMDSTR,Y   ;Backup-Inhalt
0A29:99 01 02  297         STA CMDSTR+1,Y ;um 1 nach hinten
0A2C:CC 9E 0C  298         CPY CMDIDX     ;Originalstring-Ende?
0A2F:D0 F4     299         BNE INS1
0A31:EE 9F 0C  300         INC CMDEND     ;Backup-Länge+1
0A34:68        301         PLA
0A35:          302 *
0A35:9D 00 02  303 APPEND: STA CMDSTR,X ;+ neues Zeichen
0A38:EE 9E 0C  304         INC CMDIDX
0A3B:EC 9F 0C  305         CPX CMDEND     ;Backup länger?
0A3E:90 03     306         BCC APP2
0A40:EE 9F 0C  307         INC CMDEND     ;Nein, auch verlängern
0A43:20 E6 0A  308 APP2:   JSR PRCHAR     ;neues Zeichen drucken
0A46:2C FE 09  309         BIT IFLAG      ;bei INSERT
0A49:10 03     310         BPL APP3       ;wird auch Backup
0A4B:20 1C 09  311         JSR RETYPE     ;neu gedruckt
0A4E:4C 6C 09  312 APP3:   JMP NXTCHAR
0A51:          313 *
0A51:          316 **********************
0A51:          317 *---GOTLINE/NXTCMD---*
0A51:          318 **********************
0A51:          319 *
0A51:A9 00     320 GOTLINE: LDA #00
0A53:AE 9E 0C  321         LDX CMDIDX
0A56:9D 00 02  322         STA CMDSTR,X   ;<00>: Letztes Zeichen
0A59:          323 *
0A59:8E 9F 0C  324         STX CMDEND     ;für REPRINT
0A5C:8D 9E 0C  325         STA CMDIDX     ;CMDIDX = 00
0A5F:          326 *
0A5F:AD 00 C0  327 NXTCMD: LDA KEYBOARD ;Ctrl-C (STOP)?
0A62:10 07     328         BPL NXT0
0A64:8D 10 C0  329         STA KBSTROBE
0A67:C9 83     330         CMP #$83
0A69:F0 0E     331         BEQ NXTEND     ;JMP zu GETLINE
0A6B:20 4F 0B  332 NXT0:   JSR SETV22
0A6E:AE 9E 0C  333         LDX CMDIDX     ;nächster Command
0A71:EE 9E 0C  334         INC CMDIDX
0A74:BD 00 02  335         LDA CMDSTR,X
0A77:D0 03     336         BNE NXT1
0A79:4C 44 09  337 NXTEND: JMP GETLINE    ;CMDSTR zuende
0A7C:20 DF 0A  338 NXT1:   JSR SETUC      ;Shift nach UC
0A7F:C9 A1     339         CMP #STOP      ;leerer Command?
0A81:F0 DC     340         BEQ NXTCMD
0A83:C9 A0     341         CMP #$A0       ;Space?
0A85:F0 D8     342         BEQ NXTCMD
0A87:          343 *
0A87:A0 00     344         LDY #00
0A89:D9 5D 11  345 GETCMD: CMP CMDS,Y     ;Vergleich des
0A8C:F0 0A     346         BEQ GOTCMD     ;Zeichens mit CMDS
0A8E:C8        347         INY
0A8F:C8        348         INY
0A90:C8        349         INY
0A91:C0 0F     350         CPY #LCMD      ;Tabellenende?
0A93:90 F4     351         BCC GETCMD
0A95:4C AA 0A  352         JMP CMDERR     ;"Command Error"
0A98:          353 *
0A98:B9 5E 11  354 GOTCMD: LDA CMDS+1,Y ;Startadresse low
0A9B:8D A5 0A  355         STA DOCMD+1
0A9E:B9 5F 11  356         LDA CMDS+2,Y   ;hi
0AA1:8D A6 0A  357         STA DOCMD+2
0AA4:20 00 00  358 DOCMD:  JSR $0000      ;Modifiziert!
0AA7:4C 5F 0A  359         JMP NXTCMD
```

```
ØAAA:           360 *
ØAAA:           361 *
ØAAA:AØ ØØ      362 CMDERR: LDY #ERRCMD-ERRMSGS ;Index
ØAAC:8E 9E ØC   363 DSPERR: STX CMDIDX   ;Fehler ist hier..
ØAAF:2Ø CB ØA   364         JSR PRTERR
ØAB2:2Ø 47 ØB   365         JSR SETV221  ;VTAB 22, HTAB 1
ØAB5:A2 ØØ      366         LDX #ØØ
ØAB7:EC 9E ØC   367 DSERR1: CPX CMDIDX   ;Reprint von CMDSTR:
ØABA:FØ Ø9      368         BEQ DSERR2   ;der Cursor bleibt
ØABC:BD ØØ Ø2   369         LDA CMDSTR,X ;auf dem falschen
ØABF:2Ø E6 ØA   37Ø         JSR PRCHAR   ;Command stehen
ØAC2:E8         371         INX
ØAC3:DØ F2      372         BNE DSERR1
ØAC5:A2 FF      373 DSERR2: LDX #$FF     ;Stack zurück
ØAC7:9A         374         TXS
ØAC8:4C 47 Ø9   375         JMP GETLNZ   ;GETLINE ohne VTAB 22
ØACB:           376 *
ØACB:2Ø 53 ØB   377 PRTERR: JSR SETV24   ;VTAB 24
ØACE:A9 87      378         LDA #$87     ;BELL
ØADØ:2Ø ED FD   379         JSR COUT
ØAD3:B9 DØ ØC   38Ø PRERR1: LDA ERRMSGS,Y
ØAD6:FØ Ø6      381         BEQ PRERR2   ;<ØØ> = Textende
ØAD8:2Ø E6 ØA   382         JSR PRCHAR
ØADB:C8         383         INY
ØADC:DØ F5      384         BNE PRERR1
ØADE:6Ø         385 PRERR2: RTS
ØADF:           386 *
ØADF:           387 *
ØADF:           388 *
ØADF:           389 ********************
ØADF:           39Ø *--PRINT-ROUTINEN--*
ØADF:           391 ********************
ØADF:           392 *
ØADF:           393 *
ØADF:C9 E1      394 SETUC: CMP #'a'      ;Shift nach UC
ØAE1:9Ø Ø2      395         BCC UCSET
ØAE3:29 DF      396         AND #$DF
ØAE5:6Ø         397 UCSET: RTS
ØAE6:           398 *
ØAE6:2C 98 ØC   399 PRCHAR: BIT SHIFTUP  ;nur Uppercase?
ØAE9:1Ø ØF      4ØØ         BPL PRC1     ;nein, hat LC
ØAEB:Ø9 8Ø      4Ø1         ORA #$8Ø     ;SHIFTUP: immer 4ØZ
ØAED:C9 E1      4Ø2         CMP #'a'     ;LC-Zeichen werden
ØAEF:9Ø Ø9      4Ø3         BCC PRC1     ;invers gedruckt
ØAF1:48         4Ø4         PHA
ØAF2:A9 3F      4Ø5         LDA #$3F     ;INVFLAG: INVERSE
ØAF4:85 32      4Ø6         STA INVFLAG
ØAF6:68         4Ø7         PLA
ØAF7:2Ø DF ØA   4Ø8         JSR SETUC
ØAFA:2C 9A ØC   4Ø9 PRC1:   BIT IS4Ø     ;4Ø Zeichen?
ØAFD:1Ø Ø2      41Ø         BPL PRC2
ØAFF:Ø9 8Ø      411         ORA #$8Ø
ØBØ1:2Ø ED FD   412 PRC2:   JSR COUT
ØBØ4:A9 FF      413         LDA #$FF     ;INVFLAG immer
ØBØ6:85 32      414         STA INVFLAG  ;zurück auf NORMAL
ØBØ8:6Ø         415         RTS
ØBØ9:           416 *
ØBØ9:2Ø 26 ØB   417 PRADDR: JSR PRHEX    ;Acc => 1 Hexzahl
ØBØC:2Ø 1C ØB   418         JSR PRTY     ;Y => 2 Hexzahlen
ØBØF:A9 AD      419         LDA #'-'
ØB11:2Ø ED FD   42Ø         JSR COUT
ØB14:A9 AØ      421         LDA #$AØ     ;und ein Space
ØB16:4C ED FD   422         JMP COUT
ØB19:           423 *
ØB19:2Ø 1D ØB   424 PRTAY: JSR PRBYTE    ;Acc => 2 Hexzahlen
ØB1C:98         425 PRTY:   TYA
ØB1D:48         426 PRBYTE: PHA
ØB1E:4A         427         LSR A        ;zuerst wird das
ØB1F:4A         428         LSR A        ;höhere Nibble, da-
ØB2Ø:4A         429         LSR A        ;nach das niedrigere
ØB21:4A         43Ø         LSR A        ;Nibble gedruckt
ØB22:2Ø 26 ØB   431         JSR PRHEX
ØB25:68         432         PLA
ØB26:29 ØF      433 PRHEX:  AND #$ØF     ;Acc => 1 Hexzahl
ØB28:Ø9 BØ      434         ORA #$BØ
ØB2A:C9 BA      435         CMP #$BA     ;größer "9"?
ØB2C:9Ø Ø2      436         BCC PRHX1
ØB2E:69 Ø6      437         ADC #Ø6      ;+ 7 (Carry gesetzt)
ØB3Ø:4C ED FD   438 PRHX1:  JMP COUT
ØB33:           439 *
ØB33:AØ Ø8      44Ø PRBITS: LDY #Ø8       ;Acc => "1.1.1.1."
ØB35:ØA         441 PRB1:   ASL A        ;also "richtig", d.h.
ØB36:48         442         PHA          ;MSB zuerst.
ØB37:9Ø Ø4      443         BCC PRB2     ;Bit nicht gesetzt
ØB39:A9 B1      444         LDA #'1'
ØB3B:BØ Ø2      445         BCS PRB3
ØB3D:A9 AE      446 PRB2:   LDA #'.'      ;Bit clear: "."
ØB3F:2Ø ED FD   447 PRB3:   JSR COUT
ØB42:68         448         PLA
ØB43:88         449         DEY
```

```
ØB44:DØ EF      45Ø         BNE PRB1
ØB46:6Ø         451         RTS
ØB47:           452 *
ØB47:           453 ****************************
ØB47:           454 *---CURSORPOSITIONIERUNG ---*
ØB47:           455 ****************************
ØB47:           456 *
ØB47:2Ø 4F ØB   457 SETV221: JSR SETV22
ØB4A:A9 AØ      458         LDA #$AØ      ;VTAB 22, HTAB 1
ØB4C:4C ED FD   459         JMP COUT
ØB4F:           46Ø *
ØB4F:A9 16      461 SETV22: LDA #22
ØB51:DØ Ø2      462         BNE SETVTB
ØB53:A9 18      463 SETV24: LDA #24       ;VTAB 24
ØB55:           464 *
ØB55:38         465 SETVTB: SEC
ØB56:E9 Ø2      466         SBC #Ø2       ;Faulheit:
ØB58:2Ø 7B ØB   467         JSR SETCV     ;VTAB 1 => CV=ØØ
ØB5B:A9 8D      468 DOCR:   LDA #$8D
ØB5D:EE 96 ØC   469         INC EDCV      ;internes VTAB
ØB6Ø:4C ED FD   47Ø         JMP COUT      ;<CR> macht VCALC
ØB63:           471 *
ØB63:A9 Ø5      472 CLRDSP: LDA #5        ;VTAB 5
ØB65:AA         473         TAX
ØB66:2Ø 55 ØB   474         JSR SETVTB
ØB69:2Ø A9 ØB   475 CLRD1:  JSR CLRLIN    ;löscht Display
ØB6C:EE 96 ØC   476         INC EDCV      ;von VTAB 5 bis 2Ø
ØB6F:AD 96 ØC   477 CLRDS2: LDA EDCV      ;internes CV
ØB72:C9 14      478         CMP #2Ø       ;VTAB 21 erreicht?
ØB74:9Ø F3      479         BCC CLRD1
ØB76:A9 Ø5      48Ø         LDA #5        ;ok - zurück
ØB78:4C 55 ØB   481         JMP SETVTB    ;auf VTAB 5
ØB7B:           482 *
ØB7B:           483 *
ØB7B:           484 *********************************
ØB7B:           485 *---HOME-CLEOP-CLEOL-SETCV/CH---*
ØB7B:           486 *********************************
ØB7B:           487 *
ØØ8B:           488 CLEOP8Ø: EQU $8B      ;Ctrl-K
ØØ9D:           489 CLEOL8Ø: EQU $9D      ;<GS>
FC42:           49Ø MONCLP: EQU $FC42     ;Monitor: CLEOP
FC9C:           491 MONCLL: EQU $FC9C     ;Monitor: CLEOL
ØB7B:           492 *
ØB7B:8D 96 ØC   493 SETCV: STA EDCV       ;interner Wert
ØB7E:85 25      494         STA CV        ;Monitor
ØB8Ø:2C 99 ØC   495         BIT IIEC      ;IIc/e 8ØCOL?
ØB83:1Ø Ø3      496         BPL CVSET     ;nein
ØB85:8D FB Ø5   497         STA $5FB      ;sonst direkt
ØB88:6Ø         498 CVSET: RTS
ØB89:           499 *
ØB89:8D 95 ØC   5ØØ SETCH: STA EDCH       ;interner Wert
ØB8C:85 24      5Ø1         STA CH        ;Monitor
ØB8E:2C 99 ØC   5Ø2         BIT IIEC      ;IIc/e 8ØCOL?
ØB91:1Ø Ø3      5Ø3         BPL CHSET     ;nein
ØB93:8D FB Ø5   5Ø4         STA $5FB      ;sonst direkt
ØB96:6Ø         5Ø5 CHSET: RTS
ØB97:           5Ø6 *
ØB97:A9 Ø1      5Ø7 HOME:   LDA #Ø1       ;VTAB 1
ØB99:2Ø 55 ØB   5Ø8         JSR SETVTB
ØB9C:           5Ø9 *
ØB9C:2C 9A ØC   51Ø CLEOP: BIT IS4Ø       ;Clear End_of_Page
ØB9F:3Ø Ø5      511         BMI CLP4Ø     ;4Ø Zeichen
ØBA1:A9 8B      512         LDA #CLEOP8Ø  ;Ctrl für 8Ø Col
ØBA3:4C ED FD   513         JMP COUT
ØBA6:4C 42 FC   514 CLP4Ø: JMP MONCLP     ;Monitor-Routine
ØBA9:           515 *
ØBA9:EE 96 ØC   516 CLRLIN: INC EDCV      ;Clear Line
ØBAC:AD 96 ØC   517         LDA EDCV      ;erzwingt VCALC
ØBAF:2Ø 55 ØB   518         JSR SETVTB    ;auf Ziel-Line
ØBB2:           519 *
ØBB2:2C 9B ØC   52Ø CLEOL: BIT PRINTER    ;Printer an?
ØBB5:3Ø ØD      521         BMI CLEOLDN   ;=> kein CLEOL!
ØBB7:2C 9A ØC   522         BIT IS4Ø      ;Clear End_of_Line
ØBBA:3Ø Ø5      523         BMI CLL4Ø     ;Monitor-Routine
ØBBC:A9 9D      524         LDA #CLEOL8Ø  ;Ctrl für 8Ø Col
ØBBE:4C ED FD   525         JMP COUT
ØBC1:2Ø 9C FC   526 CLL4Ø: JSR MONCLL
ØBC4:6Ø         527 CLEOLDN: RTS
ØBC5:           528 *
ØBC5:           529 ************************
ØBC5:           53Ø *--ARGUMENT-BERECHNUNG--*
ØBC5:           531 ************************
ØBC5:           532 *
ØBC5:A9 Ø2      533 GETPRM: LDA #MAXBUF   ;holt <xx>,<yy> als
ØBC7:AØ ØØ      534         LDY #ØØ       ;Start/Endadresse
ØBC9:8D A3 ØC   535         STA EADDR+1   ;aus CMDSTR
ØBCC:8C A2 ØC   536         STY EADDR     ;Default-Endadresse
ØBCF:A9 8Ø      537         LDA #$8Ø
ØBD1:8D A4 ØC   538         STA GOTPRM1   ;Annahme: Start
ØBD4:8D A5 ØC   539         STA GOTPRM2   ;und Ende explizit
```

Weselerstr. 61 4400 Münster

TLK OHG



. . .

Sie brauchen Apple Kick

weil
- es zu jeder Zeit auf Tastendruck zur Verfügung steht, ohne Ihr momentanes Programm zu stören
- es keinen Platz wegnimmt, weder auf Ihrem Schreibtisch, noch in Ihrem Programmspeicher und sich auf dem Bildschirm auf Windows beschränkt
- auf einmal Ihr Textverarbeitungsprogramm rechnen und Ihr Kalkulationsprogramm Texte erfassen und übernehmen kann
- erst durch die Kombination von Rechner, Terminkalender, Notizbuch, ASCII Tabelle, Funktionstasten, Drucker-Buffer, Screendump .... Ihre Anwendersoftware benutzerfreundlicher wird.

Der **Apple Kick** läuft auf einem Apple //e //c mit 128k Ram, CP/M 2.0, 2.2B, 2.23, 2.24
In Vorbereitung: Basis 108 mit CP/M 2 oder CP/M 3.0 sowie Apple //e mit CP/M 3.0

Sie erhalten:

Demokick: DM 10 (wird bei Kauf verrechnet)

Apple Kick: DM 96
mit: Taschenrechner mit 4 Grundrechenarten, Hex, Bin, Oct, Dez, ASCII-Tabelle, 128 frei belegbaren Funktionstasten (abspeicherbar), Macros, Drucker-Buffer und Funktionen, Screen-Dump. Ausführliches Handbuch.

Apple Kick+: DM 245
mit: dito, Terminkalender, Notizbuch mit fullscreen Editor, Filer (dir, type, era, rename, Hexdump, Laufwerkskonfig. ändern), Telefonverzeichnis, Schreibmaschinenfunktionen, erweitertem speicherresistenten CCP sowie DIR-Command mit free, search path.

Ausführliche Infos in der Mailbox M.A.U.S. 0251/522790 (300 Baud 8n1) oder Info-Blatt anfordern!

Programmierer, die die Software auf andere Rechner anpassen wollen, werden unterstützt!
Nachträgliche Änderungen in der Ausstattung vorbehalten!

Weitere Produkte: sehr komfortable Mailbox mit Protokollübertragung (s.o.), Window Routinen, Eigenständiger Editor, einzelne Module von **Apple Kick** auf Anfrage.

0251/522784
0251/522790 (300 Baud 8n1)

Ausgabe und Eingabe mit TYPETERM®

im Slot Ihres

APPLE II/IIe

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

Die neue CE-550 mit TYPETERM DM 1.398,–

TYPETERM-Interface DM 479,–
für alle BROTHER-Typenradschreibmaschinen ab CE-51 bis EM-250

CE-68 mit TYPETERM	DM 1947,–
EM-80 mit TYPETERM	DM 1887,–
TYPETERM-Kit für CE-50	DM 468,–
Cable Kit A (erforderlich ab EM-80)	DM 103,–

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

TYPETERM – ein Produkt von

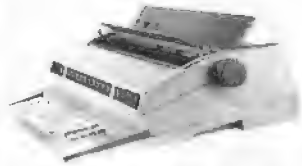
Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 0 51 39 - 8 73 93

Ausgabe mit TYPETERM® JUNIOR

im Slot Ihres

APPLE II/IIe

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



CE-550 brother

TYPETERM JUNIOR mit AX-10 – unser besonders günstiges Gespann, ebenfalls steckfertig. Mit TYPETERM JUNIOR kann die AX-10 mehr. Sie wird zum vollwertigen Typenraddrucker für Ihren Apple:
- 3 verschiedene Schriftstärken
- Automatisches Unterstreichen
- 2 Zeichensätze z.B. deutsch u. ASCII
- 2 Zeichenabstände
- 2k ROM auf der Karte für Ausgabe unter DOS, PRODOS, CP/M u. PASCAL.

TYPETERM JUNIOR – ein Produkt von

Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 0 51 39 - 8 73 93

Der problemlose Aufstieg in die Assembler-Sprache für alle Apple-Anwender.

Jeder, der die Grenzen der höheren Programmiersprachen Basic und Pascal überschreiten will, findet in diesem Buch genau die richtigen Unterlagen für seine weitere Arbeit in der 6502-Assembler-Sprache.
Dem Apple-Anwender bietet sich hier sowohl eine umfassende Einführung in das für ihn neue Gebiet der Assembler-Sprache, als auch eine äußerst wertvolle Sammlung von aussagekräftigen und nützlichen Programmen. Sie bilden den Kern des Buches. Diese Programme lassen sich als fertige Utilities ebenso anwenden, wie als Bausteine in Verbindung mit Basic- oder Pascalprogrammen. Behilflich sind diese Routinen für zahlreiche Problemlösungen, denn ihre Konzeptionen lassen sich problemlos aneinanderketten.
Das hier zusammengetragene und wohlaufbereitete Know-how wird den Apple-Anwender in die Lage versetzen, seinen Computer ganz wesentlich intensiver kennenzulernen und seine Leistungsfähigkeit besser und effektiver auszunutzen.

Assembler-Programme zum Apple-II

Problemloser Aufstieg von Basic in die Assemblersprache. Von Erich **Esders.**

204 Seiten, 90 Abbildungen Lwstr-kart. DM 48.–
ISBN 3-7723-8141-3

Franzis'

der große Fachverlag für angewandte Elektronik und Informatik
Franzis-Verlag, Postfach 37 01 20, München

(145)

UCSD-Pascal

für Apple II + //e,c-Computer und Kompatible

Wir können Ihnen folgende Dienstprogramme anbieten:

Mammut-Editor: wird höchsten Ansprüchen gerecht und eignet sich als Programm- und Texteditor. Ein File mit deutschen Compilermeldungen wird mitgeliefert. **VP sFr. 250,–**

Mammut-Bildschirm-Treiber: Zeichenausgabe an den Bildschirm mit der 4.5-fachen Geschwindigkeit. Funktioniert mit allen Ihren Programmen, ohne daß Sie diese verändern müssen. Ausdruck des Bildschirms mitten in einem Programm möglich. Es kann auch mit Windows gearbeitet werden. Einschließlich Quelltext und Anleitung. *Nur für Apple IIe mit Textcard.* **VP sFr. 150,–**

Mammut-Format: leistungsfähiges Programm zur Ausgabe von Textdateien an beliebige Drucker. Zwei verschiedene Drucker können gleichzeitig bedient werden (z. B. Nadel- und Typenraddrucker). Unterstützung der druckerspezifischen Eigenschaften bei einfachster Handhabung. Schönschrift und frei definierbarer Zeichensatz, auch für Matrixdrucker ohne NLQ-Modus. **VP sFr. 250,–**

Mit diesen Programmen kommunizieren Sie in deutscher oder englischer Sprache wahlweise. Außerdem offerieren wir eine Gewährleistung von 6 Monaten auf *Software* und Datenträger. Programme mit deutschen Anleitungen oder kostenlose Informationen erhalten Sie bei

MAMMUT Soft

Condrau und Brunner · Postfach 373 · CH-8600 Dübendorf

Eine Demo-Diskette ist gegen eine Bearbeitungsgebühr von sFr. 20,– erhältlich.

```
ØBD7:AE 9E ØC  540         LDX CMDIDX    ;Test&Berechnung
ØBDA:2Ø 36 ØC  541         JSR EVALHEX   ;erstes Argument
ØBDD:BØ 37     542         BCS SETPRM1   ;kein Argument
ØBDF:2Ø 2D ØC  543         JSR TSTARG    ;Arg > MAXBUF?
ØBE2:BØ 41     544         BCS PRMERR    ;"Parameter Error"
ØBE4:8D A1 ØC  545         STA SADDR+1
ØBE7:8C AØ ØC  546         STY SADDR
ØBEA:BD ØØ Ø2  547         LDA CMDSTR,X  ;folgt Komma?
ØBED:C9 AC     548         CMP #','
ØBEF:DØ 3Ø     549         BNE SETPRM2   ;nein, keine EADDR
ØBF1:E8        55Ø         INX           ;nach "," MUß das
ØBF2:2Ø 36 ØC  551         JSR EVALHEX   ;zweite Argument
ØBF5:BØ 2E     552         BCS PRMERR    ;folgen, sonst ERR
ØBF7:2Ø 2D ØC  553         JSR TSTARG    ;Arg > MAXBUF?
ØBFA:BØ 29     554         BCS PRMERR
ØBFC:8D A3 ØC  555         STA EADDR+1
ØBFF:48        556         PHA
ØC00:38        557         SEC           ;Test, ob SADDR
ØC01:98        558         TYA           ;größer EADDR
ØC02:ED AØ ØC  559         SBC SADDR
ØC05:68        56Ø         PLA
ØC06:ED A1 ØC  561         SBC SADDR+1
ØC09:9Ø 1A     562         BCC PRMERR    ;ja, ERR
ØC0B:C8        563         INY           ;EADDR ist Ober-
ØC0C:8C A2 ØC  564         STY EADDR     ;grenze, d.h.
ØC0F:DØ 13     565         BNE PRMSET    ;1 mehr als
ØC11:EE A3 ØC  566         INC EADDR+1   ;angegeben
ØC14:DØ ØE     567         BNE PRMSET    ;ok- "always"
ØC16:          568 *
ØC16:4E A4 ØC  569 SETPRM1: LSR GOTPRM1 ;Flag zurück
ØC19:A9 ØØ     57Ø         LDA #ØØ
ØC1B:8D AØ ØC  571         STA SADDR     ;Default-SADDR
ØC1E:8D A1 ØC  572         STA SADDR+1   ;ist ØØ ØØ
ØC21:4E A5 ØC  573 SETPRM2: LSR GOTPRM2 ;EADDR ist
ØC24:6Ø        574 PRMSET: RTS           ;bereits gesetzt
ØC25:          575 *
ØC25:4C AA ØA  576 PRMERR: JMP CMDERR
ØC28:          577 *
ØC28:2Ø 36 ØC  58Ø EVALARG: JSR EVALHEX ;Argument aus CMDSTR
ØC2B:BØ Ø8     581         BCS ARGEV     ;illegal
ØC2D:AD A7 ØC  582 TSTARG: LDA EVREG+1
ØC3Ø:AC A6 ØC  583         LDY EVREG
ØC33:C9 Ø2     584         CMP #MAXBUF   ;RTS mit C set,
ØC35:6Ø        585 ARGEV: RTS           ;wenn > MAXBUF
ØC36:          586 *
ØC36:A9 ØØ     587 EVALHEX: LDA #ØØ
ØC38:8D A7 ØC  588         STA EVREG+1
ØC3B:BD ØØ Ø2  589         LDA CMDSTR,X
ØC3E:C9 A5     59Ø         CMP #'%'      ;Variable?
ØC4Ø:DØ Ø4     591         BNE EVALlST
ØC42:E8        592         INX
ØC43:4C 58 11  593         JMP GETVAR    ;Variable einsetzen
ØC46:          594 *
ØC46:2Ø 7Ø ØC  595 EVALlST: JSR SETHEX  ;Umwandlung ASCII=>Hex
ØC49:BØ 24     596         BCS EVALQT    ;keine Hexzahl
ØC4B:8D A6 ØC  597         STA EVREG
ØC4E:E8        598 EVLOOP: INX           ;dieses Zeichen
ØC4F:8E 9E ØC  599         STX CMDIDX    ;ist erledigt
ØC52:BD ØØ Ø2  6ØØ         LDA CMDSTR,X
ØC55:2Ø 7Ø ØC  6Ø1         JSR SETHEX
ØC58:BØ 14     6Ø2         BCS EVALEND   ;Hex-Argument zuende
ØC5A:AØ Ø4     6Ø3         LDY #4
ØC5C:ØE A6 ØC  6Ø4 MULT16: ASL EVREG
ØC5F:2E A7 ØC  6Ø5         ROL EVREG+1   ;alte Zahl * 16
ØC62:88        6Ø6         DEY
ØC63:DØ F7     6Ø7         BNE MULT16
ØC65:ØD A6 ØC  6Ø8         ORA EVREG
ØC68:8D A6 ØC  6Ø9         STA EVREG     ;und neue Zahl dazu
ØC6B:4C 4E ØC  61Ø         JMP EVLOOP
ØC6E:18        611 EVALEND: CLC          ;Argument erhalten
ØC6F:6Ø        612 EVALQT: RTS           ;C set wenn illegal
ØC7Ø:          613 *
ØC7Ø:          614 *
ØC7Ø:2Ø 7F ØC  615 SETHEX: JSR TSTHEX   ;Test auf Hexzahl
ØC73:BØ Ø9     616         BCS HEXSET
ØC75:E9 AF     617         SBC #$AF      ;minus $BØ
ØC77:C9 ØA     618         CMP #$ØA
ØC79:9Ø Ø3     619         BCC HEXSET
ØC7B:E9 Ø7     62Ø         SBC #$Ø7
ØC7D:18        621         CLC
ØC7E:6Ø        622 HEXSET: RTS
ØC7F:          623 *
ØC7F:2Ø DF ØA  624 TSTHEX: JSR SETUC    ;Shift nach UC
ØC82:C9 BØ     625         CMP #'Ø'
ØC84:9Ø ØC     626         BCC NOTHEX
ØC86:C9 BA     627         CMP #$BA
ØC88:9Ø Ø9     628         BCC ISHEX
ØC8A:C9 C1     629         CMP #'A'
ØC8C:9Ø Ø4     63Ø         BCC NOTHEX
ØC8E:C9 C7     631         CMP #'G'
```

```
ØC9Ø:9Ø Ø1      632         BCC ISHEX
ØC92:38        633 NOTHEX: SEC           ;keine Hexzahl
ØC93:6Ø        634 ISHEX: RTS            ;C clear wenn ok
ØC94:          635 *
ØC94:          636 *
ØC94:          637 *
ØC94:          638 ***************
ØC94:          639 *---GLOBALS---*
ØC94:          64Ø ***************
ØC94:          641 *
ØC94:          642 GLOBALS: EQU *
ØC94:          643 *
ØC94:          644 TEMP:  DS  1      ;Scratch
ØC95:          645 EDCH:  DS  1      ;internes CH (Cursor)
ØC96:          646 EDCV:  DS  1      ;internes CV (Cursor)
ØC97:          647 *
ØC97:          648 HMAX:  DS  1      ;4Ø: 4ØZ, 8Ø:8Ø Zeichen
ØC98:          649 SHIFTUP: DS 1     ;Ø: hat LC, sonst $FF
ØC99:          65Ø IIEC:  DS  1      ;$FF, wenn IIe/c 8Ø Col
ØC9A:          651 IS4Ø:  DS  1      ;$FF: 4Ø Zeichen
ØC9B:          652 PRINTER: DS 1     ;$FF: Printer-Ouput
ØC9C:          653 NODSP: DS  1      ;$8Ø: kein DUMP-Display
ØC9D:          654 *
ØC9D:          655 MAXCMD: DS 1      ;max. Länge von CMDSTR
ØC9E:          656 CMDIDX: DS 1      ;mom. Index in CMDSTR
ØC9F:          657 CMDEND: DS 1      ;Gesamtlänge CMDSTR
ØCAØ:          658 *
ØCAØ:          659 SADDR: DS  2      ;Startadresse (Ø..MAXBUF)
ØCA2:          66Ø EADDR: DS  2      ;Endadresse (Ø..MAXBUF)
ØCA4:          661 GOTPRM1: DS 1     ;$8Ø, wenn SADDR explizit
ØCA5:          662 GOTPRM2: DS 1     ;$8Ø, wenn EADDR explizit
ØCA6:          663 EVREG: DS  2      ;momentaner Argument-Wert
ØCA8:          664 *
ØCA8:          665 NXTMODE: DS 1     ;nächster DUMP-Modus
ØCA9:          666 AMODE: DS  1      ;ASCII-Dump
ØCAA:          667 HMODE: DS  1      ;Hex/ASCII-Dump
ØCAB:          668 CMODE: DS  1      ;COMPARE-Dump
ØCAC:          669 VMODE: DS  1      ;VBM-Dump
ØCAD:          67Ø *
ØCAD:          671 LBYTES: DS 1      ;Temp für Bytes/Zeile
ØCAE:          672 BPLINE: DS 1      ;Bytes per Zeile, 8ØZ
ØCAF:          673 NBYTES: DS 1      ;gedruckte Bytes/Zeile
ØCBØ:          674 *
ØCBØ:          675 DSPBTM: DS 2      ;Untergrenze für Dump
ØCB2:          676 DSPSTART: DS 2    ;momentaner Dump-Start
ØCB4:          677 NXTADDR: DS 2     ;Zeilen-Startadresse
ØCB6:          678 DSPEND: DS 2      ;momentanes Dump-Ende
ØCB8:          679 DSPTOP: DS 2      ;obere Grenze für Dump
ØCBA:          68Ø *
ØCBA:          681 SUCCEED: DS 1     ;"+" oder "-" für JUMP
ØCBB:          682 *
ØCBB:          683 CPBASE: DS 2      ;CDUMP-Basis in BUFFER
ØCBD:          684 CPIDX: DS  2      ;COMPARE: HLBUF-Index
ØCBF:          685 *
ØCBF:          686 HLSET: DS  1      ;$8Ø: HLBUF gesetzt
ØCCØ:          687 HLNUM: DS  2      ;Anzahl Bytes in HLBUF
ØCC2:          688 *
ØCC2:          689 SETAD: DS  2      ;Startadresse von SET
ØCC4:          69Ø *
ØCC4:          691 FNADDR: DS 2      ;Startadresse von FIND
ØCC6:          692 LGLBL: EQU *      ;Variablen-Ende
ØCC6:          693 *
ØØØA:          694         DS  6Ø+GLOBALS-LGLBL ;Erweiterungen
ØCDØ:          695 *
ØCDØ:          696 ********************
ØCDØ:          697 *
ØCDØ:          698 *
ØCDØ:          699 ERRMSGS: EQU *
ØCDØ:C2 E1 E4  7ØØ ERRCMD: ASC 'Bad Command/Syntax/Parameter'
ØCEC:ØØ        7Ø1         DFB ØØ        ;Textende
ØCED:C9 EC EC  7Ø2 PATHERR: ASC 'Illegal Pathname/Volume'
ØD04:AF C6 E9  7Ø3         ASC '/File not found'
ØD13:ØØ        7Ø4         DFB ØØ
ØD14:D7 F2 E9  7Ø5 WPROT:  ASC 'Write protected'
ØD23:ØØ        7Ø6         DFB ØØ
ØD24:C9 AF CF  7Ø7 ERRWR:  ASC 'I/O ERROR'
ØD2D:ØØ        7Ø8         DFB ØØ
ØD2E:CF F5 F4  7Ø9 EXXBUF: ASC 'Out of Buffer Range'
ØD41:ØØ        71Ø         DFB ØØ
ØD42:CE EF AØ  711 NOBUF:  ASC 'No Hold Buffer set'
ØD54:ØØ        712         DFB ØØ
ØD55:CE EF AØ  713 PFXNONE: ASC 'No Prefix?'
ØD5F:ØØ        714         DFB ØØ
ØD6Ø:CA D5 CD  715 BADLBL: ASC 'JUMP Label not found/illegal'
ØD7C:ØØ        716         DFB ØØ
ØD7D:          717 *
ØD7D:          718 *
ØD7D:          719         CHN ED.DISKIO.TXT
```

ED.DISKIO.TXT

```
ØD7D:             1         PAGE
ØD7D:             2 ***********************
ØØØ2:             3 MAXBUF: EQU 2          ;$2ØØ Byte per Block
ØD7D:             4 ***********************
ØD7D:             5 *
BE3Ø:             6 VECTOUT: EQU $BE3Ø     ;I/O-Vecs BASIC.SYSTEM
BE34:             7 VDOSIO: EQU $BE34      ;Handler BASIC.SYSTEM
ØD7D:             8 *
ØD7D:A2 Ø3        9 DISCONN: LDX #Ø3       ;BASIC.SYSTEM ab:
ØD7F:BD 3Ø BE    1Ø DISCN1: LDA VECTOUT,X  ;CSW/KSW werden
ØD82:95 36       11         STA COUTV,X    ;mit den I/O-Vecs
ØD84:CA          12         DEX            ;direkt gesetzt, das
ØD85:1Ø F8       13         BPL DISCN1     ;BASIC.SYSTEM hängt
ØD87:6Ø          14         RTS            ;nicht mehr dazwischen
ØD88:            15 *
ØD88:A2 Ø3       16 DOSCONN: LDX #Ø3       ;BASIC.SYSTEM an:
ØD8A:B5 36       17 DOSCN1: LDA COUTV,X    ;CSW/KSW zurück
ØD8C:9D 3Ø BE    18         STA VECTOUT,X  ;in die I/Os
ØD8F:BD 34 BE    19         LDA VDOSIO,X   ;und der Handler
ØD92:95 36       2Ø         STA COUTV,X    ;des BASIC.SYSTEM
ØD94:CA          21         DEX            ;nach CSW/KSW
ØD95:1Ø F3       22         BPL DOSCN1
ØD97:6Ø          23         RTS
ØD98:            24 *
ØD98:4C D3 Ø3    25 EXIT:   JMP $3D3       ;DOS Kaltstart
ØD9B:            26 *
ØD9B:            27 ***********************
ØD9B:            28 *--READ_BLOCK/PREFIX--*
ØD9B:            29 ***********************
ØD9B:            3Ø *
ØD9B:A9 8Ø       31 READ:   LDA #$8Ø       ;die folgenden
ØD9D:8D A2 ØD    32         STA RDFLAG     ;Aktionen beziehen
ØDAØ:DØ Ø4       33         BNE WRITEZ     ;sich auf READ
ØDA2:            34 *
ØDA2:            35 RDFLAG: DS 1           ;B7 set, wenn READ
ØDA3:            36 *
ØDA3:            37 ************************
ØDA3:            38 *--WRITE_BLOCK/PREFIX--*
ØDA3:            39 ************************
ØDA3:            4Ø *
ØDA3:4E A2 ØD    41 WRITE: LSR RDFLAG      ;RDFLAG zurück
ØDA6:AE 9E ØC    42 WRITEZ: LDX CMDIDX
ØDA9:BD ØØ Ø2    43         LDA CMDSTR,X
ØDAC:2Ø DF ØA    44         JSR SETUC      ;Shift nach UC
ØDAF:C9 DØ       45         CMP #'P'       ;Path?
ØDB1:FØ Ø3       46         BEQ SETPATH
ØDB3:4C 9Ø ØE    47         JMP RWBLOCK
ØDB6:            48 *
ØDB6:2Ø 83 ØE    49 SETPATH: JSR GNCHAR    ;nächstes Zeichen
ØDB9:C9 AF       5Ø         CMP #'/'
ØDBB:FØ 5E       51         BEQ SPREFIX    ;Pathname folgt
ØDBD:38          52 SUNIT:  SEC
ØDBE:E9 BØ       53         SBC #'Ø'       ;muß dann SLOT sein:
ØDCØ:9Ø 1C       54         BCC UNITERR    ;kleiner 'Ø'?
ØDC2:FØ 1A       55         BEQ UNITERR    ;nur Slot 1..7
ØDC4:C9 Ø8       56         CMP #Ø8
ØDC6:BØ 16       57         BCS UNITERR
ØDC8:8D 66 ØE    58         STA PSLOT
ØDCB:2Ø 83 ØE    59         JSR GNCHAR
ØDCE:C9 AC       6Ø         CMP #','       ;danach ein Komma
ØDDØ:DØ ØC       61         BNE UNITERR
ØDD2:2Ø 83 ØE    62         JSR GNCHAR
ØDD5:38          63         SEC
ØDD6:E9 B1       64         SBC #'1'       ;und die Driveno
ØDD8:9Ø Ø4       65         BCC UNITERR
ØDDA:C9 Ø2       66         CMP #Ø2        ;=> auf Ø oder 1
ØDDC:9Ø Ø3       67         BCC SUNIT1
ØDDE:4C AA ØA    68 UNITERR: JMP CMDERR
ØDE1:E8          69 SUNIT1: INX
ØDE2:8E 9E ØC    7Ø         STX CMDIDX     ;Befehl 'abgehakt'
ØDE5:8D 65 ØE    71         STA PDRIVE
ØDE8:4A          72         LSR A          ;Driveno ins Carry
ØDE9:Ø8          73         PHP
ØDEA:AD 66 ØE    74         LDA PSLOT
ØDED:2Ø 67 ØE    75         JSR SHIFT5     ;ØØØØØxxx => xxxØØØØØ
ØDFØ:29 EØ       76         AND #$EØ       ;111Ø ØØØØ
ØDF2:28          77         PLP
ØDF3:6A          78         ROR A          ;Driveno ins MSB
ØDF4:AA          79 SETINFO: TAX           ;Unitnummer
ØDF5:AD 66 ØE    8Ø         LDA PSLOT
ØDF8:Ø9 BØ       81         ORA #$BØ       ;Slotnummer für PRINFO
ØDFA:A8          82         TAY
ØDFB:AD 65 ØE    83         LDA PDRIVE
ØDFE:18          84         CLC
ØDFF:69 B1       85         ADC #$B1       ;Driveno für PRINFO
ØE Ø1:2C A2 ØD   86         BIT RDFLAG
ØE Ø4:1Ø Ø9      87         BPL WRINFO     ;nur WRITE geändert
ØE Ø6:8D C1 1Ø   88         STA SDRIVE
ØE Ø9:8C BE 1Ø   89         STY SSLOT
```

```
ØE ØC:8E 16 ØF   9Ø         STX RDUNIT
ØE ØF:8D E6 1Ø   91 WRINFO: STA TDRIVE
ØE12:8C E3 1Ø    92         STY TSLOT
ØE15:8E 23 ØF    93         STX WRUNIT
ØE18:4C 75 1Ø    94         JMP PRINFO     ;Anzeige
ØE1B:            95 *
ØE1B:AØ ØØ       96 SPREFIX: LDY #ØØ       ;Start neuer Path
ØE1D:2Ø 6E ØE    97         JSR GETPATH    ;kopiert von CMDSTR
ØE2Ø:A9 AF       98         LDA #'/'
ØE22:D9 AØ Ø2    99         CMP PATH,Y     ;letztes Zeichen "/"?
ØE25:FØ Ø4      1ØØ         BEQ GOTPFX     ;ja
ØE27:C8         1Ø1         INY            ;sonst "/" anhängen
ØE28:99 AØ Ø2   1Ø2         STA PATH,Y
ØE2B:8C AØ Ø2   1Ø3 GOTPFX: STY PATH       ;Länge des Path
ØE2E:2Ø 5B ØE   1Ø4         JSR MLISPFX    ;MLI: SET PREFIX
ØE31:DØ 1A      1Ø5         BNE BADPATH
ØE33:           1Ø6 *
ØE33:A9 ØØ      1Ø7 GOTUNIT: LDA #ØØ
ØE35:8D 65 ØE   1Ø8         STA PDRIVE
ØE38:AD 3Ø BF   1Ø9         LDA $BF3Ø      ;"last Unit"
ØE3B:C9 7F      11Ø         CMP #$7F       ;setzt C, wenn Drive 2
ØE3D:2E 65 ØE   111         ROL PDRIVE     ;=> Ø oder 1
ØE4Ø:2Ø 67 ØE   112         JSR SHIFT5     ;xxxxØØØØ => ØØØØxxxx
ØE43:29 Ø7      113         AND #Ø7
ØE45:8D 66 ØE   114         STA PSLOT      ;Slotnummer
ØE48:AD 3Ø BF   115         LDA $BF3Ø
ØE4B:DØ A7      116         BNE SETINFO    ;"always"
ØE4D:           117 *
ØE4D:A9 ØØ      118 BADPATH: LDA #ØØ       ;Pathname auf ""
ØE4F:8D AØ Ø2   119         STA PATH
ØE52:AØ 1D      12Ø         LDY *PATHERR-ERRMSGS
ØE54:AE 9E ØC   121         LDX CMDIDX
ØE57:CA         122         DEX            ;auf letztes Zeichen
ØE58:4C AC ØA   123         JMP DSPERR     ;des Path
ØE5B:           124 *
ØE5B:2Ø ØØ BF   125 MLISPFX: JSR $BFØØ
ØE5E:C6         126         DFB $C6        ;MLI: SET PREFIX
ØE5F:62 ØE      127         DW  PRBLOCK
ØE61:6Ø         128         RTS
ØE62:Ø1         129 PRBLOCK: DFB Ø1        ;1 Parameter
ØE63:AØ Ø2      13Ø         DW  PATH       ;Pathname-Startadresse
ØE65:           131 *
ØE65:           132 PDRIVE: DS 1           ;Scratch für Driveno
ØE66:           133 PSLOT:  DS  1          ;Scratch für Slotno
ØE67:           134 *
ØE67:AØ Ø5      135 SHIFT5: LDY #Ø5        ;vertauscht oberes
ØE69:2A         136 SH51:   ROL A          ;und unteres Nibble
ØE6A:88         137         DEY            ;in Acc
ØE6B:DØ FC      138         BNE SH51
ØE6D:6Ø         139         RTS
ØE6E:           14Ø *
ØE6E:C8         141 GETPATH: INY           ;kopiert Path von
ØE6F:99 AØ Ø2   142         STA PATH,Y     ;von CMDSTR
ØE72:C9 AF      143         CMP #'/'       ;Start neuer Name?
ØE74:DØ Ø3      144         BNE NOTNXT
ØE76:8C 82 ØE   145         STY LNAME      ;ja, Index halten
ØE79:2Ø 83 ØE   146 NOTNXT: JSR GNCHAR     ;nächstes Zeichen
ØE7C:DØ FØ      147         BNE GETPATH    ;nicht Space,"!" etc.
ØE7E:8E 9E ØC   148         STX CMDIDX     ;Path 'abgehakt'
ØE81:6Ø         149         RTS
ØE82:           15Ø *
ØE82:           151 LNAME: DS  1           ;Index auf letztes "/"
ØE83:           152 *
ØE83:E8         153 GNCHAR: INX            ;holt nächstes
ØE84:BD ØØ Ø2   154 GNCHARZ: LDA CMDSTR,X  ;Zeichen aus CMDSTR
ØE87:FØ Ø6      155         BEQ GOTNCHAR   ;CMDSTR-Ende
ØE89:C9 A1      156         CMP #STOP
ØE8B:FØ Ø2      157         BEQ GOTNCHAR   ;RTS mit "EQ", wenn
ØE8D:C9 AØ      158         CMP #$AØ       ;<Space>,"!","/"
ØE8F:6Ø         159 GOTNCHAR: RTS
ØE9Ø:           16Ø *
ØE9Ø:           161 *
ØE9Ø:           162 *
ØE9Ø:2Ø 84 ØE   163 RWBLOCK: JSR GNCHARZ   ;testet auf Space,
ØE93:8E 9E ØC   164         STX CMDIDX     ;"!" etc.
ØE96:FØ 51      165         BEQ DORW       ;R/W ohne Argument
ØE98:EE 9E ØC   166         INC CMDIDX
ØE9B:C9 AB      167         CMP #'+'
ØE9D:FØ 14      168         BEQ BLOCKUP    ;Blocknummer + 1
ØE9F:C9 AD      169         CMP #'-'
ØEA1:FØ 16      17Ø         BEQ BLOCKDN    ;Blocknummer - 1
ØEA3:2Ø 36 ØC   171         JSR EVALHEX    ;folgt Nummer?
ØEA6:9Ø Ø3      172         BCC RWB1       ;ja, "R/W xxxx"
ØEA8:4C AA ØA   173         JMP CMDERR
ØEAB:AD A7 ØC   174 RWB1:   LDA EVREG+1
ØEAE:AE A6 ØC   175         LDX EVREG
ØEB1:9Ø 22      176         BCC RWB3       ;"always"
ØEB3:           177 *
ØEB3:A9 Ø1      178 BLOCKUP: LDA #Ø1       ;Blocknummer + 1
ØEB5:AØ ØØ      179         LDY #ØØ
ØEB7:FØ Ø3      18Ø         BEQ RWB2
```

```
ØEB9:A9 FF     181 BLOCKDN: LDA #$FF     ;Blocknummer - 1
ØEBB:A8        182          TAY
ØEBC:18        183 RWB2:    CLC
ØEBD:2C A2 ØD  184          BIT RDFLAG    ;RBLOCK oder WBLOCK?
ØEC0:1Ø ØB     185          BPL MODWR
ØEC2:6D 19 ØF  186          ADC RBLOCK
ØEC5:AA        187          TAX
ØEC6:98        188          TYA
ØEC7:6D 1A ØF  189          ADC RBLOCK+1
ØECA:4C D5 ØE  19Ø          JMP RWB3
ØECD:6D 26 ØF  191 MODWR:   ADC WBLOCK    ;Änderung von WBLOCK
ØED0:AA        192          TAX
ØED1:98        193          TYA
ØED2:6D 27 ØF  194          ADC WBLOCK+1
ØED5:          195 *
ØED5:2C A2 ØD  196 RWB3:    BIT RDFLAG
ØED8:1Ø Ø6     197          BPL RWB4
ØEDA:8D 1A ØF  198          STA RBLOCK+1  ;mit READ wird auch
ØEDD:8E 19 ØF  199          STX RBLOCK    ;WRITE modifiziert!
ØEE0:8D 27 ØF  2ØØ RWB4:    STA WBLOCK+1
ØEE3:8E 26 ØF  2Ø1          STX WBLOCK
ØEE6:2Ø 75 1Ø  2Ø2          JSR PRINFO    ;Anzeige der Änderung
ØEE9:          2Ø3 *
ØEE9:          2Ø4 *
ØEE9:2C A2 ØD  2Ø5 DORW:    BIT RDFLAG
ØEEC:1Ø ØB     2Ø6          BPL DOWRITE
ØEEE:2Ø ØE ØF  2Ø7          JSR MLIREAD   ;READ BLOCK
ØEF1:DØ ØC     2Ø8          BNE RWERR
ØEF3:2Ø 5C 11  2Ø9          JSR DUMP      ;evtl. neue Anzeige
ØEF6:4C FE ØE  21Ø          JMP RWDONE    ;des BUFFER-Inhalts
ØEF9:          211 *
ØEF9:2Ø 1B ØF  212 DOWRITE: JSR MLIWRITE
ØEFC:DØ Ø1     213          BNE RWERR
ØEFE:6Ø        214 RWDONE:  RTS
ØEFF:          215 *
ØEFF:AØ 54     216 RWERR:   LDY #ERRWR-ERRMSGS
ØF01:C9 2B     217          CMP #$2B      ;WRITE PROTECTED?
ØF03:DØ Ø2     218          BNE RWERRZ
ØF05:AØ 44     219          LDY #WPROT-ERRMSGS
ØF07:AE 9E ØC  22Ø RWERRZ:  LDX CMDIDX    ;Index auf letztes
ØF0A:CA        221          DEX           ;Zeichen im Command
ØF0B:4C AC ØA  222          JMP DSPERR
ØF0E:          223 *
ØF0E:2Ø ØØ BF  224 MLIREAD: JSR $BFØØ
ØF11:8Ø        225          DFB $8Ø       ;MLI: READ BLOCK
ØF12:15 ØF     226          DW  RPBLOCK
ØF14:6Ø        227          RTS
ØF15:Ø3        228 RPBLOCK: DFB Ø3        ;3 Parameter
ØF16:6Ø        229 RDUNIT:  DFB $6Ø       ;Slot 6, Drive 1
ØF17:ØØ 2Ø     23Ø          DW  BUFFER    ;Zieladresse
ØF19:ØØ ØØ     231 RBLOCK:  DW ØØØØ       ;Blocknummer
ØF1B:          232 *
ØF1B:2Ø ØØ BF  233 MLIWRITE: JSR $BFØØ
ØF1E:81        234          DFB $81       ;MLI: WRITE BLOCK
ØF1F:22 ØF     235          DW  WRPBLOCK
ØF21:6Ø        236          RTS
ØF22:Ø3        237 WRPBLOCK: DFB Ø3       ;3 Parameter
ØF23:6Ø        238 WRUNIT:  DFB $6Ø       ;Slot 6, Drive 1
ØF24:ØØ 2Ø     239          DW  BUFFER    ;Quelladresse
ØF26:ØØ ØØ     24Ø WBLOCK:  DW ØØØØ       ;Blocknummer
ØF28:          243 *
ØF28:          244 ******************
ØF28:          245 *---§FILENAME---*
ØF28:          246 ******************
ØF28:          247 *
ØF28:A9 8Ø     248 ATFILE:  LDA #$8Ø      ;für SETINFO und
ØF2A:8D A2 ØD  249          STA RDFLAG    ;READ BLOCK
ØF2D:AE 9E ØC  25Ø          LDX CMDIDX
ØF3Ø:AØ ØØ     251          LDY #ØØ       ;Annahme: voller Path
ØF32:BD ØØ Ø2  252          LDA CMDSTR,X  ;Path: 1.Zeichen
ØF35:C9 AF     253          CMP #'/'      ;Start mit "/"?
ØF37:FØ Ø8     254          BEQ ISPFX     ;ok- eigenes Prefix
ØF39:AC AØ Ø2  255          LDY PATH      ;altes Prefix?
ØF3C:FØ 4E     256          BEQ NOPFX     ;"No Prefix?"
ØF3E:8C 82 ØE  257          STY LNAME     ;Index zum letzten "/"
ØF41:2Ø 6E ØE  258 ISPFX:   JSR GETPATH   ;CMDSTR => PATH
ØF44:98        259          TYA           ;Index letztes Zeichen
ØF45:AC 82 ØE  26Ø          LDY LNAME     ;Index letztes "/"
ØF48:CØ Ø1     261          CPY #Ø1       ;= erstes Zeichen?
ØF4A:FØ 2Ø     262          BEQ GETVDIR   ;nur ein Name: VolDIR
ØF4C:8D 82 ØE  263          STA LNAME     ;für Vergleich
ØF4F:8C AØ Ø2  264          STY PATH      ;Länge "Prefix-Path"
ØF52:A2 ØØ     265          LDX #ØØ
ØF54:C8        266 GETLAST: INY           ;nächstes Zeichen,
ØF55:E8        267          INX           ;bzw. am "/" vorbei
ØF56:EØ 1Ø     268          CPX #16       ;Name > 15 Zeichen?
ØF58:BØ 21     269          BCS BADFNAME
ØF5A:B9 AØ Ø2  27Ø          LDA PATH,Y    ;Kopie letzter Name
ØF5D:29 7F     271          AND #$7F      ;des Path nach FNAME
ØF5F:9D 91 ØF  272          STA FNAME,X
ØF62:CC 82 ØE  273          CPY LNAME     ;Path-Ende?
```

```
ØF65:9Ø ED     274          BCC GETLAST
ØF67:8E 91 ØF  275          STX FNAME     ;Länge letzter Name
ØF6A:BØ 35     276          BCS SETDIR    ;"always"
ØF6C:          277 *
ØF6C:8D AØ Ø2  278 GETVDIR: STA PATH      ;Path-Gesamtlänge
ØF6F:2Ø 5B ØE  279          JSR MLISPFX   ;SET PREFIX
ØF72:DØ Ø7     28Ø          BNE BADFNAME
ØF74:A2 Ø2     281          LDX #Ø2       ;nur ein Name:
ØF76:A9 ØØ     282          LDA #ØØ       ;VolDIR (Block ØØØ2)
ØF78:4C 21 1Ø  283          JMP GOTVDIR
ØF7B:          284 *
ØF7B:2Ø 6C 1Ø  285 BADFNAME: JSR MLICLOSE ;CLOSE aller Files
ØF7E:A9 ØØ     286          LDA #ØØ       ;Pathname auf "" -
ØF8Ø:8D AØ Ø2  287          STA PATH      ;löscht leider Prefix
ØF83:AØ 1D     288          LDY #PATHERR-ERRMSGS
ØF85:AE 9E ØC  289          LDX CMDIDX
ØF88:CA        29Ø          DEX
ØF89:4C AC ØA  291          JMP DSPERR    ;"File not found.."
ØF8C:          292 *
ØF8C:AØ 85     293 NOPFX:   LDY #PFXNONE-ERRMSGS
ØF8E:4C AC ØA  294          JMP DSPERR    ;"No Prefix?"
ØF91:          295 *
ØF91:          296 FNAME:   DS  16        ;Filename+Länge
ØFA1:          297 *
ØFA1:2Ø 37 1Ø  299 SETDIR:  JSR DIRINFO   ;GET FILE INFO für
ØFA4:DØ D5     3ØØ          BNE BADFNAME  ;letzte DIR im Path
ØFA6:AD 45 1Ø  3Ø1          LDA STYPE     ;aus F'INFO
ØFA9:C9 ØD     3Ø2          CMP #$ØD      ;muß (Sub)DIR sein
ØFAB:9Ø CE     3Ø3          BCC BADFNAME  ;SType < $D/$F
ØFAD:2Ø 5Ø 1Ø  3Ø4          JSR DIROPEN   ;OPEN der Directory
ØFBØ:DØ C9     3Ø5          BNE BADFNAME
ØFB2:AD 5C 1Ø  3Ø6          LDA OREFNO    ;Refno von OPEN
ØFB5:8D 65 1Ø  3Ø7          STA RREFNO    ;=> Refno für READ
ØFB8:2Ø 5D 1Ø  3Ø8          JSR DIRREAD   ;READ erster Block
ØFBB:DØ BE     3Ø9          BNE BADFNAME
ØFBD:8D 33 1Ø  31Ø          STA BLOCK1    ;=Ø: Flag 1.Block
ØFCØ:AD 23 2Ø  311          LDA BUFFER+$23 ;READ-Ziel: BUFFER,
ØFC3:8D 34 1Ø  312          STA ELEN      ;der Header der DIR
ØFC6:AD 24 2Ø  313          LDA BUFFER+$24 ;enthält die Länge
ØFC9:8D 35 1Ø  314          STA ECOUNT    ;und Anzahl der
ØFCC:8D 36 1Ø  315          STA FCOUNTER  ;Einträge pro Block
ØFCF:          316 *
ØFCF:A9 Ø4     317 NXTBLOCK: LDA #>BUFFER+4 ;Basis des ersten
ØFD1:85 ØØ     318          STA PTR       ;Eintrags: Byte ØØØ4
ØFD3:A9 2Ø     319          LDA #<BUFFER+4
ØFD5:85 Ø1     32Ø          STA PTR+1
ØFD7:AD 33 1Ø  321          LDA BLOCK1    ;Header überspringen,
ØFDA:FØ 16     322          BEQ NOTFOUND  ;wenn 1. DIR-Block
ØFDC:          323 *
ØFDC:AC 91 ØF  324 NXTFILE: LDY FNAME     ;Namenslänge File
ØFDF:B9 91 ØF  325 SCAN:    LDA FNAME,Y   ;Name des Files
ØFE2:D1 ØØ     326          CMP (PTR),Y
ØFE4:DØ ØC     327          BNE NOTFOUND
ØFE6:88        328          DEY           ;Vergleich ohne
ØFE7:DØ F6     329          BNE SCAN      ;die Namenslänge
ØFE9:B1 ØØ     33Ø          LDA (PTR),Y   ;Namenslänge+SType
ØFEB:29 ØF     331          AND #$ØF      ;SType maskiert
ØFED:D9 91 ØF  332          CMP FNAME,Y
ØFFØ:FØ 24     333          BEQ GOTFILE
ØFF2:          334 *
ØFF2:CE 36 1Ø  335 NOTFOUND: DEC FCOUNTER ;alle Files dieses
ØFF5:FØ ØE     336          BEQ BLOCKEND  ;Blocks verglichen?
ØFF7:A5 ØØ     337          LDA PTR       ;nein,
ØFF9:18        338          CLC           ;nächster Eintrag
ØFFA:6D 34 1Ø  339          ADC ELEN
ØFFD:85 ØØ     34Ø          STA PTR
ØFFF:9Ø DB     341          BCC NXTFILE
1ØØ1:E6 Ø1     342          INC PTR+1
1ØØ3:DØ D7     343          BNE NXTFILE   ;"always"
1ØØ5:          344 *
1ØØ5:AD 35 1Ø  345 BLOCKEND: LDA ECOUNT   ;neuer Zähler für
1ØØ8:8D 36 1Ø  346          STA FCOUNTER  ;den nächsten Block
1ØØB:EE 33 1Ø  347          INC BLOCK1    ;"1.Block" zurück
1ØØE:2Ø 5D 1Ø  348          JSR DIRREAD   ;nächster Block
1Ø11:FØ BC     349          BEQ NXTBLOCK  ;nach BUFFER
1Ø13:4C 7B ØF  35Ø          JMP BADFNAME  ;"..File not found"
1Ø16:          351 *
1Ø16:2Ø 6C 1Ø  352 GOTFILE: JSR MLICLOSE ;CLOSE der DIR
1Ø19:AØ 11     353          LDY #$11      ;Bytes $11/12 im
1Ø1B:B1 ØØ     354          LDA (PTR),Y   ;File-Eintrag sind
1Ø1D:AA        355          TAX           ;die Nummer des
1Ø1E:C8        356          INY           ;ersten File-Blocks
1Ø1F:B1 ØØ     357          LDA (PTR),Y
1Ø21:8E 19 ØF  358 GOTVDIR: STX RBLOCK   ;Einsprung für VolDIR
1Ø24:8D 1A ØF  359          STA RBLOCK+1
1Ø27:8E 26 ØF  36Ø          STX WBLOCK    ;WRITE wird
1Ø2A:8D 27 ØF  361          STA WBLOCK+1  ;mit geändert!
1Ø2D:2Ø 33 ØE  362          JSR GOTUNIT   ;Unitno => Sx,Dx
1Ø3Ø:4C E9 ØE  363          JMP DORW      ;liest den Block
1Ø33:          364 *
1Ø33:          366 BLOCK1:  DS 1          ;Flag: "1.DIR-Block"
```

```
1034:                 367 ELEN:  DS  1          ;Eintragslänge in der DIR
1035:                 368 ECOUNT: DS 1          ;Anzahl Einträge/DIR-Block
1036:                 369 FCOUNTER: DS 1        ;Eintrags-Zähler
1037:                 370 *
1037:20 00 BF         371 DIRINFO: JSR $BF00    ;GET FILE INFO für die
103A:C4               372         DFB $C4       ;DIR "vor" dem File
103B:3E 10            373         DW  INFOBLOCK
103D:60               374         RTS
103E:0A               375 INFOBLOCK: DFB 10     ;10 Parameter
103F:A0 02            376         DW  PATH      ;Pathname zur DIR
1041:                 377         DS  4         ;ACCESS, F-&AUX-Type
1045:                 378 STYPE: DS  1          ;Storage Type
1046:                 379         DS  10        ;restliches INFO
1050:                 380 *
1050:20 00 BF         381 DIROPEN: JSR $BF00    ;OPEN der Directory
1053:C8               382         DFB $C8
1054:57 10            383         DW  OPENBLOCK
1056:60               384         RTS
1057:03               385 OPENBLOCK: DFB 03     ;3 Parameter
1058:A0 02            386         DW  PATH      ;Pathname zur DIR
105A:00 26            387         DW  EDITEND   ;Buffer: Programmende
105C:                 388 OREFNO: DS 1          ;Returnte Refno
105D:                 389 *
105D:20 00 BF         390 DIRREAD: JSR $BF00    ;READ der Directory
1060:CA               391         DFB $CA
1061:64 10            392         DW  READBLOCK
1063:60               393         RTS
1064:04               394 READBLOCK: DFB 04     ;4 Parameter
1065:                 395 RREFNO: DS 1          ;Refno von OPEN
1066:00 20            396         DW  BUFFER    ;Ziel ist BUFFER
1068:00 02            397         DW  $0200     ;Request: 1 Block
106A:                 398         DS  2         ;Transfer Count
106C:                 399 *
106C:20 00 BF         400 MLICLOSE: JSR $BF00   ;CLOSE aller Files
106F:CC               401         DFB $CC
1070:73 10            402         DW  CLSBLOCK
1072:60               403         RTS
1073:01               404 CLSBLOCK: DFB 01      ;1 Parameter
1074:00               405         DFB 00        ;Refno 00 => ALL
1075:                 406 *
1075:                 407 ************************
1075:                 408 *
1075:A9 03            409 PRINFO: LDA #3        ;für 80Z: VTAB 3
1077:A0 04            410         LDY #04       ;für PDUMP
1079:2C 9B 0C         411         BIT PRINTER
107C:30 09            412         BMI PINFO1    ;<CR> und INFO
107E:A0 00            413         LDY #00       ;für Screen: 4 Spaces
1080:2C 9A 0C         414         BIT IS40
1083:10 02            415         BPL PINFO1
1085:A9 02            416         LDA #2        ;für 40Z: VTAB 2
1087:20 55 0B         417 PINFO1: JSR SETVTB
108A:20 A5 10         418         JSR PINFO2    ;"Source Slot.."
108D:AD 1A 0F         419         LDA RBLOCK+1  ;und Blocknummer
1090:AC 19 0F         420         LDY RBLOCK
1093:20 19 0B         421         JSR PRTAY     ;Blocknummer
1096:A0 20            422         LDY #INFO2-INFO1
1098:20 A5 10         423         JSR PINFO2    ;"Target Slot.."
109B:AD 27 0F         424         LDA WBLOCK+1  ;und Blocknummer
109E:AC 26 0F         425         LDY WBLOCK
10A1:20 19 0B         426         JSR PRTAY
10A4:60               427         RTS
10A5:                 428 *
10A5:B9 B1 10         429 PINFO2: LDA INFO1,Y
10A8:F0 06            430         BEQ PINFO3
10AA:20 E6 0A         431         JSR PRCHAR
10AD:C8               432         INY
10AE:D0 F5            433         BNE PINFO2
10B0:60               434 PINFO3: RTS
10B1:                 435 *
10B1:A0 A0 A0         436 INFO1: ASC '    Source: S'
10BE:B6 AC C4         437 SSLOT: ASC '6,D'      ;INFO1: 4 Spaces
10C1:B1 A0 A0         438 SDRIVE: ASC '1    Blockno: $'
10D0:00               439         DFB 00        ;SDRIVE: 4 Spaces
10D1:A0 A0 A0         440 INFO2: ASC '          '
10DA:D4 E1 F2         441         ASC 'Target: S' ;INFO2: 9 Spaces
10E3:B6 AC C4         442 TSLOT: ASC '6,D'
10E6:B1 A0 A0         443 TDRIVE: ASC '1    Blockno: $'
10F5:00               444         DFB 00        ;TDRIVE: 4 Spaces
10F6:                 445 *
10F6:                 446 ************************
10F6:                 447 *
10F6:                 448         CHN ED.FUNCS1.TXT
```

ED.FUNCS1.TXT1

```
10F6:                   1         PAGE
10F6:                   2 ***************
10F6:                   3 *---MONITOR---*
10F6:                   4 ***************
10F6:                   5 *
10F6:68                 6 XMON:   PLA             ;die Returnadresse
10F7:8D 38 11           7         STA XMONRTS     ;innerhalb von EDIT
10FA:68                 8         PLA             ;wird gespeichert
10FB:8D 39 11           9         STA XMONRTS+1
10FE:                  10 *
10FE:A2 00             11         LDX #00         ;SAVE von CMDSTR,
1100:BD 00 02          12 SCMDSTR: LDA CMDSTR,X   ;der Monitor benutzt
1103:9D 00 26          13         STA EDITEND,X   ;ebenfalls $200... als
1106:E8                14         INX             ;Input-Buffer!
1107:D0 F7             15         BNE SCMDSTR
1109:                  16 *
1109:20 5B 0B          17         JSR DOCR        ;ein <CR>
110C:A9 1C             18         LDA #>MONRTS    ;Rücksprungadresse
110E:8D F9 03          19         STA YVEC        ;zu EDIT in den
1111:A9 11             20         LDA #<MONRTS    ;Ctrl-Y Vektor
1113:8D FA 03          21         STA YVEC+1
1116:20 88 0D          22         JSR DOSCONN     ;DOS anhängen
1119:4C 69 FF          23         JMP MONITOR     ;"RTS": Ctrl-Y
111C:                  24 *
111C:20 7D 0D          25 MONRTS: JSR DISCONN     ;DOS wieder ab
111F:A2 00             26         LDX #00         ;RESTORE CMDSTR
1121:BD 00 26          27 RCMDSTR: LDA EDITEND,X
1124:9D 00 02          28         STA CMDSTR,X    ;von EDITEND nach
1127:E8                29         INX             ;$200...
1128:D0 F7             30         BNE RCMDSTR
112A:                  31 *
112A:AD 39 11          32         LDA XMONRTS+1   ;RESTORE der
112D:48                33         PHA             ;RTS-Adresse
112E:AD 38 11          34         LDA XMONRTS     ;innerhalb von EDIT
1131:48                35         PHA
1132:4C 3A 11          36         JMP REPRINT     ;RESTORE Bildschirm
1135:                  37 *
1135:4C AA 0A          38 MONERR: JMP CMDERR
1138:                  39 XMONRTS: DS 2           ;RTS (intern)
113A:                  40 *
113A:20 BD 08          41 REPRINT: JSR TITLE      ;Neudruck des
113D:20 75 10          42         JSR PRINFO      ;Bildschirms
1140:AD AE 0C          43         LDA BPLINE      ;Bytes per Line
1143:20 5B 11          44         JSR SETDUMP     ;DUMP BUFFER
1146:20 47 0B          45         JSR SETV221     ;VTAB22, HTAB 1
1149:20 2D 09          46         JSR RTYP2       ;CMDSTR (alt)
114C:4C 1C 09          47         JMP RETYPE      ;CMDSTR (neu)
114F:                  48 *
114F:                  51 *************
114F:                  52 *---QUIT---*
114F:                  53 *************
114F:                  54 *
114F:20 4F 0B          55 QUIT:  JSR SETV22       ;VTAB 22
1152:20 9C 0B          56        JSR CLEOP
1155:4C 98 0D          57        JMP EXIT         ;DOS Kaltstart
1158:                  58 *
1158:                  60        CHN ED.FUNCS2.TXT
```

ED.FUNCS2.TXT1

```
1158:4C AA 0A           1 GETVAR: JMP CMDERR
115B:60                 2 SETDUMP: RTS            ;Temp!
115C:60                 3 DUMP:   RTS             ;Temp!
115D:                   4 *
115D:                   5         CHN ED.CMDS.TXT
```

ED.CMDS.TXT1

```
115D:                   1         PAGE
115D:                   2 ****************
115D:                   3 *---COMMANDS---*
115D:                   4 ****************
115D:                   5 *
115D:                   6 CMDS:   EQU *
115D:C0                 7         ASC '§'         ;Lädt den ersten Block
115E:28 0F              8         DW  ATFILE      ;eines Files
1160:D8                 9         ASC 'X'         ;MONITOR-Aufruf
1161:F6 10             10         DW  XMON
1163:D1                11         ASC 'Q'         ;Programm-Ende
1164:4F 11             12         DW  QUIT
1166:D2                13         ASC 'R'         ;Read Block/Prefix
1167:                  14 *
1167:9B 0D             15         DW  READ
1169:D7                16         ASC 'W'         ;Write Block/Prefix
116A:A3 0D             17         DW  WRITE
116C:                  18 *
000F:                  20 LCMD:   EQU *-CMDS
116C:                  21 *
116C:                  24 ******************************
116C:                  25 *
0200:                  26 BUFLEN: EQU MAXBUF*$100
116C:                  27 *
2000:                  28 BUFFER: EQU $2000
2200:                  29 HLBUF:  EQU BUFFER+BUFLEN
2400:                  30 FNBUF:  EQU HLBUF+BUFLEN
2500:                  31 FNMASK: EQU FNBUF+$100
2600:                  32 EDITEND: EQU FNMASK+$100
116C:                  33 *
116C:                  34         LST OFF
```

peeker Börse

Verkauf Hardware

Verkaufe Applehardware wegen Systemumrüstung an Privat von Privat.
Michael Sprengel,
Amselweg 4, 7903 Laichingen 3

Notverkauf IBM-Komp. PC (XT) HIT, 2 Laufw., Monitor, Software; NP 3900,–, VHB 3500,–
Tel. 06231/7467

Apple II+, 22W, c+r: 1650,– EPSON APL B: 80,– (+ Kabel)
Tel. 0631/12831

Apple IIe mit integrierter 10MB 3,5 Zoll H-Disk, 1 Laufwerk, 1 Bubble Memory, Drucker If, RGB, 64K, Serial IF u. Apple Monitor (alles neu + original) zu VB DM 5500,–
EBV Elektronik, Tel. 069/7850 37

II+ Comp. 64K, Z80, 80Z, 32K-Druckerkarte, 2 Slimline LW, Grün-Mon. Ext. 10er Block, viel Profisoftware + Literatur VHB 1200,–
Tel. 06147/8757

Apple II+, Original Profi-System; Anfragen nur Tel. 05105/84661 ab 18.00 Uhr. Preis VHB

Fernschreiberinterface am Gameport m. Programm DM 79,–
P. Benner, Hubertusstr. 131, 4150 Krefeld

Apple IIe, 128KB, Z80, Mon. III, SSC, 2 LW m/2 Contr., insg. 783KB, über 100 Disk., incl. Handb., VB DM 4500,–
Tel. 02582/1236 ab 18.00 Uhr

Apple IIe, 2 Disk 140K, Monitor, 128K RAM, 80Z, Z80, Mouse, EPSON FX-80 + Software: WS, Db2, MP, PASCAL TURBO & MT + FOR, COB, Quickfile, Mousepaint Applewriter/DM 6000,–
Tel. 040-551 12 45

Apple II K. 64K 498,– Monitor-BMC 250,–, 2 Teak FD-55F á 298,–, Erphi-Contr. 198,– Z80+80Z á 68,–, Softw. + Lit. Joyst. Tel. 0831/95558

Verkaufe: APPLE-Zubehör
Tel. 0621/711604 nach 17 Uhr

Apple IIc + Joy. (orig.) + Soft. für DM 1900,–. T. 06421/45547

IEEE-Karte mit Anleitung und Kabel DM 200,–.
Tel. 0511/513085 (abends)

Österreich: Apple II+ orig., 64K, 2 Apple-Laufw., Apple-Monitor öS 18.000,–, Z80-K. (CP/M) u. Orig. VIDEX 80Z-Karte öS 4.000,–, EPSON MX-80-Drucker mit Grafikinterf. öS 7.000,–, Joystick u. Literatur öS 2.000,–,
Tel. 07223/31623 abends

Apple II komp.
64KB, 80 Zeich. Card. LW-Controller 1×40–2×80 Track, Parallelintf. IBM-Look, abgesetzte Tastatur, Monitor VB 1900,–.
Tel. 09122/81945

Tastatur Operator IIe mit Interface (400,–), Ramdisk mit Software (CP/M, DOS, PASCAL) IBS AP 17 (300,–), Typenraddrukker EPSON DX 100 (baugleich mit Borther HR 15) mit B+K-Centronix-Interface, Kabel und diversem Zubehör, voll angepaßt an Appleworks (1000,–) Tel. 02 21/77 93 29

Apple II + Kompatibler, Z80, MACRO-Befehle auf Tastatur mit Zahlenfeld, Preis: VS
Tel. 02174/62302 abd.

Apple IIc, Monitor mit Stand, 2. Laufwerk, Z80-Karte und Software kompl. DM 3200,–
Tel. 02302/62467

Apple IIe Monitor3 mit Ständer, 2-Laufwerke, grafikfähiges DR Interface 80Z 128KB PRODOS + Editor Writer IIe ** DM 4000 **
Tel. 05021/61183

IIe + Neue ROMs + 65C02; 2 LW; Uhr; Maus; getr. Tast.; Drukker; je 150 Handb. + Progr.; neuw. 50 % VB, Tel. 02228/677423

Apple 96 E/A Slotkarte C 4* PIO 8255)! Für Meß- und Steuerungsaufgaben! Info kostenlos gegen Freiumschlag. RS Elektro Gerätebau, Hinter den Hörsten 87, 4992 Espelkamp

Dot-Matrix-Printer, original II, Itoh 8510A, Binder 8510A, NEC 8023A, div. Zusatzkarten Apple-Compatib.
A. Böck, T. 02102/68610, Ratingen

Macintosh 128 auf 512KB DM 299,–. Fa. Schlösser, Tel. ab 17.00 Uhr, 089-985889

Ankauf Hardware

Suche – defekte – oder – nicht mehr gebrauchte – Computer und Peripherie.
Chiffre Nr. P 1008

Apple II Graphics Tablet gesucht bis DM 1000,–.
Tel. 030/21141 72

Verkauf Software

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

DEUTSCHSPRACHIGES APPLESOFTBASIC
100 % KOMPATIBEL-Wahlweise zum Rom basic- in 16K-Karte + BEFEHLSED. Erstellen Sie Ihr eigenes Basic. Ab jetzt spricht der Computer Ihre Sprache! Diskette DM 39,– bei
Thies Martens, Ernestinenstr. 17, 2300 KIEL 14, T. 0431/735353

Applewriter IIe, orig. Disketten u. Handbücher (AW + TKS), Preis VS – Tel. 0203/767499

Spitzensoftware für Apple II z.B.: Fileschutz: DM 50,–, Fußballtabellenverwaltung: DM 30,– ... Info (0,80 DM) bei W. Rittmeyer, Wehrbruchweg 30, 4060 Viersen 1

Hardcopy in UCSD-Pascal, voll menügesteuert, druckt von 52 × 67 mm – 176 × 211 mm – fast – stufenlos. Bis zu 3 Grafiken nebeneinander möglich. DM 45,–
Prause, Neumannstr. 9,
5600 Wuppertal 2

The Apple IIc Reference Manual, 2 Bände, nagelneu, Neupreis DM 195,– zum Preis von DM 100,–. Tel. 06198/8145

––– STOCKMASTER II –––
Das Apple II-Programm für echte Börsengewinne: 485,– DM/SFr bei: Töngi Computer-Praxis, Aspeltstr. 4, D-6500 Mainz.
für die Schweiz: Denton Consultants AG, Auwisstr. 17, CH-8127 Forch/Zürich.

Jane (Original), Deutsche Version DM 170,–
Tel. 02226/3008

WINDOW-Technik unter APPLESOFT für alle APPLE II-Typen DM 35,–, Info DM 1,– Porto von Ernst Heinz, Waldgürtel 7, 5060 Berg. Gld. 1

Erphi+Focu Benutzer Diskreditor zum Bearbeiten von Disketten 55,–. H. Eschborn, Zehnthof 2, 6522 Osthofen, T. 06242/4006

Apple II: Public Domain, DFÜ-Progr. Kermit: Je Volume DM 15,– Lehrerprogr., Graphiksprache „Minilogo". Gratisinfo:
Fa. Waltraud Muhle, Waldwinkel 3, 2105 Seevetal 3

Topsoftware zu Niedrigstpreisen z.B.: Ballblazer: DM 110,–
Info (0,80 DM) bei W. Rittmeyer, Wehrbruchweg 30, 4060 Viersen 1

Ankauf Software

D BASE II Prg. f. KD-Datei mit Druck + Mailmerge, Übernahme f. Apple II XEBEC-10MB Festpl. CP/M Start VP.
P. Lehmann, P.O. Box 70 11 61, 2000 Hamburg 70

UCSD-PASCAL, CPM, Printshop, Uhrenkarte, AppleWorks??? Tel. 06571/6571

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

Systemwechsel Superpreise alles neuwertig! Accelerator IIe 450,–, Word-Juggler (Dt. Textv. incl. Wörterb.) 450,–, IIc 1700,–, Imagewriter 800,–, 10 MB-HD 2500,–; Gernot Eiler,
Tel. 0043-05513/67083

Ihre Erphi-Vertretung für die Schweiz: Beltronic, Im Chapf, 8455 Rüdlingen, Tel. 01867 31 41, Telex 825981.

PASCMAK: Suche Vertriebspartner und Interessenten für mein neues Immobilienmakler-Programm. Heinz, Waldgürtel 7, 5060 Berg. Gld. 1

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

1. Stuttgarter Privat-Micro-Computer-Flohmarkt am 4.5.86 in Stuttgart. Teilnahme gegen Unkostenbeitrag, Anmeldung + Info Tel. 0711/60718 52 + 60650

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, ca. DM 40,–
- ☐ Gabriel, Applewriter, 3-7785-1234-X, DM 35,–
- ☐ Hagemüller, Microsoft-BASIC, Bd. 1, 3-7785-1038-X
- ☐ 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, ca. 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# _____
  Peeker 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 48,–
- ☐ DB-Meister, Programm, DM 290,–
- ☐ Superplot, Programm, DM 48,–
- ☐ Superquick, Programm, DM 48,–
- ☐ Turtle Graphics, Programm, DM 98,–
- ☐ Disk 40, Programm, DM 48,–
- ☐ Kyan-Pascal 2.0, Programm, DM 170,–
- ☐ Fast-Writer, Programm, DM 98,–

_____
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.

Buch-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker
Leserservice

Dr. Alfred Hüthig Verlag GmbH

Postfach 10 28 69

6900 Heidelberg

POSTKARTE

Peeker
Buchabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

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

Softbreaker 1.0

Eine softwaremäßige Interrupt-Utility für die Apple IIe 64K-Karte

von U. Stiehl

1984, Diskette und Manual, DM 48,–
ISBN 3-7785-1022-3

Softbreaker ist ein Assemblerprogramm, mit dessen Hilfe Programme, die sich von der 64K-Karte (= Extended 80 Column Card für den Apple IIe) starten lassen, unterbrochen, gespeichert, geladen und exakt an der Stelle der Unterbrechung fortgeführt werden können. Dadurch ist es auch möglich, Sicherungskopien von sogenannten kopiergeschützten Programmen herzustellen.
Mit Softbreaker unterbrochene Programme werden komplett, d. h. die ganzen 64K einschließlich Language Card, in nur ca. 11 Sekunden auf einer formatierten Diskette gesichert.
Gerätevoraussetzung: Apple IIe mit 64K-Karte, nicht IIc, nicht neue ROMs

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

Superschnelle Bildschirmausgabe in Apple-Pascal

von Dr. Wolfgang Braun

Die hier vorgestellten externen Prozeduren ermöglichen es in Apple-Pascal-Programmen, Texte und Daten wesentlich schneller als mit der systemeigenen WRITELN-Prozedur auf dem 80-Zeichen-Bildschirm auszugeben. Kenntnisse in Assembler sind dazu nicht erforderlich. Erfahrungen im Umgang mit dem Apple-Pascal-System zur Einbindung der hier abgedruckten Routinen in eigene Programme sind von Vorteil.

Solche Prozeduren sind vor allem dann interessant, wenn häufig Daten, Texte, Tabellen oder längere Menüs usw. auf dem Bildschirm ausgegeben werden. Es kann sehr lästig sein, wenn man ständig einige Sekunden warten muß, bis der Bildschirm den gewünschten Text enthält. Die WRITELN-Prozedur, oft noch gekoppelt mit einer GOTOXY-Anweisung zur Cursorpositionierung, arbeitet recht langsam. Schreibt man damit eine 80 Zeichen lange Zeichenkette in die 24 Zeilen des Bildschirmes, so vergehen dabei 3.6 Sekunden. Eine der im folgenden beschriebenen Prozeduren benötigt für die gleiche Aufgabe nur einen Bruchteil dieser Zeit, nämlich 0.1 Sekunden. Diese Prozeduren sind speziell zur Verwendung in selbst erstellten Programmen gedacht. Man beachte jedoch, daß aus Gründen der Geschwindigkeit kein Scrollen implementiert wurde, so daß nur starre Bildschirmmasken ausgegeben werden können.

Pascal-Programme, in die diese externen Prozeduren eingebunden worden sind, laufen nur auf einem Apple IIe mit 80-Zeichenkarte (mit oder ohne 64K-Speichererweiterung) oder IIc.

Im folgenden wird die Anwendung dreier externer Assemblerroutinen beschrieben, mit denen es in Pascal-Programmen möglich ist, Zeichenketten, Chars oder Integerzahlen in den Modi normal oder invers an beliebigen Positionen auf dem Bildschirm zu plazieren. Die Verwendung der WRITELN-Prozedur und auch aller anderen Pascal-Prozeduren wird dadurch nicht berührt. Zunächst wird die Syntax und dann die Einbindung dieser Routinen in Pascal-Programme erläutert. Um die Namen der Prozeduren kurz und einprägsam zu gestalten, wurden dem Buchstaben P drei weitere Buchstaben zur Verdeutlichung der jeweiligen Funktion nachgestellt. Die Prozeduren haben sich nach häufiger Anwendung auch in langen und komplexen Pascal-Programmen bestens bewährt. Sie arbeiten in beiden Pascal-Versionen 1.1 und 1.2 einwandfrei.

1. Ausgabe von Zeichenketten

Die Ausgabe von Zeichenketten geschieht mit der Prozedur der Form

PSTR (M,X,Y,STR);

Die Größen M, X, Y und STR haben dabei folgende Bedeutung: M ist entweder 0 oder 1 oder eine Variable, die nur diese beiden Werte annehmen darf. Bei M=0 erfolgt die Ausgabe im Modus normal, bei M=1 erfolgt sie im Modus invers. X und Y sind die Spalten- bzw. Zeilennummer des ersten Zeichens der Zeichenkette auf dem Bildschirm. Die Wertebereiche sind die gleichen wie beim systemeigenen GOTOXY(X,Y)-Befehl. STR ist eine Zeichenkette oder String-Variable. Die Zeichenkette kann leer sein und darf höchstens 80 Zeichen enthalten. Sie darf über den rechten Rand des Bildschirms nicht hinausreichen.

Beispiel: Die Prozedur
PSTR(1,7,15,'Test')
gibt den Text „Test" in inverser Darstellung auf den Bildschirm aus. Der Buchstabe „T" befindet sich in der 8. Spalte und 16. Zeile. Anstelle der Zahlenwerte können natürlich auch entsprechend vereinbarte Variablen treten. Der Cursor wird durch diese und auch die weiter unten beschriebenen Prozeduren nicht bewegt. Falls dieser auf dem Bildschirm stört, muß er durch eine GOTOXY-Anweisung auf eine geeignete Position gesetzt werden.
Es sei an dieser Stelle darauf hingewiesen, daß der Programmierer selbst darauf achten muß, daß die Werte in den angegebenen Bereichen liegen. Der Versuch der Ausgabe einer über den rechten Bildschirmrand hinausgehenden Zeichenkette oder die Verwendung unzulässiger Werte für M, X oder Y führt zu unvorhersehbaren Ergebnissen.

2. Ausgabe von CHARS

Die Ausgabe einzelner Zeichen erfolgt durch die Prozedur

PCHA (M,X,Y,CH);

M, X und Y haben die gleiche Bedeutung wie bei der vorangegangenen Prozedur. CH steht für ein einzelnes Zeichen oder für eine Variable vom Typ CHAR. Als Beispiele seien PCHA (0,4,17,'X') bzw. PCHA (0,4,17,CHR(88)) genannt. Das Zeichen „X" erscheint mit den entsprechenden Koordinaten auf dem Bildschirm.
Einzelne Buchstaben könnten natürlich auch als Zeichenkette mit der Prozedur PSTR ausgegeben werden, so daß die Prozedur PCHA scheinbar überflüssig ist. Der Wert einer Char-Variablen kann jedoch mit PSTR nicht ausgegeben werden, weil der Compiler an der Stelle von STR eine String- und keine Char-Variable erwartet. Man kann zwar den Wert einer Char-Variablen in eine String-Variable der Länge 1 übertragen, aber genau dieser Umstand kann mit Hilfe von PCHA vermieden werden.

3. Ausgabe natürlicher Zahlen

Zur Ausgabe von ganzen positiven Zahlen steht die Prozedur der Form

PINT (S,M,X,Y,INT);

zur Verfügung. M, X und Y haben wieder die übliche Bedeutung. S ist eine ganze Zahl zwischen 1 und 5 oder eine entsprechend vereinbarte Variable. S bedeutet die Anzahl der auszugebenden Stellen einschließlich führender Leerstellen. Führende Nullen werden durch Leerzeichen ersetzt.
INT ist eine Zahl zwischen -32767 und 32767 oder eine entsprechend vereinbar-

te Variable. Damit lassen sich ganze positive Zahlen rechtsbündig ohne Vorzeichen ausgeben. Das erste Zeichen der Zahl (Leerzeichen oder Ziffer) hat die Koordinaten X und Y. Die Prozedur PINT (4,0,17,2,123) schreibt die Zahl 123 mit einer führenden Leerstelle an die gewünschte Position. Ist die Zahl S der auszugebenden Stellen kleiner als die Zahl der Ziffern, so werden die S rechten Ziffern ausgegeben. Will man eine Zahl im Bereich 32768 bis 65536 ausgeben, so ist statt dessen die Differenz dieser Zahl und 65536, also eine negative Zahl für INT einzusetzen. PINT (5,0,0,7,-1234) gibt den Wert 64302 aus, denn -1234 = 64302-65536. Die Ausgabe negativer Zahlen oder Zahlen mit Vorzeichen ist mit dieser Prozedur nicht möglich.

4. Einbindung der Routinen

Vor der Einbindung in Pascal-Programme muß der Assembler-Quelltext **PRINT.ASS.TEXT** im Pascal-Editor als Textfile eingegeben und danach mit dem Assembler in 6502-Objektcode assembliert werden. Nach dem erfolgreichen Assemblieren speichert man den Programmtext (Quellcode) und den Maschinencode (Objektcode) mit Hilfe des Befehls S(ave unter einem beliebigen Namen, z. B. PRINT.ASS.TEXT bzw. PRINT.ASS.CODE, auf einer Diskette ab. Dann kann mit dem Einbinden des Maschinencodes in ein Pascal-Programm begonnen werden.

4.1. als externe Assemblerroutinen

In diesem Fall müssen die externen Prozeduren im Pascal-Programmtext selbst als solche deklariert werden. Die Form der Deklaration ersieht man am besten dem Demonstrationsprogramm **DEMOPRINT-.TEXT**. Nach der Eingabe wird der Programmtext compiliert. Der jetzt entstandene Programmcode ist aber noch nicht lauffähig, da der Maschinencode der Ausgaberoutinen noch eingebunden werden muß. Dazu ruft man den L(inker auf und beantwortet die gestellten Fragen wie folgt:

1. Host file? <Return> bzw. Name Ihres Programmes, falls es bereits compiliert und auf Diskette abgespeichert war, und dann erst <Return>.
2. Lib file? #X:PRINT.ASS <Return> (X=Volume-Nr.)
3. Lib file? <Return>, da kein weiterer Code gelinkt werden soll.
4. Map file? <Return>
5. Output file? <Return> bzw. Name des Programmes

Wenn man bei 1. und 5. mit <Return> antwortet, dann liegt der Programmcode auf der Diskette als SYSTEM.WRK.CODE vor, der mit T(ransfer bzw. C(hange umbenannt werden sollte. Andernfalls befindet sich der Programmcode unter dem eingegebenen Namen auf der Diskette mit der benutzten Volume-Nr. Dieser Code kann dann mit X(ecute gestartet werden.

Der Nachteil dieses Verfahrens ist, daß dieser Einbindevorgang jedesmal wiederholt werden muß, wenn das Pascal-Programm neu compiliert werden soll (z.B. nach einer Änderung des Programmes). Dem steht der Vorteil gegenüber, daß die eingebundenen Routinen im Pascal-Programm selbst integriert sind und die SYSTEM.LIBRARY nicht benötigt wird.

4.2. als Library-Assemblerroutinen

Bindet man den Objektcode der Assemblerroutinen in die systemeigene Programm-Bibliothek, die SYSTEM.LIBRARY, ein, so können die drei Ausgabe-Prozeduren von jedem beliebigen Pascal-Programm aufgerufen werden, ohne daß der Objektcode jedesmal wieder eingebunden werden muß. Lediglich am Beginn des Programmes muß dem System durch die Anweisung

USES Unitname;

der Name der neuen Bibliothekseinheit (Unit) mitgeteilt werden. Vorher muß natürlich der Objektcode der Unit einmal in die SYSTEM.LIBRARY eingebracht werden. Um dies zu erleichtern, ist der Programm-Text der dazu erforderlichen Unit **FASTPRINT.TEXT** abgedruckt.

Nach dem Compilieren der Unit wird in diese mit Hilfe des L(inkers nach dem gleichen Schema wie oben der Objektcode PRINT.ASS.CODE eingebunden. Der dabei resultierende Code wird dann z.B. unter dem Namen FASTPRINT.CODE abgespeichert und mit Hilfe des auf der Diskette APPLE3: befindlichen Programms LIBRARY in die SYSTEM.LIBRARY eingebaut.

Die nach dem Programmstart von LIBRARY gestellten Fragen beantwortet man dann folgendermaßen:

1. Output Codefile → <*> <Return> (System.Library)
2. Link Codefile → <*> <Return> (System.Library)
3. <=> (alle Units übernehmen)
4. <N> (New file)
5. Link Codefile → #X:FASTPRINT <Return> (X=Vol.-Nr.)
6. <Slot-Nr. von FASTPRINT in der oberen Tabelle> <Space>
7. Slot to link into? <Nr. eines freien Slots in der unteren Tabelle> <Space>
8. Mit <Q> Programm abschließen
9. Notice? <Return>, falls kein Kommentar erwünscht

Während dieses Vorgangs wird die Original-SYSTEM.LIBRARY durch die nunmehr um die Unit FASTPRINT erweiterte SYSTEM.LIBRARY ersetzt. Damit sind die drei Ausgaberoutinen ständig verfügbar. Programme mit der Anweisung USES FASTPRINT greifen beim Aufruf der obigen Prozeduren auf die SYSTEM.LIBRARY zu, die natürlich dann „online" sein muß.

5. Demo-Programm

Das abgedruckte Demo-Programm DEMOPRINT soll den Geschwindigkeitsvorteil der drei Ausgaberoutinen gegenüber der WRITELN-Prozedur aufzeigen. Vergleichende Versuche haben ergeben, daß die PSTR-Routine zur Ausgabe einer Zeichenkette ca. 35 (fünfunddreißig) mal schneller arbeitet als die WRITELN-Prozedur mit der gleichen Kette. Dabei ist noch zu berücksichtigen, daß mit der WRITELN-Prozedur nur relativ zur laufenden Cursorposition ausgegeben werden kann. Wollte man damit auch noch bestimmte Bildschirmpositionen erreichen, so müßte noch ein GOTOXY-Befehl vorausgehen. Dadurch würde der Geschwindigkeitsgewinn zugunsten von PSTR noch deutlicher ausfallen.

Kurzhinweise

1. Zweck:
Extrem schnelle Ausgabe von starren Bildschirmmasken (ohne Scrollen);
2. Konfiguration:
IIe/c mit 80-Zeichenkarte (nicht II+, nicht Videxkarte!); Pascal 1.1/1.2; Achtung: Beim IIc bzw. IIe mit neuen ROMs werden statt inverser Buchstaben Mauszeichen angezeigt!
3. Test:
PRINT.ASS.TEXT assemblieren und DEMOPRINT.TEXT compilieren, dann PRINT.ASS.CODE und DEMOPRINT.CODE linken zu z.B. SYSTEM.WRK.CODE, der mit R(un gestartet werden kann. Zum Einbinden in Library s. Aufsatztext.
4. Sammeldisk:
PRINT.ASS.TEXT
DEMOPRINT.TEXT
FASTPRINT.TEXT
(Mit GETDOS von Sammeldisk #15 auf Pascal-Diskette konvertieren.)

PRINT.ASS.TEXT

```
;*****************************************************
;*                                                   *
;*              Externe Prozeduren zur               *
;*            schnellen Bildschirmausgabe            *
;*             in Apple-PASCAL 1.1/1.2               *
;*             Dr.W.Braun / Dezember 1984            *
;*                                                   *
;*****************************************************


;********** Softwareschalter **********

STORE8Ø  .EQU ØC ØØ1     ;aktiviert die 8Ø-Zeichen-Karte
MAINRAM  .EQU ØCØ54      ;aktiviert $Ø4ØØ-$Ø7FF Page1 (MAINRAM)
AUXRAM   .EQU ØCØ55      ;aktiviert $Ø4ØØ-$Ø7FF Page2 (AUXRAM)
STATUS   .EQU ØC Ø1C     ;Zustand von MAINRAM/AUXRAM

;********** Hilfsregister *************

RETURN   .EQU  ØØØØ      ;Rückkehradresse nach Pascal
ADRSTR   .EQU  ØØØ2      ;Adresse des Strings
YKOOR    .EQU  ØØØ4      ;y-Koordinate auf dem Bildschirm
XKOOR    .EQU  ØØØ5      ;x-Koordinate auf dem Bildschirm
XKOORH   .EQU  ØØØ6      ;Hilfsregister
ZAHL     .EQU  ØØØ7      ;Zahl der zu schreibenden Zeichen
POSØ     .EQU  ØØØ8      ;Adresse der ersten Spalte im
                         ;Bildschirmspeicher (BSp)
POSX     .EQU  ØØØA      ;Adresse des ersten Zeichens
                         ;des Strings im BSp.
INT      .EQU  ØØØC      ;Adressen der auszugebenden
                         ;Integerzahlen
LEADØ    .EQU  ØØ11      ;Hilfsregister
DIGIT    .EQU  ØØ12      ;Hilfsregister
MODUS    .EQU  ØØ13      ;Ausgabemodus: Ø=Normal  1=Invers
STELLEN  .EQU  ØØ15      ;Zahl der auszugebenden Stellen
RET      .EQU  ØØ17      ;RTS-Adresse

;********* Makro-Definitionen **********

         .MACRO POP
         PLA
         STA %1
         PLA
         STA %1+1
         .ENDM

         .MACRO PUSH
         LDA %1+1
         PHA
         LDA %1
         PHA
         .ENDM

         .MACRO MAL2
         ASL %1
         ROL %1+1
         .ENDM

         .MACRO AUSGABEMODUS

         ;Ausgabemodus festlegen

         LDA MODUS
         BEQ NORMAL

         ;Inverse Ausgabe

         LDA #29          ;Opcode von AND
         LDX #7F
         JMP LBL1

         ;Normale Ausgabe

NORMAL   LDA #Ø9          ;Opcode von ORA
         LDX #8Ø
LBL1     STA CHA          ;Normal: ORA #8Ø, Invers: AND #7F
         STX CHA+1
         .ENDM

;*********** Unterprogrammsammlung ***********

         .PROC SUBROUT1
         .DEF START

;Übernahme der Werte vom Stack und Berechnung der Bild-
;schirmkoordinaten

START    JMP PROG

;Tabelle der Adressen der ersten Stellen jeder der 24 Zeilen
;im BSp

COLØ     .WORD 4ØØ,48Ø,5ØØ,58Ø,6ØØ,68Ø,7ØØ,78Ø
         .WORD 428,4A8,528,5A8,628,6A8,728,7A8
         .WORD 45Ø,4DØ,55Ø,5DØ,65Ø,6DØ,75Ø,7DØ

;Werte vom Stack in die Hilfsregister übernehmen

PROG     POP RET          ;Rückkehradresse nach RTS
         POP RETURN       ;Rückkehradresse nach PASCAL
         POP ADRSTR       ;Adresse String/Char/Integerzahl
         PLA
         STA YKOOR        ;Y-Koordinate
         PLA              ;Highbyte der Y-Koordinate = Ø
         PLA
         STA XKOOR        ;X-Koordinate
         STA XKOORH
         PLA
         POP MODUS        ;Ausgabemodus übernehmen

;Koordinaten im BSp nach der Formel (POSØ) + X/2 bzw. (X-1)/2
;je nachdem, ob X gerade oder ungerade, berechnen

         LDA YKOOR        ;Ø<= Y-Koordinate <= 24
         ASL A            ;2x da 16 Bit-Adresse
         TAY
         LDA COLØ,Y       ;Adresse des ersten Zeichens der Zeile
         STA POSØ         ;im BSp nach POSØ/POSØ+1 setzen
         INY
         LDA COLØ,Y
         STA POSØ+1

         LDA XKOOR        ;jetzt X-Koordinate berücksichtigen
         LSR A            ;Ø:X=gerade, 1:X=ungerade
         BCC GERADE       ;verzweigt wenn X gerade
         DEC XKOOR        ;X-1 falls X ungerade
GERADE   LSR XKOOR        ;X halbieren
         CLC              ;addieren
         LDA POSØ
         ADC XKOOR
         STA POSX
         LDA POSØ+1
         ADC #ØØ          ;Highbyte von X stets Ø
         STA POSX+1       ;jetzt steht in POSX/POSX+1 die
                          ;Anfangsadresse des Strings im BSp
         PUSH RET         ;Rückkehradresse für RTS auf Stack
         RTS              ;schieben

         .PROC SUBROUT2
         .DEF PASCAL

;Wiederherstellung der alten Softswitches MAINRAM/AUXRAM
;und Rückkehr nach PASCAL

PASCAL   POP RET
         PLA              ;Speicherinhalt von STATUS

         ;Alte Schalterstellung von AUXRAM/MAINRAM
         ;wiederherstellen

         BMI PAGE2ON
         STA MAINRAM      ;PAGE1(MAINRAM) on=>enable MAINRAM
         JMP AUS
PAGE2ON  STA AUXRAM       ;PAGE2(MAINRAM) on=>enable AUXRAM
AUS      PUSH RETURN      ;zurück nach PASCAL
         RTS

;********** Ende der Unterprogramme ************************

;********** Prozedur PSTR(N,X,Y:INTEGER; CH:STRING); *******

         .PROC PSTR,4     ;4 Variablen zur Übergabe
         .REF START
         .REF PASCAL

         JSR START        ;Übernahme der Werte
         AUSGABEMODUS
         STA STORE8Ø      ;8Ø-Zeichen-Karte aktivieren
         LDA STATUS       ;Schalterstellung auf Stack retten,
         PHA              ;um nachher wieder die alte
                          ;Position herzustellen zu können
         STA AUXRAM       ;enable PAGE2(AUXRAM)
         LDA XKOORH
         LSR A            ;X auf Parität prüfen
         BCC GERADE
         STA MAINRAM      ;enable PAGE1(MAINRAM); X ungerade
GERADE   LDA ADRSTR+1
         BNE STRING
```

```
         ;genau ein Zeichen im String

         LDA #Ø1          ;ein Zeichen
         TAY              ;damit Y=1
         STA ZAHL
         LDX #ØØ          ;nur zur indirekten Adressierung
         LDA ADRSTR       ;auszugebendes Zeichen laden
         JMP CHA          ;Zeichen ausgeben

         ;mehr als ein Zeichen im String oder Leerstring

STRING   LDY #ØØ          ;zur indirekten Adressierung
         LDX #ØØ          ;X zählt die Position im BSp
         LDA (ADRSTR),Y   ;Zahl der auszugebenden Zeichen
         BEQ LEERSTR;     ja, String ist leer
         STA ZAHL
LOOP     INY              ;Y ist Zählindex, beginnt mit Y=1
         LDA (ADRSTR),Y   ;auszugebendes Zeichen laden

CHA      ORA #Ø8Ø         ;damit Ausgabe normal
         STA (POSX),X     ;Eintrag in BSp
         LDA STATUS       ;Stellung der Softwareschalter prüfen
         BMI PAGE1ON      ;minus=>PAGE2 war on => enable PAGE1
         STA AUXRAM       ;enable PAGE2 (AUXRAM)
         INC POSX         ;Pointer auf nächste Position setzen;
                          ;geschieht nur nach jedem zweiten
                          ;Zeichen; dazwischen wird AUXRAM/
                          ;MAINRAM umgeschaltet
         JMP OK
PAGE1ON  STA MAINRAM      ;enable PAGE1 (MAINRAM)
OK       CPY ZAHL         ;auszugebende Zeichenzahl erreicht?
         BNE LOOP         ;verzweigen, wenn ungleich
LEERSTR  JSR PASCAL
         RTS

;********* Prozedur PCHA(N,X,Y:INTEGER; CH:CHAR); *********

         .PROC PCHA,4     ;vier Werte zur Übernahme
         .REF START       ;funktioniert wie PSTR mit
         .REF PASCAL      ;Stringlänge 1

         JSR START
         AUSGABEMODUS

         STA STORE8Ø
         LDA STATUS
         PHA
         STA AUXRAM
         LDA XKOORH
         LSR A
         BCC GERADE
         STA MAINRAM
GERADE   LDX #ØØ
         LDA ADRSTR       ;Char laden
CHA      ORA #8Ø          ;Ausgabemodus festlegen
         STA (POSX),X     ;Char in BSp eintragen
         JSR PASCAL
         RTS

;********* Prozedur PINT(S,M,X,Y,INT:INTEGER); *********

         .PROC PINT,5     ;fünf Werte zur Übernahme
         .REF START
         .REF PASCAL

         ;Nach Übergabe in START befindet sich die auszu-
         ;gebende Zahl als Hexzahl in ADRSTR/ADRSTR+1.
         ;Zuerst werden die fünf auszugebenden Ziffern be-
         ;rechnet und in INT, ..... INT+4 gespeichert.

         JMP LBL2

         ;LSB der Zahlen 1,1Ø,1ØØ,1ØØØ,1ØØØØ

TLSB     .BYTE Ø1,ØA,64,ØE8,1Ø

         ;MSB der Zahlen 1,1Ø,1ØØ,1ØØØ,1ØØØØ

TMSB     .BYTE ØØ,ØØ,ØØ,Ø3,27

LBL2     JSR START
         POP STELLEN      ;Zahl der auszugebenden Stellen
         AUSGABEMODUS
         DEC STELLEN      ;Stellenzähler beginnt bei Ø

         ;SPACE laden und INT initialisieren

         LDA #Ø2Ø
         STA INT
         STA INT+1
         STA INT+2
         STA INT+3
         STA INT+4

         ;Berechnung der fünf auszugebenden Ziffern. Nach
         ;Randy Hyde, 'Assembly Language', S. 12-5

         LDX #Ø4
         STX LEADØ        ;positiver Wert in LEITØ
BEGIN    LDA #ØBØ         ;Zeichen 'Ø' laden
         STA DIGIT

ZEHN     SEC              ;Subtraktion vorbereiten
         LDA ADRSTR       ;LSB der auszugebenden Zahl
         SBC TLSB,X
         PHA
         LDA ADRSTR+1     ;MSB der auszugebenden Zahl
         SBC TMSB,X
         BCC LBL3         ;wenn Ziffer < TMSB+X, dann
                          ;Ziffer speichern
         STA ADRSTR+1
         PLA
         STA ADRSTR
         INC DIGIT        ;nächst größere Ziffer nehmen
         JMP ZEHN         ;neuer Versuch

LBL3     PLA              ;Stack ausgleichen
         LDA DIGIT        ;auszugebende Ziffer in Akku
         CPX #ØØ          ;wenn X=Ø, dann ist die Ziffer
                          ;an der letzten Stelle
         BEQ SPEICH       ;dann Ziffer speichern
         CMP #ØBØ         ;Akku mit  Ziffer Ø vergleichen
         BEQ TEST         ;wenn Ziffer ist Ø => verzweigen
         STA LEADØ
TEST     BIT LEADØ
         BPL NOTOUT       ;wenn LEITØ positiv (Bit Nr.7=Ø)=>
                          ;verzweigen und Ø nicht speichern
SPEICH   STA INT,X        ;Zehntausenderziffer in INT,...
                          ;Einerziffer in INT+4 speichern
NOTOUT   DEX              ;nächste Stelle
         BPL BEGIN

         ;Ausgabe der Zeichen in INT,...INT+4

         STA STORE8Ø
         LDA STATUS
         PHA
         STA AUXRAM       ;enable PAGE2(AUXRAM)
         LDA XKOORH
         LSR A
         BCC GERADE
         STA MAINRAM      ;enable PAGE1(MAINRAM)

GERADE   LDX #ØØ          ;zur indirekten Adressierung
         LDY STELLEN      ;Y ist Zählindex
LOOP     LDA INT,Y        ;Ziffer laden
CHA      ORA #Ø8Ø         ;Ausgabemodus festlegen
         STA (POSX),X     ;Ziffer in BSp eintragen
         LDA STATUS       ;PAGE2-Schalte prüfen
         BMI PAGE1ON      ;Page1 einschalten
         STA AUXRAM       ;enable PAGE2 (AUXRAM)
         INC POSX
         JMP OK
PAGE1ON  STA MAINRAM      ;enable PAGE1(MAINRAM)
OK       DEY              ;nächstes Zeichen
         BPL LOOP
         JSR PASCAL
         RTS
         .END
```

FASTPRINT.TEXT

```
{$S+}
UNIT FASTPRINT;  INTRINSIC CODE 25;
{Ø3.11.1984 Dr.W.Braun}

{Diese Pascal-Unit dient zur Einbindung der drei Ausgabe-
   routinen in die SYSTEM.LIBRARY auf der Diskette APPLE1:}

INTERFACE
PROCEDURE PSTR(M,X,Y:INTEGER; STR:STRING); {Zeichenketten}
PROCEDURE PCHA(M,X,Y:INTEGER; CHA:CHAR);   {Chars}
PROCEDURE PINT(S,M,X,Y,INT:INTEGER);       {ganze Zahlen}

IMPLEMENTATION
PROCEDURE PSTR; EXTERNAL;
PROCEDURE PCHA; EXTERNAL;
PROCEDURE PINT; EXTERNAL;
{kein Programmcode}
BEGIN {Keine Initialisierung}
END.
```

DEMOPRINT.TEXT

```
PROGRAM DEMOPRINT;
{Ø3.11.1984 Dr.W.Braun}

{Dieses Programm demonstriert den Geschwindigkeitsgewinn}
{bei Verwendung der Prozedur PSTR zur Ausgabe von Zeichen-}
{ketten im Vergleich zur Prozedur WRITELN}

VAR I,J:INTEGER;
    B:STRING;

PROCEDURE PSTR(M,X,Y:INTEGER; STR:STRING);EXTERNAL;
PROCEDURE PCHA(M,X,Y:INTEGER; CHA:CHAR);  EXTERNAL;
PROCEDURE PINT(S,M,X,Y,INT:INTEGER);      EXTERNAL;
PROCEDURE WARTE;
BEGIN
  FOR I:=1 TO 2ØØØ DO;
END;

BEGIN {Programmbeginn}
  B:='AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA';
  B:=CONCAT(B,B);

  {********* WRITELN-Prozedur *********}

  WRITELN(CHR(12),CHR(15)); {Bildschirm löschen, Invers}
  WARTE;
  GOTOXY(25,3); WRITE('PASCAL-Prozedur WRITE');
  WARTE; WRITE(CHR(14));            {Normal}
  GOTOXY(Ø,5);
  FOR I:=5 TO 22 DO WRITELN(B);

  {********* PSTR-Prozedur *********}

  WARTE; WRITE(CHR(12)); WARTE;
  PSTR(1,2Ø,3,'Prozedur PSTR(M,X,Y,STRING)');
  WARTE;
  FOR I:=5 TO 23 DO PSTR(Ø,Ø,I,B);
  WARTE;

END.
```

TIPS UND TRICKS IN PASCAL

Teil 7: Der externe Aufbau von Files

von Dieter Geiß

Bislang sind in der Tips-und-Tricks-Serie folgende Teile erschienen:
Teil 1: Die versteckte Prozedur „Idsearch", 8/85, S. 48ff.
Teil 2: Kann man den P-Code optimieren?, 9/85, S. 40ff.
Teil 3: Der P-Code-Cruncher für 128K-Pascal, 10/85, S. 50ff.
Teil 4: Die Compiler-Optionen, 11/85, S. 57ff.
Teil 5: Assembler-Ein/Ausgabe in UCSD-Pascal, 2/86, S. 47ff.
Teil 6: Der interne Aufbau von Files, 3/86, S. 48 ff.

Als Fortsetzung des letzten Teils wollen wir uns nun mit der Repräsentation von Files auf der Diskette beschäftigen. Zum Verständnis benötigen wir das Programm-Listing 1 des letzten Teils (**FILETEST**, s. 3/86, S. 54). Die Numerierung der neuen Listings soll deshalb an dieser Stelle mit 3 fortgesetzt werden. Wir starten das Programm FileTest, welches vier Diskettenfiles erzeugt. Der erste File (TEXTFILE.TEXT) ist ein Textfile, der 4 Blöcke umfaßt, wobei 5 Bytes im letzten Doppelblock benutzt sind, wie ein ausführliches Listing des Directory zeigt, auf dem sich die Files befinden. Der zweite File ist ebenfalls ein Textfile mit vier Blöcken, in dessen letztem Doppelblock 12 Bytes benutzt werden. Der dritte File besteht nur aus einem Block, in welchem erwartungsgemäß 4 Bytes belegt sind (ein Element „Typ"). Der vierte File kopiert den Codefile FILETEST.CODE in den File CODEFILE.CODE.

Im Prinzip gibt es sieben verschiedene Typen von Files (ohne Directory und Unterdirectory), die sich in der Definition des Typs „Filekind" verstecken. Nur zwei davon (Text- und Codefiles) müssen jedoch ein bestimmtes Format haben; die anderen sind benutzerabhängig.

Im Prinzip besteht jeder File aus einer Aneinanderreihung von Bytes. 512 Bytes ergeben einen Block. Wie diese Bytes vom System oder von einem Benutzer interpretiert werden, hängt zunächst einmal von dessen Filetyp ab, der sich meistens im Suffix (.TEXT, .CODE etc.) bemerkbar macht. Doch auch ein Suffix kann trügen. Es gelingt ohne große Mühe, im Filer mit Hilfe des Make-Kommandos einen Codefile zu „machen", der keinen regulären P- oder Assemblercode enthält. Man könnte diesen dann mit X(ecute zur Ausführung bringen, wobei das System mit größter Wahrscheinlichkeit abstürzen würde, da der Code „Schrott" enthält. Beim Öffnen eines Files („rewrite" oder „Make" im Filer) bekommt der geöffnete File den Filetyp des Suffixes, falls eines vorhanden ist, ansonsten den Typ „Data". So haben in unserem Listing 1 die vier Files die Typen Text, Text, Data, Code, weil die Namen der Files ein entsprechendes Suffix aufweisen. Doch zunächst zu den benutzerabhängigen Filetypen (s.a. 1/85, S. 65):

Ein **Xsdkfile** kann nicht gelesen werden, da es „Bad Blocks" aufweist. Ein solcher File kann mit Hilfe des Xamine-Befehls im Filer markiert werden.

Ein **Infofile** sollte irgendwelche Informationen enthalten. Es ist einem Programmierer freigestellt, einen File mit dem Suffix INFO zu erzeugen. Ob dieser File tatsächlich Informationen enthält, kommt auf den Programmierer an. Der Compiler benutzt den Infofile LINKER.INFO, um Informationen für den Linker kurzzeitig abzuspeichern.

Alle Files, die ohne Suffix erzeugt werden, sind Datafiles, die irgendwelche Daten enthalten. Ein Beispiel ist der File SYSTEM.MISCINFO, der z.B. Daten über den Aufbau des angeschlossenen Termi-

nals enthält. Der in Listing 1 erzeugte File DATAFILE.DATA wird gleich noch genauer untersucht werden.
Graffiles sind Files, die irgendwelche Grafiken enthalten sollen. Ich benutzte in meiner Implementierung der „TurtleGraphics" Graffiles, um den einfachen oder doppelt hochaufgelösten Grafikschirm abzuspeichern.
Fotofiles können synonym zu Graffiles verwendet werden.

1. Datafiles

Als erstes wollen wir den erzeugten Datafile DATAFILE.DATA unter die Lupe nehmen. Wie schon erwähnt, besteht er nur aus einem Block mit 4 Bytes, die in diesem Block eine sinnvolle Information darstellen. Das Programm schreibt mit Hilfe eines Put-Befehls einen Datensatz, der aus einem zweielementigen Record besteht, auf den File, der auf Diskette angelegt wurde. Die beiden Integer-Zahlen sind 513 und 1027. Die Zahlen wurden nicht zufällig so gewählt. Betrachtet man die Darstellungen von 513 und 1027 im Hexformat mit Low- und Highbyte, so ergibt sich die Darstellung

```
 513 = $Ø1 $Ø2
1Ø27 = $Ø3 $Ø4
```

Somit ergeben sich die vier Bytes, aus denen der File besteht, nämlich $01, $02, $03, und $04. Mit einem Disk-Lese-Programm könnte man dies nachprüfen. Der Rest des Blocks ist mit Nullen gefüllt. Die vier Bytes repräsentieren also genau ein Element des Files, dessen Typ Records mit 2 Integerzahlen zu je zwei Bytes enthält.

2. Textfiles

Ein mit einem Suffix TEXT erzeugter File hat immer mindestens die Größe von zwei Blöcken. Diese ersten zwei Blöcke enthalten Informationen für den Editor, mit dem der File geschrieben wurde. Falls der File vom Programm aus erzeugt wurde, wie das in unserem Beispiel der Fall ist, so werden diese zwei Blöcke mit Nullen gefüllt, d.h. es stehen keine Informationen für den Editor bereit. Je nach Art des Editors sind die beiden ersten Blöcke unterschiedlich aufgebaut. Es befinden sich dort auf jeden Fall die Positionen der Marker, das Datum des Files, Tabulatorpositionen usw. Wurde ein File mit dem normalen Apple-Editor geschrieben, so kann er zwar mit dem Advanced System Editor (ASE) eingelesen werden, aber die Editorinformationen des normalen Editors gehen verloren. Dasselbe gilt im umgekehrten Fall. Jeder Editor hat eine andere Datenstruktur für seine Informationen. Es bringt auch nichts, diese Infomationen zu ändern; deswegen möchte ich auch nicht näher auf sie eingehen.
Wird ein Textfile über eine Filevariable, welche einen Text- oder einen Interaktivfile repräsentiert, zum Lesen geöffnet, so werden automatisch die ersten zwei Blökke übersprungen, da die dort enthaltene Information nicht direkt zum Textfile gehört. Ein Textfile besteht immer aus Doppelblöcken, hat also z.B. die Länge 2, 4, 6 Blöcke usw. Auffallend ist auch, daß das Pascalsystem bei einem Zeilenvorschub, also einem
writeln (F)
prüft, ob das Ende eines Doppelblocks erreicht ist. Für das System ist es erreicht, wenn das nächste Byte, das nach dem End-Of-Line-Zeichen in den File geschrieben werden muß, näher als 128 Bytes vom letzten Byte des Doppelblocks entfernt ist, also noch weniger als 128 Bytes im Doppelblock Platz haben. Damit verschenkt das System beim folgenden
writeln (F)
genau 126 Bytes, da nach dem letzten CR noch eine Null in den File eingetragen wird. Jeder Doppelblock sollte mit mindestens einer Null beendet werden, sonst dreht der Compiler durch, weil er das Ende des Doppelblocks nicht mehr findet, da er nach der letzten Null von hinten sucht.
Der Inhalt von Textfiles besteht aus allen möglichen ASCII-Zeichen, von denen drei durch ihre besondere Bedeutung herausragen.
0 (ASCII NUL, „null") gibt das Ende eines Doppelblockes an. Nach einer Null kann keine Information mehr in diesem Doppelblock folgen.
13 (ASCII CR, „carriage return") gibt das Ende einer Zeile an.
16 (ASCII DLE, „data link escape") ist ein Spezialcode zur Komprimierung von Leerzeichen. Die der 16 folgende Zahl ist die um 32 erhöhte Anzahl von Leerzeichen (ASCII SP, „space"). Es ist nicht erlaubt, mehr als 127 Leerzeichen durch ein DLE zu komprimieren, sonst werden sie beim Lesen von diesem File ignoriert, so daß die gültigen Werte nach einem DLE von 32 bis 159 reichen. Die DLE-Komprimierung wird vor allen Dingen von den Editoren ausgenutzt. Reines Schreiben auf einen Textfile mit der gewöhnlichen Writeln-Anweisung erzeugt keine DLE-Komprimierung. Aus diesem Grund sind mit dem Editor erzeugte Textfiles fast ausnahmslos kürzer als per Programm erzeugte Textfiles.

3. Codefiles

Am weitaus kompliziertesten ist der Aufbau der Codefiles. Sie bestehen aus einem Kopf, der den ersten Block des Codefiles ausmacht und dem mehrere Blöcke Code und eventuell Linkerinformationen folgen können. Der Kopf, das sog. Segment-Dictionary, läßt sich am einfachsten durch eine Datenstruktur ausdrücken, wie sie in Listing 3 (**LISTCODEF**) gegeben ist (vgl. Apple Pascal Operating System Reference Manual, S. 266ff.). Das Programm in Listing 3 gibt alle Komponenten des Segment-Dictionary eines Codefiles aus, indem es Block 0 eines Codefiles oder einer Library einliest und die darin enthaltenen Daten als Datenstruktur „Segdic" interpretiert.

Hinweis für Anfänger:
1. Datei LISTCODEF.TEXT der Peeker-Sammeldisk #17 mit GETDOS von DOS-3.3-Sammeldisk #15 auf Pascal-Diskette konvertieren.
2. LISTCODEF.TEXT zu LISTCODEF.CODE compilieren und dann über X(ecute starten.
3. Es erscheint ein Menü – ähnlich dem Filer. Nunmehr Name des zu analysierenden Codefiles oder kurz L für Library usw. eingeben.

Ein Codefile kann maximal 16 Segmente beinhalten. Diese 16 Segmente sitzen in den 16 Slots und können mit Hilfe des Library-Programms auf der Diskette APPLE3 (Pascal 1.1) bzw. APPLE2 (Pascal 1.2) vertauscht werden. Mit Hilfe dieses Programms können auch neue Segmente in eine Library eingebracht werden, solange dort noch Slots frei sind. Jeder Array im „Segdic" hat 16 Einträge, welche die 16 Slots repräsentieren. Da alle Slots gleichwertig sind, beziehen sich die Erläuterungen jeweils auf ein Element eines Arrays.

Diskinfo besteht aus jeweils zwei Integer-Zahlen.

Diskaddr gibt die Adresse des Anfangs des Segments relativ zum Anfang des Codefiles in Blöcken an.

Codeleng gibt die Größe des Segments in Bytes an.

Segname ist der Name des Segments. Wird ein Programm ohne die erste Programmzeile
program...
geschrieben, so setzt der Compiler nur zur Übersetzungszeit den Hauptprogrammnamen PROGRAM ein. Der Segname ist jedoch leer (er besteht aus Leerzeichen).

Segkind gibt die Art des Segments an. Wir unterscheiden:
a) Linked: Das Segment ist ausführbar. Alle externen Referenzen wurden durch den Linker aufgelöst, wenn solche vorhanden waren.

b) Hostseg: Ein Hauptprogramm, das noch gelinkt werden muß, weil es entweder Assemblerroutinen oder reguläre Units benutzt. Der Linker muß die Referenzen noch auflösen.
c) Segproc: Eine Segment-Prozedur. Dieser Typ wird allerdings nicht benutzt. Eine Segment-Prozedur ist vom Typ Linked.
d) Unitseg: Eine normale Nicht-Intrinsic-Unit.
e) Seprtseg: Eine separat compiliertes oder assembliertes Segment. Da Units andere Typbezeichnungen haben, bleiben nur Assemblerroutinen übrig, die diesen Typ einnehmen können.
f) Unlinked_Intrinsic: Eine Intrinsic Unit, die Assemblerroutinen aufruft.
g) Linked_Intrinsic: Eine vollständig gelinkte Intrinsic Unit.
h) Dataseg: Ein Datensegment für eine Intrinsic Unit. Eine reguläre Unit hat kein Datensegment, da ihre Variablen als globale Variablen angesehen werden. Eine Intrinsic Unit hat ein Datensegment, wenn im Interface-Teil oder im Implementation-Teil Variablen erklärt wurden.

Textaddr gibt die Adresse des Interface-Textes einer Unit relativ zu Block 0 des Codefiles an. Interface-Text ist nicht doppelblockweise, sondern blockweise organisiert. Am Ende des Interface-Textes stehen meistens 8 Leerzeichen und dann ein abschließendes „E". Diese 8 Leerzeichen können an der zweiten und vierten Stelle von den Buchstaben P und L unterbrochen sein. Das P steht für PASCALIO und das L für LONGINTIO. Die Buchstaben treten genau dann auf, wenn eine Unit eine der beiden oder auch beide System-Units für Real-Ein/Ausgabe bzw. Long-Integer-Ein/Ausgabe benutzt. Dies ist ein Trick des Compilers, da im Interface nicht das
uses PascalIO, LongIntIO;
auftreten kann. Das kommt daher, daß diese Units vom System vordefiniert sind und implizit unter anderem durch die Befehle

```
seek, readln (R), writeln (R) bzw.
readln (L), writeln (L), str usw.
```

in ein Programm oder eine Unit eingeführt werden. Bei den eben erwähnten Befehlen ist R eine Real- und L eine Long-Integer-Variable.

Seginfo enthält verschiedene Informationen zu einem Segment, die sich in einem „packed record" des Typs „Seginfrec" verstecken.
Segnum gibt die Nummer des Segments an, die von 0 bis 31 (Pascal 1.1, Pascal 1.2 64K) oder von 0 bis 63 (Pascal 1.2 128K) reichen kann.
Mtype reicht von 0 bis 9 und gibt die Art des Maschinencodes an. Dabei bedeuten
0: Unbekannter Code
1: P-Code, wobei das signifikanteste Byte zuerst kommt (Positive Byte Sex)
2: P-Code, wobei das am wenigsten signifikante Byte zuerst kommt (Negative Byte Sex). Dieser Code wird beim Apple benutzt.
3 bis 9: Maschinencode, der von einem Assembler erzeugt wurde. Die Zahl 7 repräsentiert 6502-Code.
Das nächste Bit ist unbenutzt.

Version gibt die Version des Codes an. Für sie gilt:
1: Pascal 1.0
2: Pascal 1.1
5: Pascal 1.2
Usedintrins ist eine Menge von Intrinsic Units, die ein Programm benutzt. Die Menge reicht normalerweise von 0 bis 31, bei Pascal 1.2 128K jedoch von 0 bis 63, da dort 64 Segmente benutzt werden können. Ist z.B. Bit 22 in dieser Menge gesetzt, so braucht das Programm die Unit mit der Segmentnummer 22, welche normalerweise von der Unit Applestuff okkupiert wird.

Danach folgen 68 oder 70 Words, die keine Bedeutung haben.

Comment ist eine Zeichenkette mit maximal 79 Buchstaben, die den Kommentar enthält, den man mit Hilfe der Compiler-Option (*$C...*) in einen Codefile einbinden kann.

Ab Block 1 folgen nun entweder ein Interface-Text oder schon der Code selbst. Bevor auf ihn näher eingegangen wird, soll kurz noch auf die dem Code folgenden Blöcke eingegangen werden.
Ein Hauptprogramm oder eine Intrinsic Unit, die Assemblerprozeduren als „external" erklärt haben, oder eine reguläre Unit benötigen Informationen, damit der Linker die Referenzen auflösen kann. Sie wird vom Compiler generiert und direkt hinter den Code des jeweils erzeugten Segments geschrieben. Mit Hilfe der Datentypen LiTypes, LiEntry, Alpha, Opformat, LCrange, ICrange und Procrange kann diese Information interpretiert werden. Dazu sind jedoch noch einige Tricks nötig, die man mit Hilfe sich überlagernder varianter Records anwenden kann. Es würde in diesem Artikel zu weit führen, auf die Linkerinformation näher einzugehen, da eine detaillierte Beschreibung des Teils des Compilers, der die Information schreibt, erfolgen müßte. Unter Zuhilfenahme des Programms Libmap auf der APPLE3- bzw. APPLE2-Diskette kann die Linkerinformation jedoch abgerufen und sichtbar gemacht werden.
Nun zum eigentlichen Aufbau des Codes eines Codefiles. Wir unterscheiden zwischen P-Code und Assemblercode.

3.1. P-Code

Ein Charakteristikum des P-Codes ist es, daß er relokatibel ist, d.h. er ist überall im Speicher ablauffähig. Daraus folgt, daß in ihm keine absoluten Adressen auftreten können. Vielmehr gibt es eine große Anzahl von Zeigern, die auf den Anfang und das Ende einer Prozedur zeigen, und vieles mehr. Alle Zeiger sind relativ zu sehen, d.h. relativ zu der Stelle, an der der Zeiger steht.

Das Segment-Dictionary wurde schon oben besprochen. Mit Hilfe der beiden Variablen im Diskinfo kann der Beginn und die Länge eines Codesegmentes festgestellt werden. Der Code muß dabei von hinten (hohe Adressen) angesehen werden. In **Bild 1** ist der Code des Hauptprogramms aus Listing 4 (**LEVEL0**) dargestellt, welches zwei Prozeduren beinhaltet. Insgesamt zählt das Segment drei Prozeduren, da das Hauptprogramm ebenfalls als Prozedur gilt. Als oberstes erkennt man das sog. „Procedure Dictionary", welches Zeiger auf alle im Segment enthaltenen Prozeduren aufweist. Prozedurnummern werden vom Compiler von 1 in aufsteigender Reihenfolge bis höchstens 149 vergeben. Eine Prozedur mit der Nummer 0 existiert nicht, weswegen die beiden Bytes für Prozedur 0 benutzt werden, um die Segmentnummer (niederwertiges Byte) und die Anzahl der Prozeduren im Segment (höherwertiges Byte) anzuzeigen.

Die Zeiger auf die Prozeduren zeigen jeweils auf die „Procedure Code Section".
Sie besteht zum einen aus der „Table of Attributes" und zum anderen aus dem Code selbst. Die „Table of Attributes" besteht mindestens aus 5 Words. Das erste Word (von oben) ist in zwei Bytes unterteilt, welche die Prozedurnummer (niederwertiges Byte) und die lexikalische Verschachtelungstiefe (höherwertiges Byte) angeben. Für die Verschachtelungstiefe (Lex Level) gelten folgende Werte:
Pascalsystem = -1
Hauptprogramm oder Prozeduren des Pascalsystems = 0
Prozeduren im Hauptprogramm = 1
Prozeduren obiger Prozeduren = 2 usw.
Prozeduren mit Verschachtelungstiefe -1 oder 0 werden auch „Base" Prozeduren genannt, weil bei ihrem Aufruf das BASE-Register gerettet wird. Mit Hilfe des BASE-Registers werden globale Variablen adressiert. Prozeduren mit Verschachtelungstiefe 1 werden globale Prozeduren genannt, weil deren Markstack auf die letzte „Base Prozedur" zeigt. Prozeduren, deren Verschachtelungstiefe um eins größer ist als die sie aufrufende Prozedur, sind lokale Prozeduren. Ein Aufruf einer globa-

INTUS-Lern- und Anwenderprogramme

für Apple IIe/IIc - Computer

- Maschinenschreiben wie der Blitz DM 175,–
- Business-English, 2800 Begriffe, 16 Bereiche, 3 Übungsarten DM 120,–
- Rechtschreibtrainer, 4000 Wörter, 16 Bereiche, 4 Übungsarten DM 120,–
- Basic-Lernprogramm, ausgezeichnet DM 238,–
- Kinderschule, für Vorschulkinder DM 49,–
- Rechenmodelle für AppleWorks DM 195,–
- MailWorks für AppleWorks-Serienbriefe, für alle Drucker geeignet DM 168,–
- AppleGraph, Erstellen von Balken- und Kreis-Graphiken DM 120,–
- PriBu-Privatbuchhaltung DM 195,–
- DMP-Charger zur Gestaltung eigener Zeichensätze auf dem Matrix-Drucker DM 198,–
- Gesamtkatalog mit über 200 Programmen gratis
- **Demo-Diskette gratis** gegen Angabe/Einsendung dieses Inserates
- 6000 Frei-Programme (fast) gratis (Shareware/Public domain) Liste DM 10,–

INTUS SOFTWARE

Kaiserstr. 21, 7890 Waldshut,
Tel. 0 77 51-79 20

Jetzt für Applesoft FP Basic:

LOGIC MACHINE knackt Programme

- findet tote („abgehängte") Programmzeilen (ab einer Einsprungzeile, die einzugeben ist, wahlweise mit und ohne GOSUB-tracing)
- trennt selbständig Mainline von Subroutinen in Listings
- trennt Subroutinen optisch voneinander in Listings
- listet Variablen cross-references
- listet Zeilen cross-references
- findet alle Referenzen einer bestimmten Zeile, Variable
- findet alle Vorkommnisse eines Tokens (GOSUB, ONERR etc.)
- automatischer Zeilengenerator
- RENUMBER, MERGE u.a.

Hilft
- Listings von fremden Programmen sofort verstehen
- Subroutinen isolieren und nahtlos herauslösen (Dummy Subroutinen einrichten, zwecks Test u.a.)
- transparent und logisch in BASIC programmieren
- beim Überarbeiten von Programmen

Einfache &-Befehle. Sortierte Ausgabe. Schnell. 100 % Maschinensprache. Lauffähig unter DOS 3.3 (auch nach Verschiebung in die Language Card) und (fast alles) ProDOS, Apple II+, e, c. Diskette in DOS 3.3 Format, Dokumentation: DM 30,– Vorauskasse (bar/Scheck).

Real Soft Dorstener Str. 67, 4660 Gelsenkirchen-Buer

ZUSATZ-KARTEN:

V-24-Schnittstelle	199,-	**Z-80-Karte**	98,-
80-Zeichen-Karte m.Softswitch	236,-	**16 K-Language-Karte**	98,-
Joy Stick De Luxe	59,-	**Accelerator** 3,6 MHz	950,-
68000 Intemex	1600,-	**PAL Karte**	110,-
RGB Karte	239,-	**IEEE 488**	312,-
Koppler dataphon m. FTZ	325,-	**Z 80 B Karte** mit Software	919,-
Centronics-Karte von Epson	für Graphik 210,-	für Text	145,-
Centronics-Schnittstelle für 2 Drucker gleichzeitig			129,-

Super-Eprommer belegt keinen Slot, incl. Software für 2716-27128 ..... **239,-**

Floppy-Controller

FDC 4 für alle Laufwerke ..... 169,- Bausatz wie links 159,-
Leerplatine wie oben incl. Prom u. Eprom 98,-

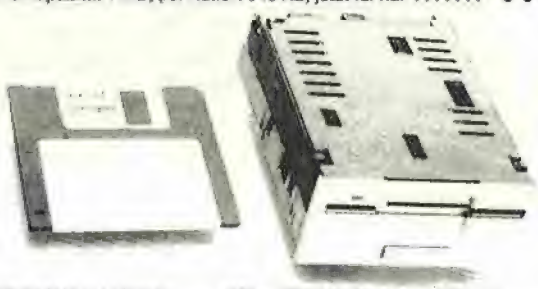
Erphi-Controller **298,-**

Disketten 1D, 48 tpi ..... 10 St. **29,-**
Disketten 2D, 48 tpi ..... 10 St. **36,-**

Preh Commander Keyboards

Wir bieten Ihnen die **Preh-Qualität** auch für Apple. AK 88 spez. mit Gehäuse, Anschlußkabel, Zehner-Tastenfeld, dt. Zeichensatz, Sondertasten für Ctrl-Codes und Rechenfunktionen **339,-**
Preh Commander Keyboard, frei programmierbar bis zu 10 Ebenen, pro Taste bis zu 250 Zeichen ..... nur **599,-**
Gleiche Tastatur wie oben **für Apple IIe** ..... nur **698,-**

TEAC 3½" Laufwerk FD 35 F Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur **498,-**



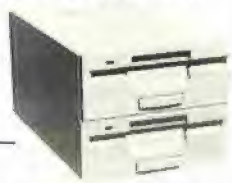
TEAC FD 55 AV 1 x 40 Track	395,-	**TEAC FD 55 BV** 2 x 40 Track	460,-
TEAC FD 55 EV 1 x 80 Track	445,-	**TEAC FD 55 FV** 2 x 80 Track	398,-

Apple®-kompatibles Laufwerk incl. Gehäuse + Kabel 599,-
320 KB Laufwerk für IIc **948,-**
640 KB Laufwerk für IIc **1088,-**

Panasonic Drucker:
1090 ..... nur **849,-**
1091 ..... nur **1095,-**
1092 ..... nur **1295,-**

Die Microfloppy mit Zukunft:

Speicherkapazität: 2 x 1 MByte formatiert: 2 x 640 kByte. Anschlußfertig mit PROM-residenter Patchsoftware für CP/M 2.2, Apple DOS 3.3, DiversiDOS 2-C, 4-C (DD MOVER), Apple Pascal 1.1, Pascal 1.2, Pro-DOS 1.0.1, 1.1, 1.1.1 zum Preis von **1498,-**
Low Power Version 1598,-



10 MB Winchester mit Software für DOS 3.3, CP/M 2.20, Pascal, Pro-DOS, incl. Controler und Gehäuse **3980,-**

Sonderangebot

Distar Laufwerk für II + IIe, incl. Kabel u. Gehäuse ..... jetzt nur **349,-**

Gesamt-Preisliste anfordern! Preise inclusive gesetzlicher Mehrwertsteuer.
Händlerpreisliste bitte schriftlich anfordern!

UEDING electronics

Holtewiese 2
5750 Menden 1

DFÜ 02373/66877
Tel. 02373/63159

len Prozedur vom Hauptprogramm führt allerdings nicht zur Erzeugung eines globalen, sondern eines lokalen Aufrufs, da globale Prozeduren lokal zum Hauptprogramm sind. Andere Relationen der Verschachtelungstiefe führen zu „Intermediate" Prozeduren.
Dann folgen der „Enter IC", der auf das erste Byte des Codes zeigt, der bei Aufruf der Prozedur ausgeführt werden muß. Der „Exit IC" zeigt auf den ersten von denjenigen Befehlen, die noch ausgeführt werden müssen, bevor die Prozedur verlassen werden kann. Diese können Instruktionen zum Schließen von Files oder zur Freigabe von Segmenten sein. „Parameter Size" gibt die Größe der an die Prozedur übergebenen Parameter an. „Data Size" ist die Größe der lokalen Variablen, beim Hauptprogramm die der globalen Variablen. Die folgende optionale „Jump Table" enthält Zeiger auf Stellen im Prozedur-Code. Die „Jump Table" wird benötigt, weil der P-Code keine Rückwärtssprünge kennt. Jeder Rückwärtssprung zeigt auf einen Eintrag in der „Jump Table", welcher wiederum auf die Stelle im Code zeigt, die angesprungen werden muß. Auch bei Vorwärtssprüngen, die in einem Byte nicht unterzubringen sind (weiter als 127 Bytes vorwärts), wird der Umweg über die „Jump Table" gewählt (s.a. Apple Pascal Operating System Reference Manual, S. 239 über „Jumps").

3.2. Assembler-Code

Das Format der „Table of Attributes" ist sehr unterschiedlich, vergleicht man es mit dem des P-Code. Zum besseren Verständnis sollte man sich **Bild 2** ansehen, in welchem der Code der Assemblerprozedur aus Listing 5 (**ASSEM**) graphisch dargestellt ist. Schon das erste Word von oben, welches wieder in zwei Bytes aufgespaltet ist, gibt eine andere Information wieder als bei P-Code-Prozeduren. Dort war es die Verschachtelungstiefe und die Prozedurnummer. Im niederwertigen Byte ist an die Stelle der Prozedurnummer eine 0 getreten, damit der Interpreter beim Laden eines Segments weiß, daß es sich um eine Maschinenprozedur handelt. Dieser Umstand ist wichtig, da nicht jeder Code eines bestimmten Prozessors relokativen Code erzeugen kann. Da Maschinenprozeduren aber mit P-Code gemischt auftreten können, muß auch jede solche Prozedur überall im Speicher ablauffähig sein. Der Interpreter muß dann zur Ladezeit eines Segments alle Referenzen auflösen, nachdem er weiß, an welcher Stelle im Speicher der Maschinencode sitzen wird.
Da Maschinenprozeduren nur in der Verschachtelungstiefe 1 auftreten können, d.h. als globale Prozeduren, ist auch diese Information nicht nötig. Statt dessen muß dem Interpreter bekannt sein, worauf sich die „BASE RELATIVE"-Adressierung bezieht. Befindet sich eine 0 im höherwertigen Byte des ersten Word, dann bezieht sich die „BASE RELATIVE"-Adressierung auf ein Programm oder eine Prozedur mit Verschachtelungstiefe 0 (gewöhnlicherweise das Hauptprogramm). Ist die Zahl ungleich 0, so bezeichnet sie die Segmentnummer einer Intrinsic Unit. Die „BASE RELATIVE"-Adressierung wird an entsprechender Stelle weiter unten erklärt.

Als zweites Word ist wie im P-Code der Enter IC vorhanden, der auf die erste Anweisung zeigt, die ausgeführt werden muß, wenn die Prozedur aufgerufen wird.

Nun folgen vier Relokativtabellen, welche alle das gleiche Format haben und die vom Interpreter benutzt werden, um den Maschinencode an der Stelle zum Laufen zu bringen, an der er geladen wurde. Als erstes taucht eine normale Integerzahl auf, die auch 0 sein kann. Sie gibt an, wieviele Einträge sich in der Tabelle befinden. In unserem Beispiel sind dies 2, 1, 3 und 2 für die vier Tabellen. Die vier Tabellen werden (von oben nach unten) BASE RELATIVE, SEGMENT RELATIVE, SELF RELATIVE und INTERPRETER RELATIVE TABLES genannt. In jeder befindet sich eine gewisse Anzahl von Zeigern, die auf den Teil des Maschinencodes zeigen, der zur Ladezeit noch auf den richtigen Stand gebracht werden muß.

Die BASE RELATIVE TABLE führt die vier Tabellen an. Globale Variablen werden über das BASE-Register der P-Maschine adressiert. Von einem Maschinenprogramm können diese mit den PUBLIC- und PRIVATE-Konstrukten erreicht werden. Ist das Maschinenprogramm in eine Intrinsic Unit eingebunden, so werden statt dessen die Variablen in deren Data Segment adressiert. In unserem Beispiel sind zwei Konstrukte (PUBLIC und PRIVATE) aufgetaucht, die Einträge in die BASE RELATIVE TABLE erzwingen.

Als nächstes folgt die SEGMENT RELATIVE TABLE, in welcher Einträge gemacht werden, wenn man eine Konstruktion mit REFs verwendet. Solche werden relativ zum Segmentanfang adressiert.

Die SELF RELATIVE TABLE benutzt den Anfang der Prozedur als Relokativinformation. In ihm befinden sich u.a. sämtliche Labels, die zur Prozedur gehören, und die darin definierten DEF-Konstruktionen. In unserem Beispiel befinden sich drei Einträge in dieser Tabelle.

Als letztes folgt die INTERPRETER RELATIVE TABLE. Sie funktioniert aber nur bei Pascal 1.2 richtig, weshalb sich alle folgenden Informationen nur auf Pascal 1.2 beziehen. Im Apple Pascal Operating System Reference Manual wird auf Seite 162 die Assembler-Direktive INTERP erklärt. Mit ihr können Stellen adressiert werden, die sich relativ zum Interpreter befinden. Drei Adressen ragen bei der Adressierung
.INTERP+n
besonders heraus:
n = 0: Die Execution Error Routine, die eine Fehlermeldung als Akkumulator-Wert ausgibt. Beispiel:

```
LDA #06 ;Division by zero
JMP @.INTERP ;$D1FB oder $D210
```

n = 2: Die Adresse der BIOS Jump Table. Beispiel:

```
JMP @.INTERP+2 ;Console Read
STA 00
```

n = 4: Die Adresse der SYSCOM (System Communication Area).

```
LDY #00
LDA (.INTERP+4),Y ;Letzter IORESULT
STA 00
```

Ein INTERP wird zunächst beim Laden als absoluter Wert ersetzt. Dabei gilt für diesen Wert:
Pascal 1.2: 064K: .INTERP = $D1AF
Pascal 1.2: 128K: .INTERP = $D1B8
Ab $D1AF bzw. $D1B8 liegen dann die sechs Bytes, die durch den obigen Ausdruck für n = 0, 1, 2, 3, 4 und 5 zu erhalten sind. Diese 6 Bytes sind als 3 Adressen zu interpretieren, die wiederum auf die drei erwähnten Stellen zeigen. Für Pascal 1.2 64K sind dies $D1FB, $FF00 und $BDDE und für Pascal 1.2 128K sind es $D210, $FF00 und $BD1E. Ein weiteres Beispiel für die INTERP-Direktive liegt in der Unit LONGINTIO vor. Dort wird von Maschinensprache aus die Exec-Error-Routine aufgerufen, wenn beispielsweise ein (Long) Integer Overflow auftritt. Da ein INTERP Befehl bei Pascal 1.1 nicht funktioniert, hängt das System bei einem Überlauf oder stürzt einfach ab.

Das nächste Mal soll eine Utility-Unit vorgestellt werden, die einige der bislang in dieser Serie behandelten Tricks benutzt.

Kurzhinweise

1. Zweck:
Analyse der externen Struktur von Files.
2. Konfiguration:
II+/e/c, UCSD-Pascal 1.1/1.2
3. Test:
X(ecute LISTCODEF.CODE
(nach Konvertierung und Compilierung)
4. Sammeldisk:
LISTCODEF.TEXT
LEVEL0.TEXT
ASSEM.TEXT
Diese Textfiles müssen zunächst mit GETDOS (s. Sammeldisk #15) auf Ihre Pascal-Arbeitsdiskette konvertiert werden.

Listing 3: LISTCODEF.TEXT

```
{$I-}
{$R-}
program ListCodefile (input, output);

const Maxslot   =  15;
      Maxseg    =  63;
      VT        =  11;
      GS        =  29;

type  Alpha     = packed array [Ø..7] of char;
      Segrange  = Ø..Maxseg;
      Slotrange = Ø..Maxslot;
      Segset    = set of Segrange;
      Segdesc   = record
                    Diskaddr : integer;
                    Codeleng : integer
                  end; {Segdes}
      Segkinds  = (Linked, Hostseg, Segproc, Unitseg, Seprtseg,
                   Unlinked_Intrinsic, Linked_Intrinsic,
                   Dataseg);
      Seginfrec = packed record
                    Segnum  : Ø..255;
                    Mtype   : Ø..15;
                    unused  : Ø..1;
                    Version : Ø..7
                  end; {Seginfrec}
      Segdic    = record
                    Diskinfo    : array [Slotrange] of Segdesc;
                    Segname     : array [Slotrange] of Alpha;
                    Segkind     : array [Slotrange] of Segkinds;
                    Textaddr    : array [Slotrange] of integer;
                    Seginfo     : array0[Slotrange] of Seginfrec;
                    Usedintrins : Segset;
                    Filler      : array [Ø..67] of integer;
                    Comment     : string [79]
                  end; {Segdic}

var   BEL       : char;
      IO        : integer;
      DoWrite   : boolean;
      Indic     : Segdic;
      Infile    : file;
      Inname    : string;
      Libdic    : Segdic;
      Libfile   : file;
      Libname   : string;
      Prompt    : string;

{-------------------------------------------------------------}

procedure Clearscreen;
begin {Clearscreen}
  page (output)
end; {Clearscreen}

{-------------------------------------------------------------}

procedure Cleop;
begin {Cleop}
  write (chr (VT))
end; {Cleop}

{-------------------------------------------------------------}

procedure Cleol;
begin {Cleol}
  write (chr (GS))
end; {Cleol}

{-------------------------------------------------------------}

procedure InitDic (var Dic : Segdic);
begin {InitDic}
  fillchar (Dic, size_of (Dic), Ø);
  with Dic do
  fillchar (Segname, size_of (Segname), ' ')
end; {InitDic}

{-------------------------------------------------------------}

procedure Init;
begin {Init}
  BEL    := chr (7);
  Prompt := 'Kommandos: C(lear, N(ew file, P(ascal, L(ibrary,
   S(ystem.wrk.code, Q(uit? '  {Bitte in eine Zeile}
end; {Init}

{-------------------------------------------------------------}

procedure ReadUpper (var C : char);
begin {ReadUpper}
  read (C);
  if C in ['a'..'z'] then C := chr (ord (C) - 32)
end; {ReadUpper}

{-------------------------------------------------------------}

procedure WriteIO;
begin {WriteIO}
  gotoxy (Ø, 23);
  Cleol;
  write (BEL, 'I/O Error #', IO);
end; {WriteIO}

{-------------------------------------------------------------}

procedure BlockØError;
begin {BlockØError}
  gotoxy (Ø, 23);
  Cleol;
  write (BEL, 'Error in reading block #Ø')
end; {BlockØError}

{-------------------------------------------------------------}

procedure WriteSlot (Dic : Segdic; Slot : integer; C : char);
var Nr : integer;
    X  : integer;
    Y  : integer;
begin {WriteSlot}
  with Dic, Diskinfo [Slot], Seginfo [Slot] do
  if (Diskaddr + Codeleng <> Ø) or
     (Segkind [Slot] in [Hostseg..Dataseg]) or
     (Segname [Slot] <> '         ') then
  begin
    if Version < 1
    then Nr := Slot
    else Nr := Segnum;
    if Nr in [Ø..15, 32..47]
    then X :=  Ø
    else X := 4Ø;
    Y := Nr mod (Maxslot + 1);
    gotoxy (X, Y + 6);
    write (C);
    write (Nr : 3, '- ');
    write (Segname [Slot]);
    write (Diskaddr : 4);
    write (Codeleng : 7, ' ');
    case Segkind [Slot] of
      Linked             : write ('Lnkd');
      Hostseg            : write ('Host');
      Segproc            : write ('Segp');
      Unitseg            : write ('Unit');
      Seprtseg           : write ('Sprt');
      Unlinked_Intrinsic : write ('UInt');
      Linked_Intrinsic   : write ('LInt');
      Dataseg            : write ('Data')
    end; {case}
    write (Textaddr [Slot] : 5);
    write (Mtype : 2);
    write (Version : 2)
  end {if, with}
end; {WriteSlot}

{-------------------------------------------------------------}

procedure PrintEmpty;
var I : integer;
    X : integer;
    Y : integer;
    S : string;
begin {PrintEmpty}
  gotoxy (19, 2);
  Cleop;
  S := ' Segm Segname  Add Length Kind Text M V';
  I := length (S);
  gotoxy (Ø, 4);
  unitwrite (1, S [1], I);
  gotoxy (4Ø, 4);
  unitwrite (1, S [1], I);
  for I := Ø to 31 do   {nur 32 Segmente}
  begin
    if I <= 15
    then X :=  Ø
    else X := 4Ø;
    Y      := I mod (Maxslot + 1);
    gotoxy (X, Y + 6);
```

```
    write (I : 4, '-')
  end {for}
end; {PrintEmpty}

{-------------------------------------------------------------}

procedure PrintCode;
var I, X, Y : integer;
begin {PrintCode}
  for I := Ø to Maxslot do WriteSlot (Indic, I, ' ')
end; {PrintCode}

{-------------------------------------------------------------}

procedure PrintLibrary;
var I : integer;
begin {PrintLibrary}
  if Indic.Usedintrins <> [] then
  begin
    Libname := '*SYSTEM.LIBRARY';
    reset (Libfile, Libname);
    IO := IO_Result;
    if IO = Ø
    then
      if blockread (Libfile, Libdic, 1) = 1
      then
        begin
          for I := Ø to Maxslot do
          if Libdic.Seginfo [I].Segnum in Indic.Usedintrins
          then WriteSlot (Libdic, I, '*')
        end {if}
      else BlockØError
    else
      begin
        gotoxy (Ø, 23);
        Cleol;
        write (BEL, 'Can''t find ', Libname);
      end; {else}
    close (Libfile)
  end {if}
end; {PrintLibrary}

{-------------------------------------------------------------}

procedure PrintSpecial (Name : string);
begin {PrintSpecial}
  close (Infile);
  Inname := Name;
  gotoxy (19, 2);
  Cleol;
  write (Inname);
  reset (Infile, Inname);
  IO := IO_Result;
  if IO <> Ø
  then WriteIO
  else
    if blockread (Infile, Indic, 1) = 1
    then
      begin
        PrintCode;
        PrintLibrary
      end {if}
    else BlockØError
end; {PrintSpecial}

{-------------------------------------------------------------}

procedure AskIn;
begin {AskIn}
  close (Infile);
  InitDic (Indic);
  repeat
    gotoxy (19, 2);
    Cleol;
    readln (Inname);
    if Inname = '' then exit (AskIn);
    DoWrite := false;
    if Inname = '*' then
    begin
      DoWrite := true;
      Inname  := '*SYSTEM.LIBRARY'
    end; {if}
    if Inname [length (Inname)] = '*' then
    begin
      DoWrite := true;
      insert ('SYSTEM.LIBRARY', Inname, length (Inname));
      delete (Inname, length (Inname), 1)
    end; {if}
    if DoWrite then
    begin
      gotoxy (19, 2);
      write (Inname)
    end; {if}
    reset (Infile, Inname);
    IO := IO_Result;
    if IO <> Ø then
    begin
      Inname := concat (Inname, '.CODE');
      reset (Infile, Inname);
      IO := IO_Result
    end; {if}
    if IO <> Ø then WriteIO
  until IO = Ø;
  if blockread (Infile, Indic, 1) = 1
  then
    begin
      PrintCode;
      PrintLibrary
    end {if}
  else BlockØError
end; {AskIn}

{-------------------------------------------------------------}

procedure List;
var C : char;
    S : string;
    X : integer;
begin {List}
  Clearscreen;
  InitDic (Indic);
  X := length (Prompt);
  write (Prompt);
  gotoxy (Ø, 2);
  write ('Input code file ->');
  PrintEmpty;
  repeat
    gotoxy (X, Ø);
    Cleol;
    ReadUpper (C);
    gotoxy (Ø, 23);
    Cleol;
    case C of
      'C' : begin
              PrintEmpty;
              AskIn
            end; {'C'}
      'N' : AskIn;
      'P' : PrintSpecial ('*SYSTEM.PASCAL');
      'L' : PrintSpecial ('*SYSTEM.LIBRARY');
      'S' : PrintSpecial ('*SYSTEM.WRK.CODE');
      'Q' : begin
              close (Infile);
              exit (List)
            end {'Q'}
    end {case}
  until false
end; {List}

{-------------------------------------------------------------}

begin {ListCodefile}
  Init;
  List
end {ListCodefile}.
```

Listing 4: LEVEL0.TEXT

```
program Level0;

procedure Level1;

  procedure Level2;
  begin {Level2}
    Level1; {Aufruf einer globalen Prozedur}
    repeat  {Eintrag in Jump Table}
    until false
  end; {Level2}

begin {Level1}
  Level2 {Aufruf einer lokalen Prozedur}
end; {Level1}

begin {PcodeBeispiel}
  Level1 {Aufruf einer globalen Prozedur,}
         {die lokal zum Hauptprogramm ist}
end {PcodeBeispiel}.
```

Listing 5: ASSEM.TEXT

```
                .PROC ASSEM_BEISPIEL

                .PUBLIC PUBL1
                .PRIVATE PRIV1
                .REF REF1
                .DEF DEF1

                LDA PUBL1
                STA PRIV1
                LDA REF1
                STA DEF1
                LDA LABEL1
                STA LABEL2
                LDA .INTERP
                STA 00
                LDA .INTERP+1
                STA 01
                JMP @00
                RTS

DEF1            .BYTE
LABEL1          .BYTE
LABEL2          .BYTE

                .END
```

Bild 1: Codesegment zu Listing 4

Alle Zahlen sind in hexadezimaler Darstellung

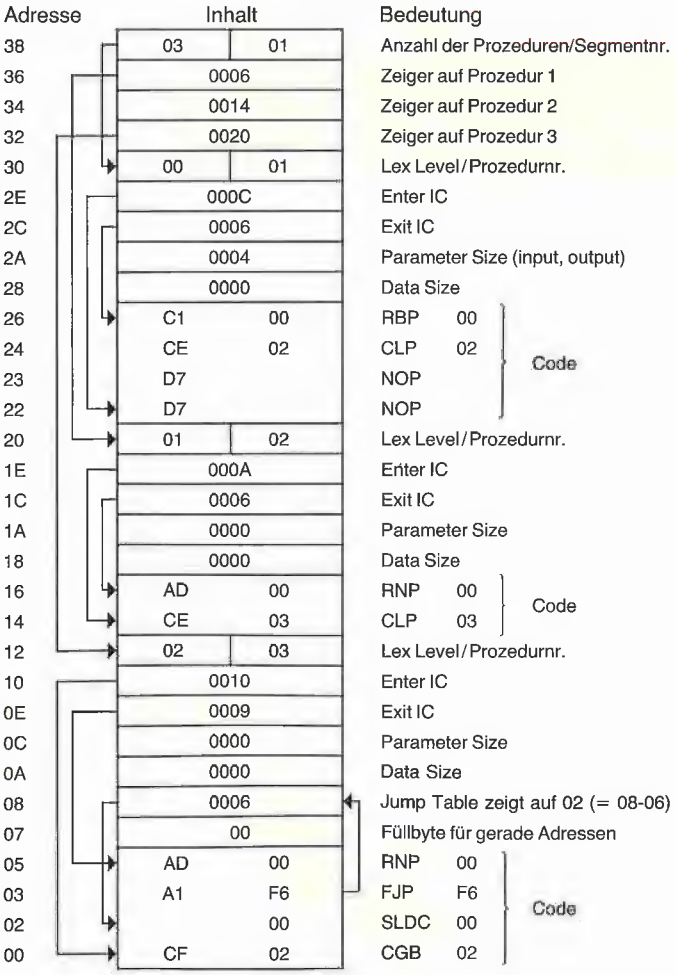
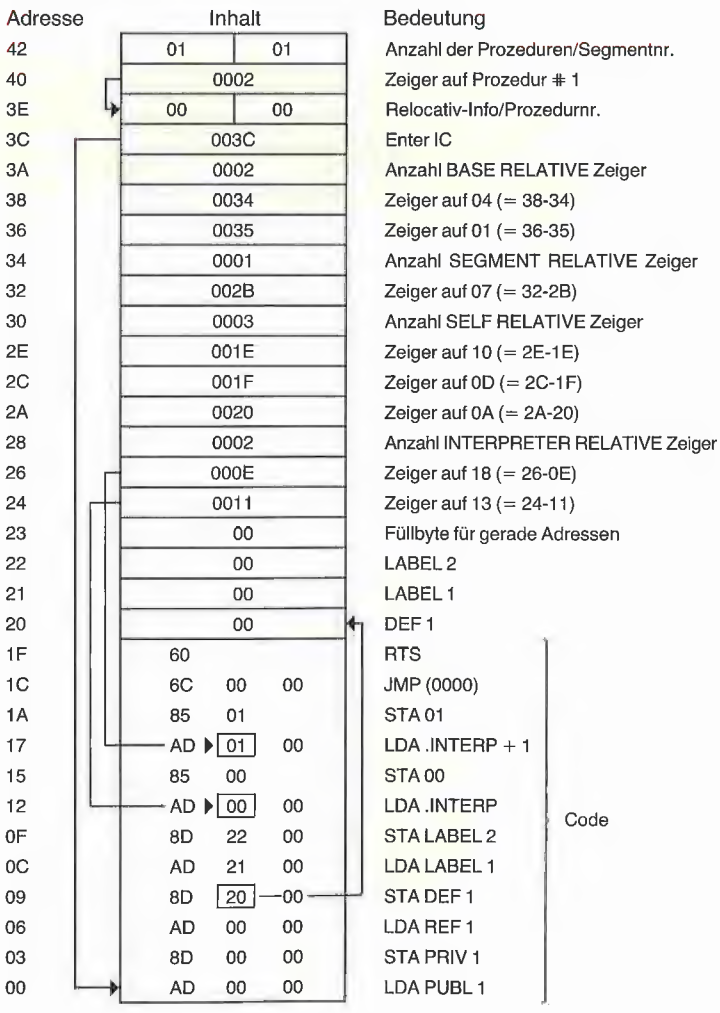


Bild 2: Codesegment zu Listing 5

Alle Zahlen sind in hexadezimaler Darstellung



Es wurden nicht alle Zeiger eingezeichnet, da dies zu verwirrend erscheinen würde.

μὴ θορυβὴσης εμοῦ κύκλους
noli turbare circulos meos
Störe meine Kreise nicht

Die Berechnur
Kre

g der szahl π

Pi mit bis zu 15.500 Stellen Genauigkeit

von Roland und Manfred Fietkau

Es war etwa zu Beginn des 19. Jahrhunderts, als die berühmte Pariser Akademie der Wissenschaften bekanntgab, man werde keine weiteren Lösungen zum Problem der *Quadratur des Kreises* mehr entgegennehmen und überprüfen.
Zumeist waren es Hobby-Mathematiker, die sich an diesem Problem versuchten, verführt von der einfachen Aufgabenstellung dieser klassischen Forderung: Ein Kreis sei mit den Hilfsmitteln Zirkel und Lineal in ein flächengleiches Quadrat zu verwandeln.
Vermutlich war es gerade diese Schlichtheit in der Formulierung, die vor allem Leute mit ansonsten nur elementaren Kenntnissen der Geometrie veranlaßte, sich auf die Suche zu machen, fasziniert von dem Gedanken, es bedürfe nur eines genialen Einfalles oder einer zufälligen Entdeckung, um in die Geschichte der Mathematik einzugehen.

1. Geschichtliches

Schon der berühmte Archimedes (287-212 v.Chr.) hatte bewiesen, daß ein Kreis flächengleich ist mit einem rechtwinkligen Dreieck, dessen eine Kathete gleich dem Radius des Kreises und dessen andere gleich dem Umfang des Kreises ist (siehe **Abb. 1**).

Bereits damals aber hatten griechische Mathematiker eingewandt, ein solches Dreieck könne nicht gezeichnet werden, denn die beiden Katheden seien inkommensurabel, d. h. es sei nicht möglich, ihr Längenverhältnis durch natürliche Zahlen auszudrücken. Andererseits war ganz offensichtlich, daß es ein solches Dreieck geben muß, denn der Umfang eines Kreises hat ja eine ganz bestimmte Länge. Mit anderen Worten: Wenn es gelingt, eine Strecke zu zeichnen, die exakt dem Umfang eines Kreises entspricht (*Rektifikation des Kreises*), ist das große Problem gelöst. Viele der in Paris eingereichten Lösungen waren sehr scharfsinnig und erfüll-

Den Satz „Störe meine Kreise nicht!" soll Archimedes einem Römer zugerufen haben, bevor er getötet wurde (anläßlich der Eroberung von Syrakus im Jahre 212 v. Chr.). Überliefert ist allerdings nur der lateinische, nicht dagegen der (hier zurückübersetzte) griechische Ausspruch. Archimedes hat sich nicht nur mit mathematischen Problemen (Kreisumfang, Kegelschnitte usw.) befaßt, sondern auch verschiedene physikalische Gesetze entdeckt (archimedische Schraube, Auftrieb usw.).

ten die Forderung mit einer Genauigkeit, die praktischen Zwecken völlig genügte. Es waren aber im strengen Sinne nur Näherungslösungen, die dem angestrebten Ergebnis fast unbegrenzt nahe kamen, es aber nicht exakt erreichten.

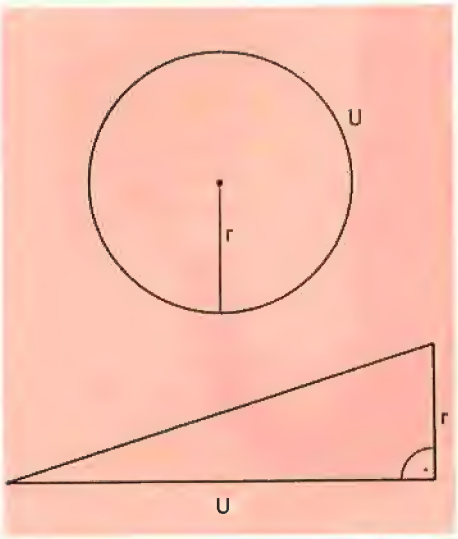


Abb. 1: Flächengleichheit

Mathematiker hatten schon seit der Zeit der griechischen Mathematik vermutet, daß die Aufgabe unlösbar ist, der Beweis wurde jedoch erst in der zweiten Hälfte des 19. Jahrhundert von F. Lindemann (1852-1939) erbracht. Die Bezeichnung Pi geht auf L. Euler (1707-1783) zurück und leitet sich vom Anfangsbuchstaben des griechischen Wortes „perimetron" (Umfang) ab. Lindemann bewies nun, daß es sich bei Pi um eine sog. transzendente Zahl handelt. Ohne auf den Beweis näher einzugehen, sei folgendes gesagt: Bildet man eine beliebige Summe aus Potenzen von Pi, gibt es beliebig viele andere solcher Summen, die dem Wert der zuerst gebildeten Summe sehr nahe kommen, es gibt aber keine, die exakt den gleichen Wert hat.
Bereits Archimedes schränkte den Wert von Pi mit
$223/71 < Pi < 22/7$
ziemlich genau ein. Später hat man den Wert noch wesentlich genauer ermittelt. Ludolph van Ceulen (1540-1610) ermittelte 35 Dezimalstellen, indem er sich dem Kreisumfang durch regelmäßige Polygone (Vielecke) von außen und von innen näherte. Nach ihm wird Pi auch häufig die Ludolfsche Zahl genannt. Die moderne Mathematik stellte dann effektivere Methoden zur Verfügung, mit deren Hilfe z.B. Z. Dase (1824-1861) 200 Dezimalstellen berechnete.
Es waren aber erst die Großrechenanlagen unseres Jahrhunderts, die wesentliche Fortschritte brachten. Mit ihrer Hilfe sind heute einige Hunderttausend Dezimalstellen von Pi bekannt. Diese Leistungen sind mit dem Apple selbstverständlich nicht zu erzielen, aber bis etwa 11.500 Stellen sind doch zu erreichen. Wird das DOS in die LC verschoben, kann man durch Änderung zweier Labels im Quelltext
Zeile 58: N EQU $6200,
Zeile 59: ERG EQU $BF00
ca. 15.500 Stellen berechnen.
Selbstverständlich muß man sich die Frage stellen, ob das Errechnen immer weiterer Dezimalstellen von Pi irgendeinen Sinn hat. Für praktische Berechnungen reichen einige wenige Dezimalen aus. Auch der wissenschaftliche Ertrag solcher Bemühungen ist nicht sehr erheblich.
Letztlich bleibt wohl nur die schon zur Genüge bekannte Antwort auf Fragen:
Warum berechnet man immer größere Primzahlen, berechnet Fakultäten mit einer astronomischen Anzahl von Ziffern und stellt immer phantastischere Rekorde bei der Berechnung von transzendenten Zahlen wie Pi oder der Eulerschen Zahl e auf?
Weil man es kann!

2. Das Programm

Die Berechnung ist mathematisch nicht sonderlich kompliziert, erfordert aber einen sehr großen Rechenaufwand. Um den zeitlichen Aufwand einigermaßen in Grenzen zu halten, ist das Programm in Assembler geschrieben. Es berechnet 1000 Dezimalen in ca. 6 Minuten. Der Zeitbedarf wächst etwa quadratisch mit der Stellenzahl, d.h. um das 10fache an Stellen zu berechnen, benötigt man nicht das 10fache, sondern das 100fache der Zeit. Das aufrufende Applesoft-Programm **PI-.START** bietet ein kleines Menü und erfordert als Eingabe lediglich die Anzahl der Nachkommastellen. Die Ausgabe erfolgt wahlweise auf Drucker, Diskette oder Bildschirm. Die vom Programm erzeugten Textfiles enthalten die errechneten Ziffern im ASCII-Format und erhalten den jeweiligen Filenamen PIx, wobei x für die Anzahl der Nachkommastellen steht.

3. Die Mathematik

Das Programm benutzt die bekannte Formel von C. F. Gauß:

$$\pi = 48 \arctan\frac{1}{18} + 32 \arctan\frac{1}{57} - 20 \arctan\frac{1}{239}$$

Hauptproblem ist es also, die Arcustangensfunktion für Argumente der Form 1/q mit q = 18,57,239 beliebig genau zu berechnen. Für den Arcustangens gibt es zum Glück eine ziemlich einfach gebaute, schnell konvergierende Reihenentwicklung, die für diesen speziellen Fall so aussieht:

$$\arctan\frac{1}{q} = \frac{1}{q} - \frac{1}{3q^3} + \frac{1}{5q^5} - \frac{1}{7q^7} \cdots$$

Wieviele Reihenglieder muß man nun berücksichtigen, um eine vorgegebene Genauigkeit zu erreichen? Das kann man leicht abschätzen. Sei n die Anzahl der Dezimalstellen hinter dem Komma und a_k das k-te Reihenglied, so muß wegen der alternierenden Glieder gelten:

$$| a_k | < 10^{-n}$$

also:

$$\frac{1}{(2k+1) \cdot q^{(2k+1)}} < 10^n$$

bzw. für die Kehrwerte:

$$(2k+1) \cdot q^{(2k+1)} > 10^n$$

Diese Ungleichung muß nach k aufgelöst werden:

$$\log_{10}(2k+1) + (2k+1) \cdot \log_{10} q > n$$

Vernachlässigt man den ersten Term und läßt die 1 in der Klammer weg, ergibt sich:

$$2k \cdot \log_{10} q > n$$

und endlich:

$$k > \frac{n}{2 \cdot \log_{10} q}$$

Um mit der Formel von Gauß Pi z.B. auf 1000 Stellen zu berechnen, braucht man nach dieser Ungleichung für den ersten Arcustangens (q = 18) ungefähr 400 Reihenglieder, für den zweiten 290 und für den dritten 210.
Insgesamt müssen also ungefähr 900 Bruchzahlen auf 1000 Dezimalstellen berechnet und summiert werden. Selbst wenn man in einer Sekunde 100 Stellen gewinnen könnte, was sicherlich viel zu optimistisch ist, dauerte es noch 9000 Sekunden, also 2 1/2 Stunden.

Zum Glück geht es viel schneller! Wenn man nämlich zunächst die ganze Reihe a_0 bis a_k auf den Hauptnenner bringt, bleibt von den vielen Divisionen am Ende nur noch eine einzige übrig. Diesen Vorteil erkauft man sich allerdings mit dem Nachteil, daß mit riesig großen Zahlen gerechnet werden muß. Der Aufwand lohnt sich

aber. Für 1000 Dezimalstellen braucht das Programm jetzt nur noch 6 Minuten Rechenzeit.

Sehen wir uns die ersten beiden Glieder, a_0 und a_1, der Reihe an und bringen sie auf den Hauptnenner:

$$\frac{1}{q} - \frac{1}{3q^3} = \frac{3q^2}{3q^2} \cdot \frac{1}{q} - \frac{1}{3q^3} = \frac{3q^2-1}{3q^3}$$

oder abgekürzt:

$$\frac{Z_1}{N_1 \cdot q^3}$$

Dazu kommt jetzt:

$$a_2 = \frac{1}{5q^5}$$

$$\frac{3q^2-1}{3q^3} + \frac{1}{5q^5} =$$

$$\frac{5q^2}{5q^2} \cdot \frac{3q^2-1}{3q^3} + \frac{3}{3} \cdot \frac{1}{5q^5} =$$

$$\frac{(3q^2-1) \cdot 5q^2+3}{3 \cdot 5 \cdot q^5}$$

oder:

$$\frac{Z_1 \cdot 5q^2 + N_1}{N_1 \cdot 5 \cdot q^5} = \frac{Z_2}{N_2 \cdot q^5}$$

Mit:

$$a_3 = -\frac{1}{7q^7}$$

ergibt sich:

$$\frac{Z_2 \cdot 7q^2 - N_2}{N_2 \cdot 7 \cdot q^7} = \frac{Z_3}{N_3 \cdot q^7}$$

Man kann die Rekursionsformel für die Z bzw. N leicht ablesen:

$$Z_{i+1} = Z_i \cdot (2i + 1) \cdot q^2 +/- N_1$$
$$N_{i+1} = N_i \cdot (2i + 1)$$

Mit diesen beiden Formeln kann man die Reihe für arctan 1/q sukzessiv auf den Hauptnenner bringen. Schließlich bekommt man einen Zähler (Z_k) und einen Nenner

$$(N_k \cdot q^{(2k+1)})$$

mit denen man die Division ausführen muß.

Das Maschinenprogramm **PI** besteht im wesentlichen aus zwei Teilen:
– Ein Teil, der Ausdrücke der Form A · arctan (1/q) nach der oben beschriebenen Methode berechnet und je nach Bedarf zum Endergebnis (nämlich Pi) addiert oder davon subtrahiert.
– Ein Teil, der es erlaubt, das Ergebnis, das im Dualsystem vorliegt, in das Dezimalsystem zu übertragen.

Die Parameter (z.B. a = 48, q = 18), die Anzahl der Reihenglieder, der Modus (addieren/subtrahieren) usw. werden von dem Applesoft-Programm mit Pokes an das Maschinenprogramm übergeben.

4. Der Algorithmus

Eine bis ins einzelne gehende Beschreibung des Programms ist mit vertretbarem Aufwand nicht möglich. Es sei an dieser Stelle auf das Listing verwiesen.

Im folgenden wird jedoch versucht, den Programmablauf wenigstens zu skizzieren.

Um die Übergabe von Parametern einigermaßen darstellen zu können, soll die Beschreibung des Programms in einer Art von Pseudo-Pascal erfolgen.

Es folgt zunächst die Beschreibung des Applesoft-Teils.

```
BERECHNE PI NACH GAUß;
BEGIN
FRAGE ANWENDER NACH STELLENZAHL;
BERECHNE ANZAHL DER BENÖTIGTEN
 REIHENGLIEDER RG1..RG3;

 ATANREIHE (48, 18, RG1, STORE, STELLENZ.);
 ATANREIHE (32, 57, RG2, PLUS, STELLENZ.);
 ATANREIHE (20, 239, RG3, MINUS, STELLENZ.);

 FOR I := 0 TO STELLENZAHL DO
 AUSDRUCKEN EINER ZIFFER;
END.
```

Beschäftigen wir uns im folgenden nur mit der Prozedur ATANREIHE; sie hat fünf Parameter.

```
ATANREIHE (A, Q, RG, MODE, STZ);

VAR Z, N : SEHR LANGE GANZZAHL;

BEGIN
{ Initialisierung }

Z := 1; N := 1;
 F := 1; VORZ := -1;
  FOR I := 1 TO RG DO
   BEGIN
    F := F + 2;
    MULTIPLIZIERE Z MIT F;
    MULTIPLIZIERE Z MIT Q ↑ 2;
    ADDIERE/SUBTRAHIERE Z
     UND N (VORZ);
    MULTIPLIZIERE N MIT F;
    VORZEICHEN UMKEHREN;
   END;

{ Nenner N ist noch nicht vollständig }

FOR I := 1 TO RG DO
 BEGIN
  MULTIPLIZIERE N MIT Q ↑ 2;
  MULTIPLIZIERE N MIT Q;
  MULTIPLIZIERE Z MIT A;

  { und schließlich }
  DIVIDIERE Z UND N (MODE)
END;
```

Als nächstes sei exemplarisch die Prozedur MULTIPLIZIERE Z MIT F aufgeführt.

```
MULTIPLIZIERE Z MIT F;
BEGIN
 ERSTELLE MULTIPLIKATIONSTABELLE FÜR F;
 BEREITE MULTIPLIKATION VOR (ADRESSE VON
  Z, LÄNGE VON Z);
 MULTIPLIZIERE ( ZEIGER AUF TABELLE);
END;
```

Die anderen Multiplikationen haben im Prinzip den gleichen Aufbau, nur daß im Falle von q ↑ 2 die entsprechende Multiplikationstabelle nur einmal am Anfang erstellt wird.

Als letztes die Division:

```
 DIVIDIERE Z UND N (MODE);
 { zunächst Vorkommastellen }
 HILF := 0;
  WHILE Z >= N DO
  BEGIN
   ADDIERE/SUBTRAHIERE Z UND N (-1);
   HILF := HILF + 1;
  END;

 TRAGEIN (HILF, MODE);

{ jetzt normaler Divisionsalgorithmus }

 REPEAT
  BEGIN
   IF Z >= N
    THEN
    BEGIN
     ADDIERE/SUBTRAHIERE Z UND N (-1);
     TRAGEIN (1, MODE)
    END
    ELSE
     TRAGEIN (0, MODE);
   Z EINE STELLE NACH LINKS SCHIEBEN;
  END
 UNTIL STELLENZAHL ERREICHT;
END; { Division }

TRAGEIN (ERGEBNISZIFFER, MODE)
BEGIN
 CASE MODE OF
 STORE : ERGEBNIS := ERGEBNISZIFFER;
 PLUS  : ERGEBNIS := ERGEBNIS +
                     ERGEBNISZIFFER;
 MINUS : ERGEBNIS := ERGEBNIS -
                     ERGEBNISZIFFER;

 END
END;
```

Literatur:

1) Heinrich Tietze: Gelöste und ungelöste mathematische Probleme, München, 1957
2) Paul Karlson: Zauber der Zahlen, Frankfurt/M-Berlin, 1954

Kurzhinweise

1. Zweck:
Berechnung der Zahl Pi auf bis zu 15.500 Stellen
2. Konfiguration:
II+, IIe oder IIc;
DOS 3.3 (ggf. 64K-DOS)
3. Test:
RUN PI.START
4. Sammeldisk:
PI.START
(Applesoft-Rahmenprogramm)
T.PI
(Big-Mac-Quelltext)
PI
(Assembler-Programm)

Die Zahl Pi mit 100 Stellen Genauigkeit

3.14159265358979323846264338327950288419716939937510
58209749445923078164062862089986280348253421170679

PI.START

```
1000 REM ZEHNERLOGARITHMUS
1010 DEF FN ZL(X) = LOG (X) / LOG (10)
1020 DEF FN H(X) = INT (X / 256)
1030 DEF FN L(X) = X - 256 * FN H(X)
1040 X = 3 * 256 + 10: POKE X,44: POKE X + 1,16: POKE X + 2,192:
     POKE X + 3,44: POKE X + 4,0: POKE X + 5,192: POKE X + 6,16:
     POKE X + 7,251: POKE X + 8,44: POKE X + 9,16: POKE X +
     10,192: POKE X + 11,96:H0 = X: REM WAIT
1050 TEXT : HOME :A = 1:B = 1
1060 PRINT : PRINT CHR$ (4);"BLOAD PI"
1070 HOME : INVERSE : PRINT "PROGRAMM ZUR BERECHNUNG VON PI":
     NORMAL
1080 PRINT : PRINT "--------------------------------"
1090 PRINT "A)USGABE AUF : ";
1100 IF A = 1 THEN INVERSE : PRINT "DRUCKER";: NORMAL : PRINT
     "/BILDSCHIRM/DISK"
1110 IF A = 2 THEN PRINT "DRUCKER/";: INVERSE : PRINT
     "BILDSCHIRM";: NORMAL : PRINT "/DISK"
1120 IF A = 3 THEN PRINT "DRUCKER/BILDSCHIRM/";: INVERSE : PRINT
     "DISK": NORMAL
1130 PRINT
1140 PRINT "D)ATEI       : ";
1150 IF B = 1 THEN INVERSE : PRINT "BERECHNEN";: NORMAL : PRINT
     "/VON DISKETTE"
1160 IF B = 2 THEN PRINT "BERECHNEN/";: INVERSE : PRINT "VON
     DISKETTE": NORMAL
1170 PRINT : PRINT "S)TART"
1180 PRINT : PRINT "E)NDE"
1190 PRINT : PRINT "--------------------------------"
1200 PRINT : PRINT "-> ";: GET T$
1210 IF T$ = "E" THEN PRINT "ENDE": END
1220 IF T$ = "S" THEN 1290
1230 IF T$ = "A" THEN 1270
1240 IF T$ < > "D" THEN 1070
1250 B = B + 1: IF B > 2 THEN B = 1
1260 GOTO 1070
1270 A = A + 1: IF A > 3 THEN A = 1
1280 GOTO 1070
1290 REM START
1300 C = A + 10 * B
1310 IF C = 23 THEN PRINT CHR$ (7); CHR$ (7);: GOTO 1070
1320 IF C = 21 THEN 1750
1330 IF C = 22 THEN 1910
1340 GOSUB 2030
1350 RESTORE :K = 3 * 256:RECHNEN = 28 * 256:DRUCKEN = RE + 3
1360 READ X,Q,WZT
1370 IF X = - 1 THEN 1460: REM AUSGABE
1380 BS = NN / FN ZL(2) / 8 + 1
1390 POKE K, FN L(BS): POKE K + 1, FN H(BS)
1400 AZ = NN / 2 / FN ZL(Q)
1410 POKE K + 2, FN L(AZ): POKE K + 3, FN H(AZ)
1420 POKE K + 4, FN L(Q * Q): POKE K + 5, FN H(Q * Q)
1430 POKE K + 6,Q: POKE K + 7,X: POKE K + 8,WZT
1440 HGR : CALL RECHNEN
1450 GOTO 1360
1460 TEXT : HOME : REM AUSGABE
1470 IF A = 3 THEN 1650: REM DISK
1480 IF A = 2 THEN 1600: REM BILDSCHIRM
1490 PRINT : PRINT CHR$ (4);"PR#1"
1500 PRINT "PI AUF ";NN - 10;"  STELLEN ": PRINT
     "----------------------------": CALL DR
1510 PRINT ".":ZZ = 0:Z = 0
1520 FOR X = 1 TO NN - 10
1530 CALL DR:ZZ = ZZ + 1
1540 IF ZZ = 10 THEN ZZ = 0:Z = Z + 1: PRINT "  ";
1550 IF Z > 5 THEN Z = 0: PRINT
1560 NEXT
1570 PRINT : PRINT "----------------------------"
1580 PRINT : PRINT CHR$ (4);"PR#0"
1590 GOTO 1070
1600 REM BILSCHIRM
1610 CALL DR: PRINT "."
1620 FOR X = 1 TO NN - 10
1630 CALL DR: IF PEEK (12 * 4096) > 128 THEN CALL H0
1640 NEXT : PRINT : PRINT "<TASTE>";: CALL H0: GOTO 1070
1650 REM DISK
1660 PRINT CHR$ (4);"OPEN";NA$: PRINT CHR$ (4);"CLOSE"
1670 PRINT CHR$ (4);"DELETE";NA$
1680 PRINT : PRINT CHR$ (4);"OPEN";NA$
1690 PRINT : PRINT CHR$ (4);"WRITE";NA$
1700 FOR X = 0 TO NN - 10
1710 CALL DR
1720 NEXT
1730 PRINT : PRINT CHR$ (4);"CLOSE"
1740 GOTO 1070
1750 REM VON DISK
1760 GOSUB 2030
1770 PRINT CHR$ (4);"PR#1"
1780 PRINT : PRINT CHR$ (4);"OPEN";NA$
1790 PRINT CHR$ (4);"READ";NA$
1800 ZZ = 0:Z = 0
1810 PRINT "PI AUF ";NN - 10;"  STELLEN": PRINT
     "----------------------------"
1820 GET X$: PRINT CHR$ (4);X$;"."
1830 FOR X = 1 TO NN - 10
1840 GET X$
1850 PRINT CHR$ (4);X$;:ZZ = ZZ + 1
1860 IF ZZ = 10 THEN ZZ = 0:Z = Z + 1: PRINT "  ";
1870 IF Z > 5 THEN Z = 0: PRINT
1880 NEXT : PRINT : PRINT "----------------------------"
1890 PRINT CHR$ (4);"PR#0"
1900 PRINT CHR$ (4);"CLOSE": GOTO 1070
1910 GOSUB 2030
1920 HOME : PRINT "PI AUF ";NN - 10;" STELLEN": PRINT
     "----------------------"
1930 PRINT : PRINT CHR$ (4);"OPEN";NA$
1940 PRINT CHR$ (4);"READ";NA$
1950 GET X$: PRINT CHR$ (4);X$;"."
1960 FOR X = 1 TO NN - 10
1970 GET X$: PRINT CHR$ (4);X$;
1980 IF PEEK (12 * 4096) > 128 THEN CALL H0
1990 NEXT
2000 PRINT : PRINT "<TASTE>";: CALL H0
2010 PRINT : PRINT CHR$ (4);"CLOSE"
2020 GOTO 1070
2030 REM ANZAHL DER STELLEN/FILENAME
2040 INPUT "ANZAHL DER STELLEN : ";NN
2050 IF NN > 0 THEN NA$ = "PI" + STR$ (NN):NN = NN + 10:
     RETURN
2060 GOTO 2040
2070 DATA 48,18,64
2080 DATA 32,57,0
2090 DATA 20,239,128
2100 DATA -1,0,0
```

PI

BSAVE PI, A$1C00, L$0377

```
1    **********************
2    *   Berechnung der   *
3    *    Kreiszahl Pi    *
4    *                    *
5    *   Roland Fietkau   *
6    *                    *
7    *     Juli 1985      *
8    *                    *
9    **********************
10            ORG  $1C00
11   PTR1     EQU  $00
12   PTR1LO   EQU  PTR1
13   PTR1HI   EQU  PTR1LO+1
14   LENZLO   EQU  $02
15   LENZHI   EQU  $03
16   PTR2     EQU  $04
17   PTR2LO   EQU  PTR2
18   PTR2HI   EQU  PTR2LO+1
19   LENNLO   EQU  $06
20   LENNHI   EQU  $07
21   PTRE     EQU  $08
22   PTRELO   EQU  PTRE
23   PTREHI   EQU  PTRELO+1
24   UE1      EQU  $FC
25   UE2      EQU  $FD
26   SAM8BIT  EQU  $FE
27   ZAEHL1   EQU  $A0
28   ZAEHL2   EQU  $A1
29   AP1      EQU  $A0
30   FHI      EQU  $2FE
31   FLO      EQU  $2FD
32   VORZ     EQU  $2FF
33   *
34   * Werden vom BASIC-Programm
35   * vorbesetzt
36   *
37   BASLIST  EQU  $300
38   BINSTLO  EQU  BASLIST+0
39   BINSTHI  EQU  BASLIST+1
40   ANZLLO   EQU  BASLIST+2
41   ANZLHI   EQU  BASLIST+3
```

```
                        42  Q2LO     EQU  BASLIST+4
                        43  Q2HI     EQU  BASLIST+5
                        44  Q        EQU  BASLIST+6
                        45  VORFKTOR EQU  BASLIST+7
                        46  WZT      EQU  BASLIST+8
                        47  PIANFLO  EQU  $316
                        48  PIANFHI  EQU  $317
                        49  *
                        50  QTABL    EQU  $1600
                        51  QTABM    EQU  $1700
                        52  QTABH    EQU  $1800
                        53  *
                        54  FTABL    EQU  $1900
                        55  FTABM    EQU  $1A00
                        56  FTABH    EQU  $1B00
                        57  *
                        58  Z        EQU  $2000
                        59  N        EQU  $5800
                        60  ERG      EQU  $9500
                        61  *
                        62  * Bei START1 aufgerufen, berechnet
                        63  * das Programm den Arcustangens
                        64  * für die vom BASIC-Programm
                        65  * übergebenen Parameter
                        66  *
                        67  START1   EQU  *
                        68  *
                        69  * Bei START2 aufgerufen, wird die
                        70  * berechnete Zahl ins Dez.-System
                        71  * umgewandelt und via $FDED
                        72  * ausgegeben.
                        73  *
1C00: 4C 86 1E          74  START2   JMP  ARCTAN
1C03: A9 0A             75          LDA  #10
1C05: A0 00             76          LDY  #$00
1C07: 20 F9 1C          77          JSR  FAKTTAB
                        78  *
                        79  * umwandeln
                        80  *
1C0A: AD 00 95          81          LDA  ERG
1C0D: 18                82          CLC
1C0E: 69 B0             83          ADC  #$B0
1C10: 20 ED FD          84          JSR  $FDED
1C13: A9 00             85          LDA  #$00
1C15: 8D 00 95          86          STA  ERG
                        87  *
1C18: AC 17 03          88          LDY  PIANFHI
1C1B: AD 00 03          89          LDA  BINSTLO
1C1E: AE 01 03          90          LDX  BINSTHI
1C21: 20 2F 1C          91          JSR  PARM1
1C24: AD 16 03          92          LDA  PIANFLO
1C27: 85 00             93          STA  PTR1LO
1C29: A2 19             94          LDX  #>FTABL
1C2B: 20 43 1C          95          JSR  MULTPL
1C2E: 60                96          RTS
                        97  *
                        98  * Sammlung verschiedener
                        99  * Unterprogramme
                       100  *
                       101  * MULTIPLIKATION
                       102  *
                       103  PARM1    EQU  *
                       104  * Bereitet Multiplikation vor.
                       105  * Anfangsadresse der langen Zahl
                       106  * kommt nach PTR1.
                       107  * Einerkomplement der Länge
                       108  * in ZAEHL1, ZAEHL2
                       109  *
                       110  * YREG = ANFANGHI
                       111  * ACC  = LAENGLO
                       112  * XREG = LAENGHI
                       113  *
1C2F: 84 01            114          STY  PTR1HI
                       115  * Low-Byte ist immer Null
1C31: A0 00            116          LDY  #$00
1C33: 84 00            117          STY  PTR1LO
1C35: 49 FF            118          EOR  #$FF
1C37: 85 A0            119          STA  ZAEHL1
1C39: 8A               120          TXA
1C3A: 49 FF            121          EOR  #$FF
1C3C: 85 A1            122          STA  ZAEHL2
                       123  *
                       124  * Übertrag-Bytes löschen
                       125  *
1C3E: 84 FC            126          STY  UE1        ;YREG = 0
1C40: 84 FD            127          STY  UE2
1C42: 60               128          RTS
                       129  *
                       130  MULTPL   EQU  *
                       131  * Setzt die Adressen der richtigen
                       132  * Tafel in die Mult.-Routine ein,
                       133  * multipliziert und gibt
```

```
                    134  * die korrigierte Länge zurück
                    135  *
                    136  * XREG = TABHIBYTE
                    137  * YREG = Ø
                    138  *
1C43: 8E 54 1C      139          STX  MTAB1+2
1C46: E8            14Ø          INX
1C47: 8E 5B 1C      141          STX  MTAB2+2
1C4A: E8            142          INX
1C4B: 8E 62 1C      143          STX  MTAB3+2
                    144  *
1C4E: 18            145          CLC
1C4F: B1 ØØ         146  MULTP1  LDA  (PTR1),Y
1C51: AA            147          TAX
1C52: BD ØØ 11      148  MTAB1   LDA  $11ØØ,X
1C55: 65 FC         149          ADC  UE1
1C57: 91 ØØ         15Ø          STA  (PTR1),Y
1C59: BD ØØ 22      151  MTAB2   LDA  $22ØØ,X
1C5C: 65 FD         152          ADC  UE2
1C5E: 85 FC         153          STA  UE1
1C6Ø: BD ØØ 33      154  MTAB3   LDA  $33ØØ,X
1C63: 69 ØØ         155          ADC  #$ØØ
1C65: 85 FD         156          STA  UE2
1C67: C8            157          INY
1C68: DØ Ø2         158          BNE  *+4
1C6A: E6 Ø1         159          INC  PTR1HI
1C6C: E6 AØ         16Ø          INC  ZAEHL1
1C6E: DØ DF         161          BNE  MULTP1
1C7Ø: E6 A1         162          INC  ZAEHL2
1C72: DØ DB         163          BNE  MULTP1
                    164  *
                    165  *
1C74: 84 ØØ         166          STY  PTR1LO
                    167  * PTR1 zeigt jetzt auf das erste
                    168  * Byte über der alten Zahl
1C76: AØ ØØ         169          LDY  #$ØØ
                    17Ø  *
1C78: A5 FC         171          LDA  UE1
1C7A: Ø5 FD         172          ORA  UE2
1C7C: DØ Ø1         173          BNE  UEUNGLØ
1C7E: 6Ø            174          RTS             ;beide Ø
1C7F: A5 FC         175  UEUNGLØ LDA  UE1
1C81: 91 ØØ         176          STA  (PTR1),Y
1C83: C8            177          INY
1C84: A5 FD         178          LDA  UE2
1C86: DØ Ø1         179          BNE  UE2UNGLØ
1C88: 6Ø            18Ø          RTS             ;UE2 = Ø
1C89: 91 ØØ         181  UE2UNGLØ STA  (PTR1),Y
1C8B: C8            182          INY
1C8C: 6Ø            183          RTS
                    184  * YREG = Wert, um den die Länge
                    185  * vom aufrufenden Programm
                    186  * korrigiert werden muß (Ø,1,2)
                    187  *
                    188  ZMALQ2   EQU  *
                    189  * Zähler := Zähler * Q ↑ 2
1C8D: AØ 2Ø         19Ø          LDY  #>Z
1C8F: A5 Ø2         191          LDA  LENZLO
1C91: A6 Ø3         192          LDX  LENZHI
1C93: 2Ø 2F 1C      193          JSR  PARM1
                    194  *
1C96: A2 16         195          LDX  #>QTABL
1C98: 2Ø 43 1C      196          JSR  MULTPL
1C9B: 4C AC 1C      197          JMP  NEULAENG
                    198  ZMALF    EQU  *
                    199  * Zähler := Zähler * F
1C9E: AØ 2Ø         2ØØ          LDY  #>Z
1CAØ: A5 Ø2         2Ø1          LDA  LENZLO
1CA2: A6 Ø3         2Ø2          LDX  LENZHI
1CA4: 2Ø 2F 1C      2Ø3          JSR  PARM1
1CA7: A2 19         2Ø4          LDX  #>FTABL
1CA9: 2Ø 43 1C      2Ø5          JSR  MULTPL
                    2Ø6  * Länge korrigieren
                    2Ø7  NEULAENG EQU  *
1CAC: 98            2Ø8          TYA
1CAD: 65 Ø2         2Ø9          ADC  LENZLO     ;C = Ø
1CAF: 85 Ø2         21Ø          STA  LENZLO
1CB1: 9Ø Ø2         211          BCC  *+4
1CB3: E6 Ø3         212          INC  LENZHI
1CB5: 6Ø            213          RTS
                    214  NMALF    EQU  *
                    215  * Nenner := Nenner * F
1CB6: AØ 58         216          LDY  #>N
1CB8: A5 Ø6         217          LDA  LENNLO
1CBA: A6 Ø7         218          LDX  LENNHI
1CBC: 2Ø 2F 1C      219          JSR  PARM1
1CBF: A2 19         22Ø          LDX  #>FTABL
1CC1: 2Ø 43 1C      221          JSR  MULTPL
1CC4: 98            222          TYA
1CC5: 65 Ø6         223          ADC  LENNLO
1CC7: 85 Ø6         224          STA  LENNLO
```

```
1CC9: 9Ø Ø2         225          BCC  *+4
1CCB: E6 Ø7         226          INC  LENNHI
1CCD: 6Ø            227          RTS
                    228  *
                    229  * MULTIPLIKATIONSTABELLE
                    23Ø  * ACC = kleine Zahl - Low-Byte
                    231  * YREG= " HI
                    232  QUADTAB  EQU  *
1CCE: 85 AØ         233          STA  AP1
1CDØ: A9 ØØ         234          LDA  #$ØØ
1CD2: 8D ØØ 16      235          STA  QTABL
1CD5: 8D ØØ 17      236          STA  QTABM
1CD8: 8D ØØ 18      237          STA  QTABH
1CDB: A2 Ø1         238          LDX  #$Ø1
                    239  QTAB1    EQU  *
1CDD: BD FF 15      24Ø          LDA  QTABL-1,X
1CEØ: 18            241          CLC
1CE1: 65 AØ         242          ADC  AP1
1CE3: 9D ØØ 16      243          STA  QTABL,X
1CE6: 98            244          TYA             ;* HIBYTE
1CE7: 7D FF 16      245          ADC  QTABM-1,X
1CEA: 9D ØØ 17      246          STA  QTABM,X
1CED: BD FF 17      247          LDA  QTABH-1,X
1CFØ: 69 ØØ         248          ADC  #$ØØ
1CF2: 9D ØØ 18      249          STA  QTABH,X
1CF5: E8            25Ø          INX
1CF6: DØ E5         251          BNE  QTAB1
1CF8: 6Ø            252          RTS
                    253  *
                    254  * Das gleiche nochmal aber
                    255  * mit anderen Adressen
                    256  *
                    257  FAKTTAB  EQU  *
1CF9: 85 AØ         258          STA  AP1
1CFB: A9 ØØ         259          LDA  #$ØØ
1CFD: 8D ØØ 19      26Ø          STA  FTABL
1DØØ: 8D ØØ 1A      261          STA  FTABM
1DØ3: 8D ØØ 1B      262          STA  FTABH
1DØ6: A2 Ø1         263          LDX  #$Ø1
                    264  FTAB1    EQU  *
1DØ8: BD FF 18      265          LDA  FTABL-1,X
1DØB: 18            266          CLC
1DØC: 65 AØ         267          ADC  AP1
1DØE: 9D ØØ 19      268          STA  FTABL,X
1D11: 98            269          TYA             ;* HIBYTE
1D12: 7D FF 19      27Ø          ADC  FTABM-1,X
1D15: 9D ØØ 1A      271          STA  FTABM,X
1D18: BD FF 1A      272          LDA  FTABH-1,X
1D1B: 69 ØØ         273          ADC  #$ØØ
1D1D: 9D ØØ 1B      274          STA  FTABH,X
1D2Ø: E8            275          INX
1D21: DØ E5         276          BNE  FTAB1
1D23: 6Ø            277          RTS
                    278  *
                    279  *
                    28Ø  * ADDIEREN, SUBTRAHIEREN
                    281  *
                    282  * 1. Z := Z + N
                    283  * 2. Z := Z - N
                    284  * (Bedingung ist N <= Z)
                    285  *
                    286  ZPLMINN  EQU  *
1D24: A9 2Ø         287          LDA  #>Z
1D26: 85 Ø1         288          STA  PTR1HI
1D28: A9 58         289          LDA  #>N
1D2A: 85 Ø5         29Ø          STA  PTR2HI
                    291  *
1D2C: AØ ØØ         292          LDY  #$ØØ
1D2E: 84 ØØ         293          STY  PTR1LO
1D3Ø: 84 Ø4         294          STY  PTR2LO
                    295  *
                    296  * Schleifenlänge wird durch
                    297  * die Länge des Nenners bestimmt
1D32: A5 Ø6         298          LDA  LENNLO
1D34: 49 FF         299          EOR  #$FF
1D36: AA            3ØØ          TAX
1D37: A5 Ø7         3Ø1          LDA  LENNHI
1D39: 49 FF         3Ø2          EOR  #$FF
1D3B: 85 AØ         3Ø3          STA  ZAEHL1
                    3Ø4  *
                    3Ø5  * Abziehen oder addieren?
1D3D: 2C FF Ø2      3Ø6          BIT  VORZ
1D4Ø: 1Ø 25         3Ø7          BPL  ADDIEREN
                    3Ø8  *
                    3Ø9  * Abziehen
1D42: 38            31Ø          SEC
1D43: B1 ØØ         311  ZMI1    LDA  (PTR1),Y
1D45: F1 Ø4         312          SBC  (PTR2),Y
1D47: 91 ØØ         313          STA  (PTR1),Y
1D49: C8            314          INY
1D4A: DØ Ø4         315          BNE  *+6
1D4C: E6 Ø1         316          INC  PTR1HI
```

```
1D4E: E6 Ø5     317             INC  PTR2HI
                318  *DURCH?
1D5Ø: E8        319             INX
1D51: DØ FØ     32Ø             BNE  ZMI1
1D53: E6 AØ     321             INC  ZAEHL1
1D55: DØ EC     322             BNE  ZMI1       ;nein
                323  * Ja!
                324  * Muß Übertrag noch berück-
                325  * sichtigt werden?
1D57: BØ ØD     326             BCS  ZMIRTS     ;nein
                327  * Ja!
1D59: B1 ØØ     328  ZMI2      LDA  (PTR1),Y
1D5B: E9 ØØ     329             SBC  #$ØØ
1D5D: 91 ØØ     33Ø             STA  (PTR1),Y
1D5F: C8        331             INY
1D6Ø: DØ Ø2     332             BNE  *+4
1D62: E6 Ø1     333             INC  PTR1HI
1D64: 9Ø F3     334             BCC  ZMI2
1D66: 6Ø        335  ZMIRTS    RTS
                336  *
                337  ADDIEREN EQU  *
                338  * Erster Teil analog zu oben
                339  *
1D67: 18        34Ø             CLC
1D68: B1 ØØ     341  ZPL1      LDA  (PTR1),Y
1D6A: 71 Ø4     342             ADC  (PTR2),Y
1D6C: 91 ØØ     343             STA  (PTR1),Y
1D6E: C8        344             INY
1D6F: DØ Ø4     345             BNE  *+6
1D71: E6 Ø1     346             INC  PTR1HI
1D73: E6 Ø5     347             INC  PTR2HI
1D75: E8        348             INX
1D76: DØ FØ     349             BNE  ZPL1
1D78: E6 AØ     35Ø             INC  ZAEHL1
1D7A: DØ EC     351             BNE  ZPL1
1D7C: 9Ø 1F     352             BCC  ZPLRTS
1D7E: B1 ØØ     353  ZPL2      LDA  (PTR1),Y
1D8Ø: 69 ØØ     354             ADC  #$ØØ
1D82: 91 ØØ     355             STA  (PTR1),Y
1D84: C8        356             INY
1D85: DØ Ø2     357             BNE  *+4
1D87: E6 Ø1     358             INC  PTR1HI
1D89: BØ F3     359             BCS  ZPL2
                36Ø  *
                361  * Durch Addieren kann sich die
                362  * Länge von z erhöhen
                363  * PTR1 - LENZ < Anfang von Z?
1D8B: 98        364             TYA
1D8C: 38        365             SEC
1D8D: E5 Ø2     366             SBC  LENZLO
1D8F: A5 Ø1     367             LDA  PTR1HI
1D91: E5 Ø3     368             SBC  LENZHI
1D93: C9 2Ø     369             CMP  #>Z
1D95: BØ Ø6     37Ø             BCS  ZPLRTS     ;>=
1D97: E6 Ø2     371             INC  LENZLO
1D99: DØ Ø2     372             BNE  *+4
1D9B: E6 Ø3     373             INC  LENZHI
1D9D: 6Ø        374  ZPLRTS    RTS
                375  *
                376  * DIVISION
                377  *
                378  VERGLZN  EQU  *
                379  * Vergleicht Zähler mit Nenner
                38Ø  * und entscheidet, ob Nenner
                381  * abgezogen werden kann
                382  * oder nicht
                383  *
                384  * Zunächst die Längen von
                385  * Zähler und Nenner vergleichen
1D9E: A5 Ø2     386             LDA  LENZLO
1DAØ: C5 Ø6     387             CMP  LENNLO
1DA2: A5 Ø3     388             LDA  LENZHI
1DA4: E5 Ø7     389             SBC  LENNHI
1DA6: BØ Ø6     39Ø             BCS  LZOFFSET
                391  * Länge von N als Offset
1DA8: A4 Ø6     392             LDY  LENNLO
1DAA: A6 Ø7     393             LDX  LENNHI
1DAC: 9Ø Ø4     394             BCC  VERG2
1DAE: A4 Ø2     395  LZOFFSET LDY  LENZLO
1DBØ: A6 Ø3     396             LDX  LENZHI
1DB2: C8        397  VERG2     INY
1DB3: DØ Ø1     398             BNE  *+3
1DB5: E8        399             INX
1DB6: 18        4ØØ             CLC
1DB7: 8A        4Ø1             TXA
1DB8: 69 2Ø     4Ø2             ADC  #>Z
1DBA: 85 Ø1     4Ø3             STA  PTR1HI
1DBC: 8A        4Ø4             TXA
1DBD: 69 58     4Ø5             ADC  #>N
1DBF: 85 Ø5     4Ø6             STA  PTR2HI
1DC1: A9 ØØ     4Ø7             LDA  #$ØØ
1DC3: 85 ØØ     4Ø8             STA  PTR1LO
1DC5: 85 Ø4     4Ø9             STA  PTR2LO
```

```
1DC7: C8        410          INY
1DC8: 88        411  VERG1   DEY
1DC9: B1 ØØ     412          LDA  (PTR1),Y
1DCB: D1 Ø4     413          CMP  (PTR2),Y
1DCD: FØ Ø1     414          BEQ  NXTVGL
1DCF: 6Ø        415          RTS
                416  * C = 1: abziehbar
                417  * C = Ø: nicht abziehbar
                418  NXTVGL  EQU  *
                419  * Unentschieden
                42Ø  * Vergleiche die nächst niederen
                421  * Bytes
1DDØ: CØ ØØ     422          CPY  #$ØØ
1DD2: DØ F4     423          BNE  VERG1
1DD4: C6 Ø1     424          DEC  PTR1HI
1DD6: C6 Ø5     425          DEC  PTR2HI
                426  *
                427  * Durch?
1DD8: A5 Ø1     428          LDA  PTR1HI
1DDA: C9 2Ø     429          CMP  #>Z
1DDC: BØ EA     43Ø          BCS  VERG1      ;nein
                431  * Durch!
                432  * Z = N, also abziehbar
1DDE: 38        433          SEC
1DDF: 6Ø        434          RTS
                435  *
                436  ASLZ    EQU  *
                437  * Schiebt Z 1 Bit nach links
1DEØ: AØ 2Ø     438          LDY  #>Z
1DE2: A5 Ø6     439          LDA  LENNLO
1DE4: A6 Ø7     44Ø          LDX  LENNHI
1DE6: 18        441          CLC
1DE7: 69 Ø1     442          ADC  #$Ø1
1DE9: 9Ø Ø1     443          BCC  *+3
1DEB: E8        444          INX
                445  * Länge des Nenners plus eins
1DEC: 2Ø 2F 1C  446          JSR  PARM1
1DEF: A6 AØ     447          LDX  ZAEHL1
1DF1: 18        448          CLC
1DF2: B1 ØØ     449  ASLZ1   LDA  (PTR1),Y
1DF4: 2A        45Ø          ROL
1DF5: 91 ØØ     451          STA  (PTR1),Y
1DF7: C8        452          INY
1DF8: DØ Ø2     453          BNE  *+4
1DFA: E6 Ø1     454          INC  PTR1HI
1DFC: E8        455          INX
1DFD: DØ F3     456          BNE  ASLZ1
1DFF: E6 A1     457          INC  ZAEHL2
1EØ1: DØ EF     458          BNE  ASLZ1
1EØ3: 6Ø        459          RTS
                46Ø  *
                461  *
                462  TRAGEIN  EQU  *
                463  * Trägt die anfallenden Ergebnis-
                464  * Bits des Divisionsprogramms
                465  * der Reihe nach ein.
                466  * Berücksichtigt dabei, ob das
                467  * Ergebnis einfach gespeichert,
                468  * addiert oder subtrahiert werden
                469  * muß.
                47Ø  * Das Ergebnis wächst nach unten!
                471  * Ist die festgelegte Stellenzahl
                472  * erreicht, ist das Carry gesetzt.
                473  *
1EØ4: 26 FE     474          ROL  SAM8BIT
1EØ6: BØ Ø1     475          BCS  *+3
1EØ8: 6Ø        476          RTS
                477  * SAM8BIT sammelt immer 8 Bits
                478  *
1EØ9: A5 Ø9     479          LDA  PTREHI
1EØB: 48        48Ø          PHA             ;HIBYTE
1EØC: AØ ØØ     481          LDY  #$ØØ
1EØE: 2C Ø8 Ø3  482          BIT  WZT        ;was tun?
1E11: 7Ø 3Ø     483          BVS  STORE
1E13: 3Ø 17     484          BMI  SUBTR
                485  * Addieren
1E15: 18        486          CLC
1E16: A5 FE     487          LDA  SAM8BIT
1E18: 71 Ø8     488          ADC  (PTRE),Y
1E1A: 91 Ø8     489          STA  (PTRE),Y
1E1C: 9Ø 29     49Ø  WHILEC1 BCC  ADDENDE
1E1E: C8        491          INY
1E1F: DØ Ø2     492          BNE  *+4
1E21: E6 Ø9     493          INC  PTREHI
1E23: B1 Ø8     494          LDA  (PTRE),Y
1E25: 69 ØØ     495          ADC  #$ØØ       ;C = 1
1E27: 91 Ø8     496          STA  (PTRE),Y
1E29: 4C 1C 1E  497          JMP  WHILEC1
                498  *
1E2C: 38        499  SUBTR   SEC
1E2D: B1 Ø8     5ØØ          LDA  (PTRE),Y
1E2F: E5 FE     5Ø1          SBC  SAM8BIT
```

```
1E31: 91 Ø8     5Ø2          STA  (PTRE),Y
1E33: BØ 12     5Ø3  WHILEC0 BCS  SUBENDE
1E35: C8        5Ø4          INY
1E36: DØ Ø2     5Ø5          BNE  *+4
1E38: E6 Ø9     5Ø6          INC  PTREHI
1E3A: B1 Ø8     5Ø7          LDA  (PTRE),Y
1E3C: E9 ØØ     5Ø8          SBC  #$ØØ
1E3E: 91 Ø8     5Ø9          STA  (PTRE),Y
1E4Ø: 4C 33 1E  51Ø          JMP  WHILEC0
1E43: A5 FE     511  STORE   LDA  SAM8BIT
1E45: 91 Ø8     512          STA  (PTRE),Y
                513  SUBENDE  EQU  *
                514  ADDENDE  EQU  *
1E47: 68        515          PLA
1E48: AA        516          TAX
1E49: A4 Ø8     517          LDY  PTRELO
1E4B: DØ Ø1     518          BNE  *+3
1E4D: CA        519          DEX
1E4E: 88        52Ø          DEY
1E4F: 84 Ø8     521          STY  PTRELO
1E51: 86 Ø9     522          STX  PTREHI
                523  * Ausreichende Stellenzahl?
                524  * Wenn ja, Carry setzen!
1E53: CC 16 Ø3  525          CPY  PIANFLO
1E56: DØ Ø7     526          BNE  CLEARC
1E58: EC 17 Ø3  527          CPX  PIANFHI
1E5B: DØ Ø2     528          BNE  CLEARC
1E5D: 38        529          SEC
1E5E: 6Ø        53Ø          RTS
1E5F: 18        531  CLEARC  CLC
                532  * Nicht fertig
1E6Ø: A9 Ø1     533          LDA  #$Ø1
1E62: 85 FE     534          STA  SAM8BIT
1E64: 6Ø        535          RTS
                536  *
                537  LOESCHE  EQU  *
                538  * Den Speicherbereich
                539  * Z bis Pi-Anfang
                54Ø  * ZLO, HI...PIANFLO, HI
                541  *
1E65: AØ 2Ø     542          LDY  #>Z
1E67: AD 17 Ø3  543          LDA  PIANFHI
1E6A: 38        544          SEC
1E6B: E9 2Ø     545          SBC  #>Z
1E6D: AA        546          TAX
1E6E: AD 16 Ø3  547          LDA  PIANFLO
1E71: 2Ø 2F 1C  548          JSR  PARM1
1E74: A6 AØ     549          LDX  ZAEHL1
1E76: 98        55Ø          TYA             ;$ØØ
1E77: 91 ØØ     551  LOE1    STA  (PTR1),Y
1E79: C8        552          INY
1E7A: DØ Ø2     553          BNE  *+4
1E7C: E6 Ø1     554          INC  PTR1HI
1E7E: E8        555          INX
1E7F: DØ F6     556          BNE  LOE1
1E81: E6 A1     557          INC  ZAEHL2
1E83: DØ F2     558          BNE  LOE1
1E85: 6Ø        559          RTS
                56Ø  *
                561  ARCTAN   EQU  *
                562  * Berechnet A * Arctan (1/Q)
                563  *
                564  * Zunächst wird POINTER auf das
                565  * Ergebnis (Pi) gesetzt ...
1E86: A9 ØØ     566          LDA  #$ØØ       ;LOW
1E88: 38        567          SEC
1E89: 85 Ø8     568          STA  PTRELO
1E8B: ED ØØ Ø3  569          SBC  BINSTLO
1E8E: 8D 16 Ø3  57Ø          STA  PIANFLO
1E91: A9 95     571          LDA  #>ERG
1E93: 85 Ø9     572          STA  PTREHI
1E95: ED Ø1 Ø3  573          SBC  BINSTHI
1E98: 8D 17 Ø3  574          STA  PIANFHI
                575  * ...und die Adresse des
                576  * niederstwertigen Bytes bestimmt
                577  * (PIANFLO, HI)
                578  * Als nächstes wird der gesamte
                579  * Bereich von Z bis PIANF
                58Ø  * gelöscht
1E9B: 2Ø 65 1E  581          JSR  LOESCHE
                582  *
                583  * Initialisieren einiger Speicher
                584  * Z := 1; N := 1;
                585  * F := 1; VORZ := -1
1E9E: A9 Ø1     586          LDA  #$Ø1
1EAØ: 8D ØØ 2Ø  587          STA  Z
1EA3: 8D ØØ 58  588          STA  N
1EA6: AØ ØØ     589          LDY  #$ØØ
1EA8: 8D FD Ø2  59Ø          STA  FLO
1EAB: 8C FE Ø2  591          STY  FHI
                592  *
1EAE: 85 Ø2     593          STA  LENZLO
```

```
1EBØ: 84 Ø3     594           STY  LENZHI
1EB2: 85 Ø6     595           STA  LENNLO
1EB4: 84 Ø7     596           STY  LENNHI
                597 *
1EB6: 88        598           DEY             ;$FF
1EB7: 8C FF Ø2  599           STY  VORZ
                6ØØ *
                6Ø1 * Multiplikationstabelle für
                6Ø2 * Q ↑ 2 erstellen
1EBA: AD Ø4 Ø3  6Ø3           LDA  Q2LO
1EBD: AC Ø5 Ø3  6Ø4           LDY  Q2HI
1ECØ: 2Ø CE 1C  6Ø5           JSR  QUADTAB
                6Ø6 *
1EC3: AD Ø2 Ø3  6Ø7           LDA  ANZLLO
1EC6: 49 FF     6Ø8           EOR  #$FF
1EC8: AA        6Ø9           TAX
1EC9: AD Ø3 Ø3  61Ø           LDA  ANZLHI
1ECC: 49 FF     611           EOR  #$FF
                612 *
1ECE: 48        613 AT1       PHA
1ECF: 8A        614 AT2       TXA
1EDØ: 48        615           PHA
                616 * F := F + 2
1ED1: AD FD Ø2  617           LDA  FLO
1ED4: AC FE Ø2  618           LDY  FHI
1ED7: 18        619           CLC
1ED8: 69 Ø2     62Ø           ADC  #$Ø2
1EDA: 9Ø Ø1     621           BCC  *+3
1EDC: C8        622           INY
1EDD: 8D FD Ø2  623           STA  FLO
1EEØ: 8C FE Ø2  624           STY  FHI
1EE3: 2Ø F9 1C  625           JSR  FAKTTAB
                626 * Multiplikationstabelle für F
                627 *
                628 * Z := Z * F; Z := Z * Q ↑ 2
                629 * Z := Z +/- N
                63Ø * N := N * F
1EE6: 2Ø 9E 1C  631           JSR  ZMALF
1EE9: 2Ø 8D 1C  632           JSR  ZMALQ2
1EEC: 2Ø 24 1D  633           JSR  ZPLMINN
1EEF: 2Ø B6 1C  634           JSR  NMALF
                635 *
                636 * VORZ := -VORZ
1EF2: AD FF Ø2  637           LDA  VORZ
1EF5: 49 FF     638           EOR  #$FF
1EF7: 8D FF Ø2  639           STA  VORZ
                64Ø *
                641 * Durch?
1EFA: 68        642           PLA
1EFB: AA        643           TAX
1EFC: E8        644           INX
1EFD: DØ DØ     645           BNE  AT2        ;nein
1EFF: 68        646           PLA
1FØØ: 18        647           CLC
1FØ1: 69 Ø1     648           ADC  #$Ø1
1FØ3: DØ C9     649           BNE  AT1        ;nein
                65Ø * Durch!
                651 * Nenner muß noch einige Male
                652 * mit Q ↑ 2 malgenommen werden
1FØ5: AD Ø4 Ø3  653           LDA  Q2LO
1FØ8: AC Ø5 Ø3  654           LDY  Q2HI
1FØB: 2Ø F9 1C  655           JSR  FAKTTAB
1FØE: AD Ø2 Ø3  656           LDA  ANZLLO
1F11: 49 FF     657           EOR  #$FF
1F13: AA        658           TAX
1F14: AD Ø3 Ø3  659           LDA  ANZLHI
1F17: 49 FF     66Ø           EOR  #$FF
                661 *
1F19: 48        662 AT3       PHA
1F1A: 8A        663 AT4       TXA
1F1B: 48        664           PHA
1F1C: 2Ø B6 1C  665           JSR  NMALF
1F1F: 68        666           PLA
1F2Ø: AA        667           TAX
1F21: E8        668           INX
1F22: DØ F6     669           BNE  AT4
1F24: 68        67Ø           PLA
1F25: 18        671           CLC
1F26: 69 Ø1     672           ADC  #$Ø1
1F28: DØ EF     673           BNE  AT3
                674 *
                675 * Und noch einmal mit Q
1F2A: AD Ø6 Ø3  676           LDA  Q
1F2D: AØ ØØ     677           LDY  #$ØØ
1F2F: 2Ø F9 1C  678           JSR  FAKTTAB
1F32: 2Ø B6 1C  679           JSR  NMALF
                68Ø *
                681 * Z := A * Z; wegen:
                682 * 48 * ARCTAN..., 32 * ARCTAN..
1F35: AD Ø7 Ø3  683           LDA  VORFKTOR
1F38: AØ ØØ     684           LDY  #$ØØ
1F3A: 2Ø F9 1C  685           JSR  FAKTTAB
```

```
1F3D: 2Ø 9E 1C  686           JSR  ZMALF
                687 *
                688 * Zähler und Nenner sind fertig
                689 * berechnet.
                69Ø * Was fehlt ist nur noch Z/N
                691 *
                692 * Rechne zunÄchst Z := Z - N
                693 * bis Z <= N
1F4Ø: A9 FF     694           LDA  #$FF
1F42: 8D FF Ø2  695           STA  VORZ       ;abziehen
1F45: 18        696           CLC
1F46: 69 Ø1     697 D1        ADC  #$Ø1
1F48: 48        698           PHA
1F49: 2Ø 9E 1D  699           JSR  VERGLZN
1F4C: 9Ø Ø7     7ØØ           BCC  D2
1F4E: 2Ø 24 1D  7Ø1           JSR  ZPLMINN
1F51: 68        7Ø2           PLA
1F52: 18        7Ø3           CLC
1F53: 9Ø F1     7Ø4           BCC  D1
1F55: 68        7Ø5 D2        PLA
1F56: 38        7Ø6           SEC
1F57: 6A        7Ø7           ROR
1F58: 85 FE     7Ø8           STA  SAM8BIT
1F5A: 2Ø Ø4 1E  7Ø9           JSR  TRAGEIN
                71Ø *
1F5D: 2Ø EØ 1D  711           JSR  ASLZ
                712 *
                713 DIVID2    EQU  *
1F6Ø: 2Ø 9E 1D  714           JSR  VERGLZN
1F63: Ø8        715           PHP
1F64: 2Ø Ø4 1E  716           JSR  TRAGEIN
1F67: BØ ØC     717           BCS  DIVDURCH
1F69: 28        718           PLP
1F6A: 9Ø Ø3     719           BCC  NURASL
1F6C: 2Ø 24 1D  72Ø           JSR  ZPLMINN
1F6F: 2Ø EØ 1D  721 NURASL    JSR  ASLZ
1F72: 4C 6Ø 1F  722           JMP  DIVID2
1F75: 28        723 DIVDURCH  PLP
1F76: 6Ø        724           RTS

887 Bytes
```

Gemeinsame F8-Monitor-Routinen

Apple II+/e/c/enhanced

zusammengestellt von Ulrich Stiehl

In Ergänzung zu unserem Beitrag „Die neuen ROMs“ in Peeker, 10/85, S. 22ff., bringen wir nachfolgend eine Kurzaufstellung der bei allen Apple-II-Typen gemeinsam verwendbaren F8-Monitor-Routinen (F8 bedeutet $F800-$FFFF).
l: Die mit „l“ für legal markierten Routinen wurden von der Firma Apple dokumentiert. Allerdings findet man in den verschiedenen Handbüchern zum II+, IIe usw. unterschiedliche Listen. Als legal gelten daher alle irgendwo von Apple aufgeführten Routinen. Diese dürften auch noch bei zukünftigen ROMs benutzbar sein.
s: Die mit „s“ für speziell gekennzeichneten Routinen sollten möglichst vermieden werden, auch wenn sie z.Zt. auf allen Apple-II-Typen aufgerufen werden können.
w: Die mit „w“ für wissenswert markierten Tabellen und Vektoren sind keine aufrufbaren Routinen. Das gleiche gilt für die zusätzlich gelisteten Zero-Page-Adressen.

Routinen und Zero-Page-Adressen, die *nicht markiert* sind, können trotzdem z.Zt. auf allen Apple-II-Typen verwendet werden, auch wenn sie nicht seitens der Firma Apple dokumentiert worden sind.
Es gibt darüber hinaus noch zahlreiche andere Routinen, die jedoch nicht auf allen vier Apple-II-Typen gleichermaßen benutzt werden können und deshalb vermieden werden sollen.
Hinweis: Für Anfänger bringen wir zu einem späteren Zeitpunkt Übungsbeispiele zu ausgewählten Monitor-Routinen.

```
Gemeinsame Monitor-Routinen II+/e/n/c
=====================================

l = (l)egal und von Apple dokumentiert
s = (s)peziell und nur für Eingeweihte
w = (w)issenswert, aber nicht aufrufen
+ = Apple II+
e = Apple IIe
n = Apple IIe mit neuen ROMs
c = Apple IIc
A = A-Register, Akkumulator
X = X-Register
Y = Y-Register

Zero-Page-Adressen
==================

Bootslot
--------
LOC0      EQU  $0000 ;Boot-Vektor
LOC1      EQU  $0001

Lores
-----
GBASL     EQU  $0026 ;Basis-Adresse
GBASH     EQU  $0027
H2        EQU  $002C ;Temp HLINE
V2        EQU  $002D ;Temp VLINE
MASK      EQU  $002E ;Farbmaske
COLOR     EQU  $0030 ;Loresfarbe

Text
----
WNDLFT    EQU  $0020 ;linker Rand
WNDWDTH   EQU  $0021 ;Fensterbreite
WNDTOP    EQU  $0022 ;oberer Rand
WNDBTM    EQU  $0023 ;unterer Rand

Bildschirmsteuerung
-------------------
CH        EQU  $0024 ;horiz.Cursor
CV        EQU  $0025 ;vertik.Cursor
BASL      EQU  $0028 ;Basis-Adresse
BASH      EQU  $0029
BAS2L     EQU  $002A ;Scroll-Basis-A.
BAS2H     EQU  $002B
INVFLG    EQU  $0032 ;Inverseflag

Eingabe
-------
PROMPT    EQU  $0033 ;"Prompt"
KSWL      EQU  $0038 ;Eingabe-Vektor
KSWH      EQU  $0039
RNDL      EQU  $004E ;Random-Wait
RNDH      EQU  $004F

Ausgabe
-------
CSWL      EQU  $0036 ;Ausgabe-Vektor
CSWH      EQU  $0037

Monitorbefehle
--------------
LMNEM     EQU  $002C ;l.Mmemonic
RMNEM     EQU  $002D ;r.Mnemonic
FORMAT    EQU  $002E ;Opcode-Format
LENGTH    EQU  $002F ;Befehlslänge
MODE      EQU  $0031 ;Befehlsmodus
YSAV      EQU  $0034 ;Inputposition
YSAV1     EQU  $0035 ;Y-Register
PCL       EQU  $003A ;Programmzähler
PCH       EQU  $003B
A1L       EQU  $003C ;Hilfsregister
A1H       EQU  $003D
A2L       EQU  $003E ;Hilfsregister
A2H       EQU  $003F
A3L       EQU  $0040 ;Hilfsregister
A3H       EQU  $0041
A4L       EQU  $0042 ;Hilfsregister
A4H       EQU  $0043
A5L       EQU  $0044 ;Hilfsregister
A5H       EQU  $0045

Interrupt, Break
----------------
MACSTAT   EQU  $0044 ;Maschinestatus
ACC       EQU  $0045 ;A-Register
XREG      EQU  $0046 ;X-Register
YREG      EQU  $0047 ;Y-Register
STATUS    EQU  $0048 ;Status-Reg.
SPNT      EQU  $0049 ;Stack-Pointer

F8-Routinen
===========

Lores
-----
SETGR     EQU  $FB40
CLRSCR    EQU  $F832 ;l
CLRTOP    EQU  $F836 ;l
PLOT      EQU  $F800 ;l
HLINE     EQU  $F819 ;l
VLINE     EQU  $F828 ;l
SETCOL    EQU  $F864 ;l
NXTCOL    EQU  $F85F ;l
GBASCALC  EQU  $F847
SCRN      EQU  $F871 ;l

Text
----
INIT      EQU  $FB2F
SETTXT    EQU  $FB39
SETWND    EQU  $FB4B

Monitorbefehle
--------------
INSTDSP   EQU  $F8D0
INSDS2    EQU  $F88C ;s
PCADJ     EQU  $F953 ;s
XAM       EQU  $FDB3
MOVE      EQU  $FE2C ;l
VERIFY    EQU  $FE36 ;l
LIST      EQU  $FE5E
REGDSP    EQU  $FAD7
RGDSP1    EQU  $FADA
RESTORE   EQU  $FF3F ;l; IOREST
SAVE      EQU  $FF4A ;l; IOSAVE
NXTA4     EQU  $FCB4 ;s
NXTA1     EQU  $FCBA ;s
LT        EQU  $FE20 ;s
PRA1      EQU  $FD92 ;s
ZMODE     EQU  $FFC7 ;s

Daten, Tabellen, Vektoren
-------------------------
FMT1      EQU  $F962 ;w
TITLEALT  EQU  $FB09 ;w
TITLENEU  EQU  $FF0A ;w
VERSION   EQU  $FBB3 ;l
ZIDBYTE   EQU  $FBC0 ;l
CHRTBL    EQU  $FFCC ;w; c:$FFCD!
NMIVECT   EQU  $FFFA ;w ;nicht c!
RESETVEC  EQU  $FFFC ;w
IRQVECT   EQU  $FFFE ;w

Interrupt, Break, Reset
-----------------------
IRQ       EQU  $FA40 ;w
NEWBRK    EQU  $FA47 ;w (n/c)
BREAK     EQU  $FA4C ;w
OLDBRK    EQU  $FA59
RESET     EQU  $FA62
NEWMON    EQU  $FA81
PWRUP     EQU  $FAA6
SETPWRC   EQU  $FB6F
OLDRST    EQU  $FF59
MON       EQU  $FF65
MONZ      EQU  $FF69
XBASIC    EQU  $FEB0 ;$E000
BASCONT   EQU  $FEB3 ;$E003

Bildschirmsteuerung 40/80 Z/Z
-----------------------------
TABV      EQU  $FB5B
VTAB      EQU  $FC22
VTABZ     EQU  $FC24 ;l
VTAB23    EQU  $FB59
HOME      EQU  $FC58 ;l
CLREOP    EQU  $FC42 ;l
CLREOL    EQU  $FC9C ;l
CLEOLZ    EQU  $FC9E ;l
SCROLL    EQU  $FC70
BASCALC   EQU  $FBC1
SETINV    EQU  $FE80 ;l
SETNORM   EQU  $FE84 ;l

Bildschirmsteuerung nur 40 Z/Z
------------------------------
ADVANCE   EQU  $FBF4
BS        EQU  $FC10
UP        EQU  $FC1A
LF        EQU  $FC66
CR        EQU  $FC62
STORADV   EQU  $FBF0  ;s
VIDWAIT   EQU  $FB78  ;s

Eingabe
-------
SETKBD    EQU  $FE89
INPORT    EQU  $FE8B
RDKEY     EQU  $FD0C ;l; auch DOS
KEYIN0    EQU  $FD18
KEYIN     EQU  $FD1B ;l
RDCHAR    EQU  $FD35 ;l
GETLN     EQU  $FD6A ;l
GETLNZ    EQU  $FD67 ;l
GETLN1    EQU  $FD6F ;l
GETNUM    EQU  $FFA7 ;s
PREAD     EQU  $FB1E ;l
READ      EQU  $FEFD ;l; nicht c

Ausgabe
-------
SETVID    EQU  $FE93
OUTPORT   EQU  $FE95
COUT      EQU  $FDED ;l; auch DOS
COUT1     EQU  $FDF0 ;l; n/c auch 80Z/Z
COUTZ     EQU  $FDF6
CROUT     EQU  $FD8E ;l
CROUT1    EQU  $FD8B ;l
PRBYTE    EQU  $FDDA ;l
PRHEX     EQU  $FDE3 ;l
PRNTYX    EQU  $F940
PRNTAX    EQU  $F941 ;l
PRBLNK    EQU  $F948 ;l
PRBL2     EQU  $F94A ;l
APPLEII   EQU  $FB60
PRERR     EQU  $FF2D ;l
BELL      EQU  $FF3A ;l
BELL1     EQU  $FBD9 ;c:CHKBELL
BELL1A    EQU  $FBDD ;l; c:BELL1
WAIT      EQU  $FCA8 ;l
WRITE     EQU  $FECD ;l; nicht c

Slot-Routinen (nicht II+)
-------------------------
C3ENTRY   EQU  $C300 ;l; e/n/c
MOVEAUX   EQU  $C311 ;l; e/n/c
XFER      EQU  $C314 ;l; e/n/c
```

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	à DM	gesamt DM

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

_____
Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

PEEKER

Info-Karte

Schreiben Sie uns, wenn Sie technische Fragen zu Aufsätzen im **Peeker** haben. Beachten Sie jedoch, daß die Redaktion keine Auskünfte über Bezugsquellen erteilt:

_____

_____

_____

_____

_____

_____

Peeker-Börse

Vorname, Name

Firma

Straße

Wohnort

PLZ/Ort

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

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

Datum Unterschrift

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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

Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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 Anfrag

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

PEEKER

Handmac

Ein Macintosh-Paket für den Handwerker

von Harald Grumser

Das deutsche Software-Paket Handmac von Copy Team Software, Bamberg, wurde speziell für den kleinen und mittleren Handwerksbetrieb entwickelt und bietet zu einem Preis von DM 2498,- (außer Kunden-Stammdaten) fast alle Module, die im handwerklichen Bereich benötigt werden. Die Handhabung ist wegen der Mac-typischen Benutzerführung auch für einen Computer-Laien schnell erlernbar.
Die Anwendung läßt sich in fünf Bereiche gliedern, die untereinander Daten austauschen können:

– Lagerverwaltung
– Kalkulation
– Abrechnung
– Rechnungsstellung
– Textverarbeitung

Darüber hinaus gibt es einen weiteren Programmteil, der zur Ablage und Organisation (Erstellen von Sicherheitskopien, Auslagern von Dateien) dient.

Lagerverwaltung

Die Lagerverwaltung kann insgesamt 3000 Posten verwalten. Diese einzelnen Artikel werden nicht, wie üblich, durch Artikelnummern bezeichnet, sondern durch Kategorie und Kurzbezeichnung beschrieben. Somit erhält die Dachlatte, 5 * 7 cm nicht die Artikelnummer 1234567, sondern wird z.B. unter der Kategorie „Holz", Kurzbezeichnung „Dachl. 5 * 7" wiedergefunden.
Diese Lagerverwaltung ist in einem weiteren Sinn zu sehen, da die einzelnen Artikel nicht in jedem Fall materielle Güter sein müssen, sondern auch Posten wie Maschinenstunden oder Lohnkosten bezeichnen können (was sich bei der Fakturierung als günstig erweist), für die jedoch kein Vorrat oder Minimum angegeben werden kann.
Die Stammdaten der einzelnen Artikel enthalten neben der ausführlichen Bezeichnung den Ein- und Verkaufspreis, den Steuersatz, die Einheit (z.B. m, kg) und ggf. die Bestellnummer.

Über die Option Wareneinund ausgang wird der Lagerbestand neben dem automatischen Abgleich bei der Abrechnung aktualisiert. Über „Inventur" kann die Lagerbestandsliste in regelmäßigen Abständen auf den neusten Stand gebracht werden, um u.a. den erfahrungsgemäß auftretenden Schwund zu erfassen.
Wie bei den anderen Punkten ist es auch hier möglich, eine Liste nach bestimmten Gesichtspunkten auszudrucken, um z.B. alle Artikel zu erhalten, deren Mindestbestand unterschritten ist.

Kalkulation

Bei der Kalkulation werden verschiedene Posten zusammengetragen, um die voraussichtlichen Gesamtkosten eines Auftrags zu ermitteln. Dabei erhält jede Kalkulation einen Namen und ggf. eine Einheit mit Steuersatz, um sie später wiederum als Posten einer Gesamtkalkulation aufzunehmen.
Die Berechnung kann sowohl als Vollkostenrechnung als auch als Deckungsbeitragsrechnung erfolgen:
– Bei der Vollkostenrechnung werden die absoluten Beträge (also z.B. Verkaufspreis der Einzelposten) zugrunde gelegt.
– Bei der Deckungsbeitragsrechnung wird der erwünschte Gewinn angegeben (absolut oder prozentual), und die Berechnung erfolgt unter Brücksichtigung von sonstigen Kosten (die jeweils angegeben werden) nach dieser Vorgabe.
Neben den Posten, die bereits durch die Lagerverwaltung bekannt sind oder aus anderen Kalkulationen entnommen werden können, werden andere Einzelposten (z.B. Artikel, die nicht auf Lager gehalten werden) direkt angegeben, wie auch die Preise der Lagerstammdaten, die sich ausnahmsweise geändert haben.

Abrechnung

Die Abrechnung kann, falls eine Kalkulation erstellt wurde, mit der entsprechenden Anpassung an die tatsächlich erbrachte Leistung von dort abgeleitet oder neu erstellt werden. Dabei besteht die Möglichkeit, die verbrauchten Artikel vom Lager abzubuchen.
Neben diesen Daten müssen hier noch die Kundendaten eingegeben werden, was laut Herstellerangaben später durch ein weiteres Kundenstamm-Modul ergänzt werden soll.

Rechnungsstellung

Grundlage der Rechnungsstellung ist die Abrechnung. Hier können Abrechnungen oder gleiche Posten (5 m Dachlatten und 7 m Dachlatten werden 12 m Dachlatten) zusammengefaßt werden. Außerdem bietet sich die Möglichkeit, einen Rabatt zu gewähren.
Beim Bearbeiten der Rechnung wird festgelegt, ob diese bereits bezahlt wurde. Bei der Ausgabe des Rechnungsjournals (Drucker oder Bildschirm), das alle Rechnungen in Kurzform auflistet, sieht man dann alle offenen Posten.

Textverarbeitung

Die Textverarbeitung bietet die Möglichkeit, Briefe zu schrieben, die Kalkulationen, Abrechnungen oder Rechnungen enthalten können. Diese Dokumente lassen sich dann entsprechend ändern oder einfach nur anders gestalten (Schriftgrößen, Unterstreichen, Fettdruck). So ist es möglich, aus Standart-Modulen Angebotsschreiben, Zwischenabrechnungen oder Mahnungen zu erstellen.

Inserentenverzeichnis Peeker 5/86

Erscheinungs- und Anzeigenschlußtermine für Peeker

Ausgabe	Erstverkaufstag	Anzeigenschluß
6	6.06.86	05.05.86
7	23.06.86	02.06.86
8	21.07.86	01.07.86
9	25.08.86	04.08.86
10	22.09.86	01.09.86
11	20.10.86	01.10.86
12	24.11.86	03.11.86

MS-File, MS-Word und Imagewriter II

Anspruch und Wirklichkeit im Büroeinsatz

von Dr. Martin Wolter

Da das, was ich im folgenden beschreiben will, wahrscheinlich in manchen Ohren nicht sehr freundlich klingt, möchte ich einigen möglichen Einwänden von vornherein begegnen. Zunächst: Ich liebe meinen Macintosh heiß und innig. Ich habe ihn heranwachsen sehen, von ursprünglich 128K auf 512K, und kann es kaum erwarten, die volle 1M-Grenze zu überspringen. Er hat mir immer treu gedient und hat meine Augen nie zum Tränen gebracht, auch wenn ich Stunden vor seinem kleinen Bildschirm gesessen habe. Mit Großrechnern und ihren Betriebssystemen habe ich im Laufe der Jahre meine Erfahrungen gesammelt. Ich möchte nicht mehr zurück in die schlechte alte Zeit. Aber ich bin auch nicht betriebsblind.

Mein Macintosh war eines der ersten in Deutschland verfügbaren Geräte, es war Liebe auf den ersten Blick. Seit es ihn hier gibt, habe ich mit seiner Hilfe viele, viele Seiten Text geschrieben, Zeichnungen erstellt, Statistiken ausgewertet, aber auch auf Monster geschossen. Ich habe gern Adventure gespielt: Wenn man mit Hilfe diverser Editoren neue Eigenarten und verborgene Schätze des Betriebssystems oder undokumentierter Programme findet, ist das schon von eigenem Reiz. Ebenso wie die Programmierung des Macintosh selbst. Allen Unkenrufen zum Trotz, vor allem aus dem Hause Apple, wonach man ohne die Lisa niemals Programme auf dem Macintosh werde entwickeln können – mit der Programmierung geht es doch.

Als mich ein guter Bekannter bat, ihm im Geschäft bei der Umstellung der EDV auf Mac zu helfen, habe ich sofort zugesagt. Damals habe ich noch geglaubt, das sei mit meinen Erfahrungen eigentlich im Handumdrehen zu erledigen. Im Laufe dieses Abenteuers entpuppte sich das Problem als ein bösartiger, widerspenstiger Drachen; kaum ist ein Kopf abgeschlagen, wachsen viele neue nach.

Bisher wurde in dem Geschäft die Adressenliste auf einem CP/M-System und dem bewährten, aber betagten Wordstar erledigt. Teilweise werden Etiketten und Paketaufkleber auch mit einer Speicherschreibmaschine bedruckt das Überspielen der Adressenliste auf den Macintosh warf keine Probleme auf. Die überspielten Daten ließen sich anschließend sehr übersichtlich in MS-File organisieren. MS-File ist eines meiner Lieblingsprogramme auf dem Macintosh, sehr angenehm (maclike eben) zu bedienen, und zudem ist es leistungsfähig. Immer mit der Einschränkung, daß es sich um keine vollwertige Datenbank mit allen Schikanen handelt. Bis hier eine Sache von wenigen Stunden

Irgendwann muß man die Daten auch einmal ausdrucken, das ist nun einmal in einem Unternehmen so. In einem Geschäftsbetrieb ist Zeit gleich Geld. Aber das Ausdrucken mit einem Macintosh kostet viel Zeit, sehr viel Zeit.

Also ganz langsam. Die Adressenliste umfaßt mehr als 1000 Einträge, da hört der Spaß auf und das Programm muß hart arbeiten. Die Formatmaske in MS-File bietet ein klares, sauberes Layout und ist so aufgebaut, daß sechs Kundeneinträge auf eine Seite passen. Gedruckt werden soll auf dem neuen Imagewriter II. Drucke ich im **Draft-Modus,** geht es zwar zügig voran, aber das ganze, feine Layout ist für die Katz. Im **Standard-Modus** bleiben die begrenzenden Linien und fett zu druckenden Texte zwar so, wie vorgesehen, nur mit dem kleinen Manko, daß für eine Seite ca. 100 Sekunden benötigt werden. Das bedeutet bei 1000 Kunden knapp fünf Stunden Druckzeit. Fünf Stunden, die ein bezahlter Mitarbeiter damit zubringt, darauf zu achten, daß sich der Drucker nicht verheddert. Was er ab und an tut. Wielange sich das Drucken ausdehnt, wenn gar im **Qualitäts-Modus** gedruckt würde, habe ich erst gar nicht errechnet.

Ist denn die Druckqualität wenigstens „Standard"? Nein, sie ist es nicht. Der Imagewriter II druckt bidirektional und ist immer noch nicht teuer genug, um gerade senkrechte Linien senkrecht bleiben zu lassen. Er hat auch sonst mit dem akuraten Druck so seine Schwierigkeiten. Der alte Imagewriter ist langsamer, billiger und druckt besser (nicht gut). Daß es sich nicht um ein Gerät aus einer Montagsserie handelt, zeigen Versuche auf zwei weiteren Imagewriter-II-Geräten beim Händler. Übrigens hat der Händler auch einen Joyce von Schneider im Angebot, mit einem Matrixdrucker vom Feinsten in bezug auf die Druckqualität. Nur daß ich dann für den Preis eines Imagewriter II einen Drucker erhalte, aber mit einem Computer obendrein.

Was den Mac zu einer so phantastischen Maschine macht, daß er nämlich grundsätzlich Grafik druckt (wenn er maclike agiert), macht ihn für seriöse Aufgaben fast unbrauchbar. Derart lange Druckzeiten sind indiskutabel. Die Ausgabe auf einem schnellen Typenraddrucker würde die Arbeit erheblich beschleunigen, nur mit dem Handikap, daß dies von MS-File aus nicht geht und zudem sämtliche Formatierungen hinfällig wären.

Im Geschäft sind weiterhin Etiketten, diverse Aufkleber, Post- und Paketkarten und Rechnungen zu schreiben. Also wird die Kombination MS-File und MS-Word bemüht. An sich ein ideales Paar. Von MS-File lassen sich die benötigten Records bequem und rasch selektieren, im Print-Merge-Format übergeben und mit hoher Flexibilität in MS-Word weiterbehandelt. Bis dahin ließ sich eine wunderschöne Lösung erstellen, die wohl auf keinem anderen Rechner so benutzerfreundlich abliefe. Wenn nur der leidige Ausdruck nicht wäre! Und wieder der Imagerwriter.

Mit einem Einzelblatteinzug zieht der Imagewriter II das erste Blatt grundsätzlich auf eine andere Höhe ein als das zweite und jedes folgende. Bei ca. 10 verschiedenen Formulartypen eine sehr lästige Eigenschaft, die weder in der Werbung noch im Handbuch dokumentiert ist. Um auch das erste Blatt immer auf der gleichen Höhe zu bedrucken, muß das Papier mit viel Gefühl eingeführt werden. Nun ja, ein neuer Mitarbeiter wird es im Laufe weniger hundert Blätter und Wochen schon lernen.

Eines der Formulare muß dicht unter dem Rand bedruckt werden: Das geht grundsätzlich bei Einzelblatteinzug nicht. Erst ca.

2.5 cm unterhalb des Papieranfangs läßt sich tatsächlich drucken. Eine „sinnreiche" Automatik im Imagerwriter II verhindert, daß man mit einem eigenen Programm das Papier auf die richtige Höhe zurückfährt. Darüber hinaus muß für jedes Formular eine etwas veränderte Einstellung gewählt werden. Selbstverständlich könnte man auch auf Endlosformulare drucken. Volkswirtschaftlich wäre die Investition in ein knappes Dutzend Drucker nur zu begrüßen. Der Versuch, mehreren vorhandenen Druckern verschiedene Aufgaben zuzuweisen und mit dem Microsoft-Enhancer umzuschalten, scheiterte bislang, weil Microsofts sogenannte Hotline innerhalb von sechs Wochen auch nicht weiterhelfen konnte. Ein neuer Enhancer jedenfalls „ist schon unterwegs".

Eine weitere Schwierigkeit beim Bedrukken von Formularen tritt auf: Ich möchte gern die entsprechenden Felder millimetergenau bedrucken. Das „millimetergenau" nehme ich, speziell bei Einzelblatteinzug, nicht mehr ganz wörtlich. Lasse ich MS-Word mit den vollen Fähigkeiten des Macintosh arbeiten, dann kann ich verschiedene Schriftarten, unterschiedliche Schriftgrößen und Darstellungen (fett, kursiv usw.) drucken. Das geht bei einem leeren, weißen Blatt Papier auch gut, da mir der Zeilenabstand ziemlich gleichgültig ist. Aber die kleinste Veränderung verschiebt alle folgenden Zeilen. Abhilfe: Einstellen eines festen Zeilenabstands. Jetzt kann ich mich darauf verlassen, daß eine Änderung in der ersten Zeile nicht die restlichen Einstellungen und Abstände unbrauchbar macht. Ebenso, wie auf einer ganz gewöhnlichen Schreibmaschine. Aber die ganze Macintosh-Flexibilität ist zum Teufel. Sicher könnte man im Laufe vieler Stunden auch MS-Word dazu bringen, „millimetergenau" zu arbeiten und ein Formular genau auf der anvisierten Linie zu bedrucken. Der benötigte Aufwand stünde aber kaum noch in einem akzeptablen Verhältnis zum Ergebnis, ganz abgesehen davon, daß mein spezielles Formular von nun an heilig wäre, weil mögliche Änderungen unabsehbare Folgen nach sich ziehen würden.

Die Verwaltung großer Datenmengen erfordert eine Harddisk. Apples Harddisk kommt mit zwei Jahren Verspätung, zeichnet sich aber immerhin dadurch aus, langsam und vergleichsweise teuer zu sein. Zum Glück ist im Geschäft eine Hyperdrive mit 20M installiert. Dem Gerät können akzeptable Antwortzeiten bescheinigt werden. Mit einer Hyperdrive macht es Spaß, auf dem Mac zu arbeiten.

Großes Finale: Die bisherige Standardlösung des Etikett- und Formulardrucks mit einer Speicherschreibmaschine erfordert ca. **vier Minuten** Aufwand. Die Lösung auf dem Macintosh zeichnet sich dadurch aus, vergleichsweise benutzerfreundlich zu sein. Was später einmal mit dem Programm „Tempo" geschehen soll, nämlich per Tastendruck eine Aufgabe komplett abwickeln zu können, muß jetzt noch per Hand gemacht werden. Das heißt konkret, daß innerhalb einer Aufgabe etwa 20mal verschiedene Menüs ausgewählt werden müssen, die man sich erst einmal merken muß (um gerecht zu sein, es wäre mit CP/M und Wordstar um einiges unangenehmer). Nun denn, ich bin geübt, habe die abkürzenden Tastaturkommandos parat und lasse meine Finger tanzen. Und bin immerhin mit **sechs Minuten** guter Zweiter. Gegenüber einer Speicherschreibmaschine brauche ich mit dem fortschrittlichen Mac nur knapp 50 Prozent länger.

Eigentlich ist der Macintosh eine tolle Maschine. Aber wenn er einmal wirklich auf größere Probleme losgelassen wird, geht ihm die Puste sehr schnell aus. Eine Kette ist nur so stark wie ihr schwächstes Glied. Nach zwei Jahren Einsatz des Macintosh im Uni-, Heim- und Hobbybereich mag ich meinen Mac immer noch. Aufgrund meiner gesammelten Erfahrungen im Bürobereich wundere ich mich allerdings darüber, daß der Macintosh tatsächlich als Managercomputer vermarktet werden soll. Luxuspreis = Luxusgerät = Luxuslösung, diese Rechnung jedenfalls geht nicht auf.

Microsoft und Microservice

Normalerweise hätten wir den Beitrag von Dr. Wolter an die Firma Microsoft zur Stellungnahme weiterleiten müssen, aber Microsoft schweigt sich nach unseren Erfahrungen bei technischen Anfragen aus. Wir haben dies bereits mit der Microsoft-Premium-Softcard durchexerziert. Nach unserem Testbericht im Peeker, Heft 11/85 gingen diverse Leserbriefe ein, insbesondere wegen der Anpassung der Premium-Softcard an 80-Spurlaufwerke und RAM-Karten. Für die eingehende Post gilt bei Microsoft offenbar das Motto „Die guten ins Töpfchen, die schlechten ins Kröpfchen": Die „guten" (= Bestellungen) werden sofort bearbeitet, die „schlechten" (= Rückfragen) später oder gar nicht. Wir mußten deshalb leider den von einem Peeker-Leser eingereichten Aufsatz „Inside Microsoft Premium Softcard" ablehnen, weil dieser Beitrag neue Briefe provoziert hätte, die in Aschheim-Dornach wiederum ins „Kröpfchen" gewandert wären.

Bei der Premium-Softcard kann man vielleicht noch unterstellen, daß diese Karte trotz des stolzen Preises nur für Freaks konstruiert worden ist. Wie steht es aber mit MS-File und MS-Word? Sind auch diese Produkte nur für Programmierer gedacht, von denen man erwartet, daß sie sich ihren eigenen Printer-Driver schreiben können? Der Bericht von Dr. Wolter behandelt eines der typischen „Zwischen-den-Stühlen"-Probleme. Der Käufer eines Text- oder Dateiprogramms will seine Texte oder Daten schließlich irgendwann einmal ausdrucken, wie umgekehrt der Besitzer einer Premium-Softcard auch irgendwann einmal seine RAM-Karte einsetzen möchte. Microsoft kann sich natürlich auf den Standpunkt stellen, daß Apple gefälligst den Imagewriter an MS-Word anzupassen habe, während umgekehrt Apple die Meinung vertreten kann, daß Microsoft bitte schön MS-Word an den Imagewriter zu adaptieren habe. Bei dieser starren Haltung – Apple hilft nicht bei Word, Microsoft hilft nicht beim Imagewriter – bleibt indessen der Anwender auf der Strecke. Daß man bei MS-File 5 Stunden benötigt, um 1000 Adressen (also nicht 1000 Briefe o.ä.) auszudrucken, ist eine Armutszeugnis für eine solch renommierte Firma wie Microsoft. Und daß sie ihren Kunden dann nicht einmal hilft, ist noch beschämender. Die Ware wurde geliefert, die Rechnung bezahlt, und das war's!

Ulrich Stiehl

von Dr. Jürgen B. Kehrel

Erphi-Patch für Superquick

Jetzt auch 160 Tracks in Windeseile kopiert

Das Superquick-Programm von Arne Schäpers ist ein sehr schnelles und sicheres Kopierprogramm für ganze Disketten unter DOS 3.3, ProDOS, Pascal und CP/M. Für mich hatte es nur einen Fehler: Meine 640K-Disketten mit 160 Tracks am Erphi-Controller ließen sich nicht bearbeiten. Ein Patch macht auch dies jetzt möglich.

Gerade für die „großen" Disketten ist ein schnelles Kopierprogramm wichtig. Wenn Sie eine Diskette erst formatieren und dann die Files einzeln mit NEWFIDM kopieren, wie dies bei Erphi-Besitzern bisher üblich war, dauert das bei einer vollen Diskette 15 Minuten oder gar mehr. Das gepatchte Superquick ist mit 2 Laufwerken und 128K Apple IIe in 110 Sekunden fertig.

Der Erphi-Controller benutzt ein unübliches Aufzeichnungsverfahren: Alle Tracks mit gerader Spurnummer liegen auf der Seite 0 (unten), alle Tracks mit ungerader Spurnummer auf der Seite 1 (oben). Der große Vorteil ist, daß dadurch im Normalbetrieb weniger Kopfbewegungen notwendig sind. Müssen Sie beispielsweise ein File von den Tracks 4 bis 7 lesen, so sind nur 2 Kopfbewegungen notwendig: 1. Bewegung hin zu Track 4, Lesen der Tracks 4 und 5, 2. Kopfbewegung zu Track 6, Lesen der Tracks 6 und 7. Bei einem anderen Controller sind 4 Bewegungen notwendig. Der Nachteil: Kein Kopierprogramm beherrscht diese Technik.

Mit ein paar Ergänzungen kann Superquick dazu gebracht werden, nur noch bei einem Übergang von einem Track mit ungerader zu einem Track mit gerader Spurnummer den Kopfschlitten zu bewegen. Bei jedem Trackwechsel muß aber der aktive Kopf gewechselt werden. Es würde an dieser Stelle zu weit führen, alle Einzelheiten der Programmierung zu besprechen. Es wurde ein Applesoft-Programm entwickelt, das Ihnen alle Patcharbeit abnimmt. Sie brauchen nur den Bildschirmanweisungen zu folgen.

Handhabung

Tippen Sie das Applesoft-Programm ab. Speichern Sie es unter dem Namen SQ.PATCHER auf einer DOS-3.3- oder ProDOS-Diskette. Benutzen Sie ProDOS, wenn Sie eine Kopie Ihres Original-Superquick patchen wollen. Benutzen Sie DOS 3.3, wenn Sie Superquick mit CONVERT oder PROTODOS konvertiert haben. Es ist wichtig, daß das Patch-Programm im selben Diskettenformat vorliegt wie Superquick! Geben Sie die Zeilen von PATCH-.STARTER entweder direkt über die Tastatur ein (mit <RETURN> nach jeder Zeile) oder schreiben Sie sich einen EXEC-File mit diesem Inhalt. Folgen Sie den Bildschirmanweisungen.

Starten Sie SQ.PATCHER niemals direkt ohne den PATCH.STARTER, da dann Teile des BASIC-Programms überschrieben werden.

Es entsteht ein neues Programm EQUICK, mit dem Sie nun 160-Track Disketten am Erphi-Autopatch-Controller AFDC kopieren können. Die Handhabung ist gegenüber dem Original unverändert. Sie können allerdings die Parameter für die Trackanzahl nicht mehr ändern. Auch sollten Sie das Installations-Programm SQUICK.INSTALL für EQUICK nicht benutzen. EQUICK wurde so eingestellt, das sowohl die Language-Card als auch die zweite 64K-Bank des IIe, falls vorhanden, als auch IBS-RAM-Karten bei Vorhandensein benutzt werden. Der Bereich von DOS 3.3 ist nach wie vor geschützt und kann nur über die <P>-Option mitbenutzt werden. Je mehr Speicher Sie haben, desto schneller geschieht die Kopie.

EQUICK hat wieder eine Kaffee-Pause aus Ihrem Computeralltag weg„rationalisiert".

Hinweise zur Peeker-Sammeldisk

Die Sammeldisk enthält die Programme PATCH.STARTER und SQ.PATCHER, die zuvor mit DOSTOPRO auf eine *Kopie* der Superquick-Programmdiskette konvertiert werden müssen. Oder man kopiert umgekehrt die Superquick-Dateien auf eine DOS-Diskette mittels PROTODOS. Danach EXEC PATCH.STARTER, fertig. Alle seit dem 1.3.1985 ausgelieferten Superquick-Disketten enthalten bereits das für 160-Track-Erphi-Laufwerke modifizierte EQUICK. Altbesteller, die keine Sammeldiskettenbezieher sind, können die geänderte Programmdiskette kostenlos anfordern, wenn sie 160-Spur-Erphi-Laufwerke besitzen. Es sein noch darauf hingewiesen, daß der Programmautor Arne Schäpers mangels Erphi-Laufwerk den Patch nicht selbst schreiben konnte.

PRODOS.PATCH.1.1.1
(Bezieht sich auf Seite 60: „Kyan-Club-Nachrichten")

```
$0900: A9 01 D0 05 20 B6 09 A9
$0908: 02 8D F7 09 20 E3 03 84
$0910: CE 85 CF A0 01 B1 CE 8D
$0918: EC 09 8D FA 09 C8 B1 CE
$0920: 8D ED 09 8D FB 09 A9 10
$0928: 85 E3 A9 00 8D EF 09 20
$0930: 62 09 A9 20 85 E3 EE EF
$0938: 09 20 62 09 A9 30 85 E3
$0940: EE EF 09 20 62 09 A9 40
$0948: 85 E3 EE EF 09 20 62 09
$0950: A9 50 85 E3 EE EF 09 20
$0958: 62 09 AD F7 09 C9 01 F0
$0960: 30 60 A2 10 86 D7 C6 D7
$0968: A6 D7 10 01 60 8E F0 09
$0970: 18 BD DB 09 65 E3 8D F4
$0978: 09 A9 09 A0 EB 20 D9 03
$0980: A9 00 85 48 90 E0 68 68
$0988: 20 58 FC AD F8 09 4C DA
$0990: FD A0 00 B9 00 20 AA B9
$0998: 00 1E 99 00 20 8A 99 00
$09A0: 1E C8 D0 EF B9 00 21 AA
$09A8: B9 00 1F 99 00 21 8A 99
$09B0: 00 1F C8 D0 EF 60 A0 00
$09B8: B9 00 1E AA B9 00 20 99
$09C0: 00 1E 8A 99 00 20 C8 D0
$09C8: EF B9 00 1F AA B9 00 21
$09D0: 99 00 1F 8A 99 00 21 C8
$09D8: D0 EF 60 00 0E 0D 0C 0B
$09E0: 0A 09 08 07 06 05 04 03
$09E8: 02 01 0F 01 60 01 00 00
$09F0: 00 FC 09 00 10 00 00 01
$09F8: 00 00 60 01 00 01 EF D8
```

PATCH.STARTER

Über die Tastatur eingeben oder als Textfile speichern und mit
EXEC PATCH.STARTER
laden.

```
POKE 103,1
POKE 104,64
POKE 16384,0
LOAD SQ.PATCHER
CALL 54514
RUN
```

SQ.PATCHER

```
10 REM *** PATCHER FUER SUPERQUICK
11 REM *** 160 TRACK ERPHI-CONTROLLER AFDC 2
12 REM *** DR. JUERGEN B. KEHREL 1986
20 HOME : PRINT "LEGEN SIE DIE DISKETTE MIT"
21 PRINT "SUPERQUICK IN LAUFWERK 1 UND"
22 PRINT "DRUECKEN SIE DANN EINE TASTE."
23 POKE - 16368,0: WAIT - 16384,128,0
24 PRINT CHR$ (4);"BLOAD SQUICK,A$AC3,D1"
25 RESTORE : POKE 4841,4
26 FOR I = 0 TO 2: READ P%: POKE 4844 + I,P%: NEXT
27 DATA 176,247,96
28 FOR I = 0 TO 5: READ P%: POKE 4868 + I,P%: NEXT
29 DATA 160,255,0,0,0,0
30 FOR I = 0 TO 4: READ P%: POKE 5306 + I,P%: NEXT
31 DATA 240,9,206,20,19
32 POKE 5737,73: POKE 5738,24: POKE 6022,74
33 POKE 6029,74: POKE 6034,40: POKE 6035,24
34 FOR I = 0 TO 38: READ P%: POKE 6184 + I,P%: NEXT
35 DATA 32,157,23,174,26,19,173,20,19,41,1,240,10
36 DATA 221,136,192,221,141,192,221,137,192,96,221
37 DATA 136,192,221,140,192,221,137,192,96,32,154,22,76,43,24
38 POKE 6661,48: POKE 6662,6
39 FOR I = 0 TO 9: READ P%: POKE 7304 + I,P%: NEXT
40 DATA 212,242,225,227,235,243,186,177,182,176
41 FOR I = 0 TO 3: READ P%: POKE 7637 + I,P%: NEXT
42 DATA 76,171,30,234
43 POKE 7984,208: POKE 7985,24
44 FOR I = 0 TO 28: READ P%: POKE 8212 + I,P%: NEXT
45 DATA 177,182,176,160,212,242,225,227,235,243,186,160
46 DATA 160,197,242,240,232,233,173,195,239,238,244
47 DATA 242,239,236,236,229,242
48 FOR I = 0 TO 33: READ P%: POKE 8244 + I,P%: NEXT
49 DATA 199,229,240,225,244,227,232,229,228,160,246
50 DATA 239,238,186,160,196,242,174,160,202,245,229,242
51 DATA 231,229,238,160,203,229,232,242,229,236,160
60 PRINT : PRINT "BITTE LEGEN SIE DIE DISKETTE"
61 PRINT "EIN, AUF DIE EQUICK GESPEICHERT"
62 PRINT "WERDEN SOLL. DRUECKEN SIE DANN"
63 PRINT "EINE BELIEBIGE TASTE."
64 POKE - 16368,0: WAIT - 16384,128,0
65 PRINT CHR$ (4);"BSAVE EQUICK,A$AC3,L$1830"
66 PRINT : INVERSE : PRINT "FERTIG": NORMAL : END
```

Kyan-Club-Nachrichten

Der exakte Versandtermin stand zum Redaktionsschluß noch nicht fest, doch dürfte Ihr 2.0-Päckchen mit 350seitigem Handbuch im Ringordner, Reference-Chart und zwei beidseitig bespielten Disketten (mit kompletem ProDOS 1.1.1 einschließlich BASIC.SYSTEM 1.1!), wenn Sie diese Zeilen lesen, gerade unterwegs oder vielleicht schon bei Ihnen eingetroffen sein. Übrigens: Interessenten können auch jetzt noch Club-Mitglieder werden. Der Preis von DM 170,- für Kyan 2.0 bleibt unverändert, doch wird die Version 1.2 jetzt nicht mehr verschickt, da damit erzeugte *Assembler*-Routinen mit 2.0 nicht kompatibel sind. Unsere erste Club-Diskette (Club-Preis DM 20,-), die Sie bei Bedarf anfordern können, ist in Vorbereitung und enthält neben Anfängerprogrammen diverse Assembler-Include-Files. Näheres im Kyan-Club-Rundschreiben. Die Kyan-Pascal-Toolkits, aus denen Sie je nach Interesse und Bedarf auswählen können, erscheinen wahrscheinlich erst im Juni:
1. Mouse-Text-Toolkit $48.00
2. Advanced-Graphics-Toolkit $48.00
3. Programming-Utility-Toolkit $48.00
Club-Preis je Toolkit nur DM 120,-. Näheres zum Inhalt im Kyan-Club-Rundschreiben. Achtung: Unsere Rundschreiben mit Tips und Tricks, Mitgliedslisten und sonstigen Infos erhalten nur diejenigen Besteller, die „Club ja" vermerkt haben!

1. ProDOS-1.1.1-Kompatiblen-Patch

Wie bereits bekannt, läuft Kyan-Pascal auf manchen Kompatiblen nicht. Da Arne Schäpers kurzfristig erkrankt ist, habe ich den nachfolgenden Patch für ProDOS 1.1.1 selbst schreiben müssen. Die S(eitenangaben) beziehen sich auf seine „ProDOS-Analyse". Im übrigen funktioniert der Patch bei jeder „1.1.1-Users.Disk" mit „PRODOS" (30 Blocks, 18-Sept-84) als *erster* Datei auf einer normalen 1.1.1-Diskette. Verfahren Sie wie folgt:

Schritt 1: Kopieren Sie die Kyan-Pascal-2.0-Diskette „Disk 2, Side 2", die mit der ProDOS-„Users.Disk" identisch ist, mit einem beliebigen 35-Spur-Kopierprogramm, z.B. COPYA u.ä.

Schritt 2: Booten Sie eine DOS-3.3-Arbeitsdiskette von einem beliebigen Slot und geben Sie dann das auf S. 59 abgedruckte Maschinenprogramm ein und speichern Sie es sicherheitshalber mit PRODOS.PATCH,1.1.1, A$900, L$100, D1 auf Ihrer DOS-3.3-Arbeitsdiskette. Das Programm ist also jetzt gesichert und zusätzlich noch im Speicher.

Schritt 3: Legen Sie die *Kopie* von „Disk 2, Side 2" (= Users.Disk) in Drive 1, gehen Sie mit CALL -151 in den Monitor und tippen Sie
900G.
Damit werden die ersten 5 Spuren von „Disk 2, Side 2" wie folgt in den Speicher eingelesen:
$1000-$13FF: 2 Boot-Blocks
$1400-$1BFF: 4 Volume-Directory-Blocks
$1C00-$1DFF: 1 Volume-Bit-Map-Blocks
$1E00-$1FFF: 1 PRODOS-Index-Block
$2000-$59FF: 29 PRODOS-Programm-Blocks

Schritt 4: Führen Sie folgende zwei Patches durch:
2415: D0 17 (= BNE $242E; S. 319; statt 2415: E0 EA = CPX #$EA: Machine-ID-Patch $FBB3).
269E: 38 EA (= SEC NOP; S. 324; statt 269E: D0 03 = BNE $26A3: Apple-Logo-Patch $FB09).
Wenn das letzte Byte Ihres kompatiblen DOS-Controllers ($C6FF) *nicht $00* enthält, dann führen Sie noch die nächsten zwei Patches durch, nach denen allerdings z.B. die Megaboard-Festplatte nicht mehr erkannt werden würde:
254B: A9 00 (= LDA #$00; S. 321; statt 254B: B1 10 = LDA ($10),Y: Disk-Slot-Suche-Patch)
101D: A9 00 (= LDA #$00; S. 183; statt 101D: B1 48 = LDA ($48),Y: Boot-Slot-Patch).

Schritt 5: Tippen Sie nun
904G,
womit die ersten 5 Spuren wieder auf die Diskette zurückgeschrieben werden. Jetzt müßte die „Users.Disk" korrekt booten. Wenn nicht, dann haben Sie keinen Kompatiblen, sondern einen „Unkompatiblen" und müssen anhand der „ProDOS-Analyse" umfassendere Patches vornehmen.

2. Anfänger-Arbeitsdiskette

Anfänger haben wahrscheinlich zunächst Schwierigkeiten mit den ProDOS-Subdirectories sowie mit den zahlreichen Kix-Utilities und sollten sich ein Volume „/K" mit zunächst nur folgenden Dateien zusammenstellen (im Klammern Blockanzahl):
/K
PRODOS (30) – Version 1.1.1
KIX.SYSTEM (6) – Haupt-Menü
LS (16) – Directory-Befehl
ED (21) – Editor (gerätespezifisch!)
PC (55) – Pascal-Compiler
AS (22) – Assembler
LIB (25) – Library
STDLIB.S (21) – Standard-Lib-Source
Dieses Volume mit PR#6 booten, dann nach Prompt „%" das halbfett Gedruckte eingeben; Leertasten beachten!
% **LS -N -L**
für Anzeige des Directory.
% **ED TEST.P**
für Editor starten.
Jetzt Testprogramm eingeben:
Program Test;
Begin
Writeln ('Test')
End.
Jetzt die Tasten
Esc X
tippen zum automatischen Abspeichern von „TEST.P". Nun compilieren:
% **PC TEST.P**
erzeugt fertigen Objektcode „TEST".
Zur Kontrolle Directory ansehen mit
% **LS -N -L**
% **TEST**
startet Objektcode „TEST" und zeigt „Test" am Bildschirm an. Ende!
Für Fortgeschrittene:
% **PC TEST.P -S**
erzeugt Assembler-Source „P.OUT".
„P.OUT" = Pascal-Compiler-Output.
% **AS P.OUT**
erzeugt Objektcode „A.OUT".
„A.OUT" = Assembler-Output.
Alternativ:
% **AS P.OUT -O TEST**
erzeugt Objektcode „TEST"
= identisch mit „A.OUT".
Wir merken uns:
TEST.P = Pascal-Quelltext
(Suffix „.P" stets selbst tippen!)
TEST = Pascal-Objektcode
(Suffix „.P" entfällt dann automatisch)
P.OUT = Assembler-Quelltext
A.OUT = Assembler-Objektcode

Ulrich Stiehl

DOS-3.3-Backup-Programm

für die Megaboard-Festplatte

von Ulrich Stiehl

Das folgende Kopierprogramm eignet sich für Festplatten, die an den Megaboard-Controller der Firma Frank & Britting angeschlossen worden sind, insbesondere für die anstelle des Netzteils eingebaute Mega-Core oder die externe MDB (= „mobile Datenbox").

Wenn man eine Festplatte erwirbt, so muß man sich auch Gedanken über eine geeignete Datensicherung machen. Unser Progamm kopiert ein 140K-DOS-3.3-Volume von der MDB auf ein 35-Spur-Laufwerk und umgekehrt in nur 22,5s. Das vorgelagerte Applesoft-Programm MDB.KOPY dient als Muster, das man sich auf seine eigenen Verhältnisse abändern sollte.

Backup-Beispiel

Nehmen wir an, daß sich auf einer MDB10 neben ProDOS-, UCSD- und CP/M-Volumes und dem sog. DOS-System-Volume insgesamt 9 weitere DOS-3.3-Volumes mit je 2 Drive zu je 140K befinden (= Volumes 2-10). Man würde dann 18 35-Spur-Disketten benötigen, die aus Gründen der Geschwindigkeit bei der späteren Datensicherung bereits vorformatiert sein sollten (mit INIT HELLO, o.ä.). Nehmen wir ferner an, daß sich das 35-Spur-Laufwerk in Slot 6, Drive 1 befindet und die 9 MDB-Volumes die Nummern 2-10 haben. Dann sind folgende 18 Kopierdurchläufe erforderlich:

S7,D1,V2 → S6,D1 1. Diskette
S7,D2,V2 → S6,D1 2. Diskette
...
S7,D1,V10 → S6,D1 17. Diskette
S7,D2,V10 → S6,D2 18. Diskette

Man kann dann das Applesoft-Programm MDB.KOPY durch das Mini-Programm MDB.KOPY.BEISPIEL ersetzen, das die Slot-, Drive- und Volume-Parameter automatisch zuweist. Dann benötigt man, weil die Eingabe über die Tastatur entfällt, einschließlich des Diskettenwechselns etwa 30s pro Diskettenkopie, mithin für 18 Disketten insgesamt 9 Minuten. Dies ist eine durchaus erträgliche Zeit für immerhin 2 * 140 * 9 = 2.520K (etwa 2.5M). Damit alles reibungslos abläuft, sollte man für eine deutliche Beschriftung der Backup-Disketten Sorge tragen. Ein weiterer Praxis-Tip: Legen Sie auf jeder Backup-Diskette eine zusätzliche Datei mit demjenigen Namen an, mit dem Sie das Etikett der Diskette beschriftet haben, z.B.

SAVE KUNDENDATEI-V2-D1

Das Programm MDB.KOPY zeigt nämlich aus Sicherheitsgründen *vor* jedem Kopierdurchlauf das Inhaltsverzeichnis der Backup-Kopie an.

MDB.KOPY.BEISPIEL

18 Backup-Disketten für Slot 6, Drive 1
Volumes 2-10, Drive 1-2, Slot 7 (Harddisk)

```
10 PRINT  CHR$ (4)"BLOAD MDB.KOPY.OBJ":
20 PRINT  CHR$ (21): TEXT : HOME : INVERSE :
   PRINT "MDB.KOPIERER DOS 3.3":
   PRINT "VON U.STIEHL  1.4.86": NORMAL :S = 3300
30 FOR V = 2 TO 10: FOR D = 1 TO 2: PRINT :
   PRINT "BACKUP FUER V";V;",D";D;" EINLEGEN ";:
   POKE 49168,0: GET X$
40 POKE S + 3,7 * 16: POKE S + 4,D: POKE S + 5,V:
   POKE S + 6,6 * 16: POKE S + 7,1: POKE S + 8,0
50 HOME : PRINT : PRINT  CHR$ (4)"CATALOG,S6,D1":
   PRINT "DUPLIKATDISK, RICHTIG J/N ";
60 GET X$: ON X$ = "N" GOTO 50: IF X$ <  > "J" GOTO 60
70 PRINT : CALL S: NEXT D,V
```

MDB.KOPY.OPJ

```
BSAVE MDB.KOPY.OBJ, A3300, L498
```

```
                        1             ORG  3300       ;$0CE4
                        2   *
                        3   * MDB.KOPY.OBJ US/4/86
                        4   * ==========
                        5   *
                        6   * MDB-Mobile-Datenbox-Kopierer
                        7   * für 35-Spur-DOS-Volumes
                        8   * 64K-Apple mit 48K-DOS-3.3
                        9   * Kopierzeiten:
                        10  * 41,0s Disk to Disk
                        11  * 22,5s Disk to Hard
                        12  *  6,2s Hard to Hard
                        13  *
0CE4: 4C EE 0C          14            JMP  START
                        15  *
0CE7: 70                16  ORIGSLOT HEX  70         ;3300+3
0CE8: 01                17  ORIGDRIV HEX  01         ;3300+4
0CE9: 03                18  ORIGVOL  HEX  03         ;3300+5
0CEA: 70                19  DUPLSLOT HEX  70         ;3300+6
0CEB: 02                20  DUPLDRIV HEX  02         ;3300+7
0CEC: 03                21  DUPLVOL  HEX  03         ;3300+8
0CED: 00                22  INITFLAG HEX  00         ;3300+9
                        23  *
                        24  IOBL     EQU  $CE
                        25  IOBH     EQU  $CF
                        26  DOSWARM  EQU  $3D0
                        27  RWTS     EQU  $3D9
                        28  GETIOB   EQU  $3E3
                        29  HOME     EQU  $FC58
                        30  HEXOUT   EQU  $FDDA
                        31  PRINT    EQU  $FDED
                        32  PUFFER0  EQU  $1000
                        33  PUFFER1  EQU  $D000       ;Bank1
                        34  PUFFER2  EQU  $D000       ;Bank2
                        35  *
                        36  * DOS-Input-Output-Block nehmen
                        37  *
0CEE: 20 58 FC          38  START    JSR  HOME
0CF1: A9 CB             39           LDA  #"K"
0CF3: 20 ED FD          40           JSR  PRINT
0CF6: 20 E3 03          41           JSR  GETIOB
0CF9: 84 CE             42           STY  IOBL
0CFB: 85 CF             43           STA  IOBH
                        44  *
                        45  * Read Tracks 34-27, 26, 25-23
                        46  *
0CFD: 20 81 0E          47           JSR  ORIGINAL
0D00: 20 8E 0D          48           JSR  T34_27
0D03: 20 32 0E          49           JSR  BLOCK0
0D06: 20 9B 0D          50           JSR  T26_26
0D09: 20 32 0E          51           JSR  BLOCK0
0D0C: 20 A8 0D          52           JSR  T25_23
0D0F: 20 32 0E          53           JSR  BLOCK0
                        54  *
                        55  * 0 = kein Init, 1 = Init
                        56  * Bei Festplatte kein Init!!!
                        57  *
0D12: AD ED 0C          58  INITJA   LDA  INITFLAG
0D15: F0 12             59           BEQ  INITNEIN
0D17: 20 A6 0E          60           JSR  DUPLIKAT
0D1A: A0 0C             61           LDY  #$0C        ;Command
0D1C: A9 04             62           LDA  #4          ;Init
0D1E: 91 CE             63           STA  (IOBL),Y
0D20: A0 03             64           LDY  #$03        ;Volume
0D22: A9 00             65           LDA  #0          ;0=254
0D24: 91 CE             66           STA  (IOBL),Y
0D26: 20 74 0E          67           JSR  RWTS1
0D29: EA                68  INITNEIN NOP
                        69  *
                        70  * Write Tracks 34-27, 26, 25-23
                        71  *
0D2A: 20 A6 0E          72           JSR  DUPLIKAT
0D2D: 20 8E 0D          73           JSR  T34_27
0D30: 20 32 0E          74           JSR  BLOCK0
0D33: 20 9B 0D          75           JSR  T26_26
0D36: 20 32 0E          76           JSR  BLOCK0
0D39: 20 A8 0D          77           JSR  T25_23
0D3C: 20 32 0E          78           JSR  BLOCK0
                        79  *
                        80  * Read Tracks 22-15, 14, 13-11
                        81  *
0D3F: 20 81 0E          82           JSR  ORIGINAL
0D42: 20 B5 0D          83           JSR  T22_15
0D45: 20 32 0E          84           JSR  BLOCK0
0D48: 20 C2 0D          85           JSR  T14_14
0D4B: 20 32 0E          86           JSR  BLOCK0
0D4E: 20 CF 0D          87           JSR  T13_11
0D51: 20 32 0E          88           JSR  BLOCK0
                        89  *
                        90  * Write Tracks 22-15, 14, 13-11
                        91  *
0D54: 20 A6 0E          92           JSR  DUPLIKAT
0D57: 20 B5 0D          93           JSR  T22_15
0D5A: 20 32 0E          94           JSR  BLOCK0
0D5D: 20 C2 0D          95           JSR  T14_14
0D60: 20 32 0E          96           JSR  BLOCK0
0D63: 20 CF 0D          97           JSR  T13_11
0D66: 20 32 0E          98           JSR  BLOCK0
                        99  *
                        100 * Read Tracks 10-3, 2-0
                        101 *
0D69: 20 81 0E          102          JSR  ORIGINAL
0D6C: 20 DC 0D          103          JSR  T10_03
0D6F: 20 32 0E          104          JSR  BLOCK0
0D72: 20 E9 0D          105          JSR  T02_00
0D75: 20 32 0E          106          JSR  BLOCK0
                        107 *
                        108 * Write Tracks 10-3, 2-0
                        109 *
0D78: 20 A6 0E          110          JSR  DUPLIKAT
0D7B: 20 DC 0D          111          JSR  T10_03
0D7E: 20 32 0E          112          JSR  BLOCK0
0D81: 20 E9 0D          113          JSR  T02_00
0D84: 20 32 0E          114          JSR  BLOCK0
                        115 *
0D87: AD 81 C0          116          LDA  $C081
0D8A: AD 81 C0          117          LDA  $C081
0D8D: 60                118          RTS              ;Ende
                        119 *
0D8E: A9 22             120 T34_27   LDA  #34         ;8
0D90: 20 F6 0D          121          JSR  TRACKER
0D93: A9 08             122          LDA  #8
0D95: 8D 5E 0E          123          STA  TCOUNT
0D98: 4C 05 0E          124          JMP  PUF0
                        125 *
0D9B: A9 1A             126 T26_26   LDA  #26         ;1
0D9D: 20 F6 0D          127          JSR  TRACKER
0DA0: A9 01             128          LDA  #1
0DA2: 8D 5E 0E          129          STA  TCOUNT
0DA5: 4C 10 0E          130          JMP  PUF1
                        131 *
0DA8: A9 19             132 T25_23   LDA  #25         ;3
0DAA: 20 F6 0D          133          JSR  TRACKER
0DAD: A9 03             134          LDA  #3
0DAF: 8D 5E 0E          135          STA  TCOUNT
0DB2: 4C 21 0E          136          JMP  PUF2
                        137 *
0DB5: A9 16             138 T22_15   LDA  #22         ;8
0DB7: 20 F6 0D          139          JSR  TRACKER
0DBA: A9 08             140          LDA  #8
0DBC: 8D 5E 0E          141          STA  TCOUNT
0DBF: 4C 05 0E          142          JMP  PUF0
                        143 *
0DC2: A9 0E             144 T14_14   LDA  #14         ;1
0DC4: 20 F6 0D          145          JSR  TRACKER
0DC7: A9 01             146          LDA  #1
0DC9: 8D 5E 0E          147          STA  TCOUNT
0DCC: 4C 10 0E          148          JMP  PUF1
                        149 *
0DCF: A9 0D             150 T13_11   LDA  #13         ;3
0DD1: 20 F6 0D          151          JSR  TRACKER
```

```
ØDD4: A9 Ø3     152           LDA  #3
ØDD6: 8D 5E ØE  153           STA  TCOUNT
ØDD9: 4C 21 ØE  154           JMP  PUF2
                155  *
ØDDC: A9 ØA     156  T1Ø_Ø3   LDA  #1Ø        ;8
ØDDE: 2Ø F6 ØD  157           JSR  TRACKER
ØDE1: A9 Ø8     158           LDA  #8
ØDE3: 8D 5E ØE  159           STA  TCOUNT
ØDE6: 4C Ø5 ØE  16Ø           JMP  PUFØ
                161  *
ØDE9: A9 Ø2     162  TØ2_ØØ   LDA  #2         ;3
ØDEB: 2Ø F6 ØD  163           JSR  TRACKER
ØDEE: A9 Ø3     164           LDA  #3
ØDFØ: 8D 5E ØE  165           STA  TCOUNT
ØDF3: 4C 21 ØE  166           JMP  PUF2
                167  *
ØDF6: AØ Ø4     168  TRACKER  LDY  #$Ø4       ;Track
ØDF8: 91 CE     169           STA  (IOBL),Y
ØDFA: 6Ø        17Ø           RTS
                171  *
ØDFB: AØ Ø8     172  BUFLOW1  LDY  #$Ø8       ;low
ØDFD: 91 CE     173           STA  (IOBL),Y
ØDFF: 6Ø        174           RTS
                175  *
ØEØØ: AØ Ø9     176  BUFHIGH1 LDY  #$Ø9       ;high
ØEØ2: 91 CE     177           STA  (IOBL),Y
ØEØ4: 6Ø        178           RTS
                18Ø  *
                181  * PUFØ: 1ØØØ-8FFF-RAM:   8 Spuren
                182  *
ØEØ5: A9 ØØ     183  PUFØ     LDA  #<PUFFERØ
ØEØ7: 2Ø FB ØD  184           JSR  BUFLOW1
ØEØA: A9 1Ø     185           LDA  #>PUFFERØ
ØEØC: 2Ø ØØ ØE  186           JSR  BUFHIGH1
ØEØF: 6Ø        187           RTS
                188  *
                189  * PUF1: DØØØ-DFFF-Bank1: 1 Spur
                19Ø  *
ØE1Ø: A9 ØØ     191  PUF1     LDA  #<PUFFER1
ØE12: 2Ø FB ØD  192           JSR  BUFLOW1
ØE15: A9 DØ     193           LDA  #>PUFFER1
ØE17: 2Ø ØØ ØE  194           JSR  BUFHIGH1
ØE1A: AD 8B CØ  195           LDA  $CØ8B
ØE1D: AD 8B CØ  196           LDA  $CØ8B       ;RD/WR
ØE2Ø: 6Ø        197           RTS
                198  *
                199  * PUF2: DØØØ-FFFF-Bank2: 3 Spuren
                2ØØ  *
ØE21: A9 ØØ     2Ø1  PUF2     LDA  #<PUFFER2
ØE23: 2Ø FB ØD  2Ø2           JSR  BUFLOW1
ØE26: A9 DØ     2Ø3           LDA  #>PUFFER2
ØE28: 2Ø ØØ ØE  2Ø4           JSR  BUFHIGH1
ØE2B: AD 83 CØ  2Ø5           LDA  $CØ83
ØE2E: AD 83 CØ  2Ø6           LDA  $CØ83       ;RD/WR
ØE31: 6Ø        2Ø7           RTS
                2Ø8  *
                2Ø9  * Block-Transfer
                21Ø  *
ØE32: AØ Ø5     211  BLOCKØ   LDY  #$Ø5       ;Sektor
ØE34: A9 ØF     212           LDA  #15
ØE36: 91 CE     213           STA  (IOBL),Y
ØE38: 2Ø 74 ØE  214  BLOCK1  JSR  RWTS1
ØE3B: AØ Ø9     215           LDY  #$Ø9       ;Puffer
ØE3D: B1 CE     216           LDA  (IOBL),Y
ØE3F: 18        217           CLC
ØE4Ø: 69 Ø1     218           ADC  #1         ;INC
ØE42: 91 CE     219           STA  (IOBL),Y
ØE44: AØ Ø5     22Ø           LDY  #$Ø5       ;Sektor
ØE46: B1 CE     221           LDA  (IOBL),Y
ØE48: 38        222           SEC
ØE49: E9 Ø1     223           SBC  #1         ;DEC
ØE4B: 91 CE     224           STA  (IOBL),Y
ØE4D: 1Ø E9     225           BPL  BLOCK1
ØE4F: AØ Ø4     226           LDY  #$Ø4       ;Track
ØE51: B1 CE     227           LDA  (IOBL),Y
ØE53: 38        228           SEC
ØE54: E9 Ø1     229           SBC  #1         ;DEC
ØE56: 91 CE     23Ø           STA  (IOBL),Y
ØE58: CE 5E ØE  231           DEC  TCOUNT
ØE5B: DØ D5     232           BNE  BLOCKØ
ØE5D: 6Ø        233           RTS
ØE5E: ØØ        234  TCOUNT   HEX  ØØ
                235  *
                236  * Lesefehler
                237  *
ØE5F: 68        238  ERROR    PLA
ØE6Ø: 68        239           PLA
ØE61: AD 81 CØ  24Ø           LDA  $CØ81
ØE64: AD 81 CØ  241           LDA  $CØ81      :RDROM
ØE67: 2Ø 58 FC  242           JSR  HOME
```

```
ØE6A: AØ ØD     243           LDY  #$ØD       ;Error
ØE6C: B1 CE     244           LDA  (IOBL),Y
ØE6E: 2Ø DA FD  245           JSR  HEXOUT
ØE71: 4C DØ Ø3  246           JMP  DOSWARM
                247  *
ØE74: 2Ø E3 Ø3  248  RWTS1    JSR  GETIOB
ØE77: 2Ø D9 Ø3  249           JSR  RWTS
ØE7A: BØ E3     25Ø           BCS  ERROR
ØE7C: A9 ØØ     251           LDA  #Ø
ØE7E: 85 48     252           STA  $48
ØE8Ø: 6Ø        253           RTS
                254  *
                255  * Original-Volume: 1 Read
                256  *
ØE81: AØ Ø1     257  ORIGINAL LDY  #$Ø1       ;Slot
ØE83: AD E7 ØC  258           LDA  ORIGSLOT
ØE86: 91 CE     259           STA  (IOBL),Y
ØE88: C8        26Ø           INY             ;Drive
ØE89: AD E8 ØC  261           LDA  ORIGDRIV
ØE8C: 91 CE     262           STA  (IOBL),Y
ØE8E: C8        263           INY             ;Volume
ØE8F: AD E9 ØC  264           LDA  ORIGVOL
ØE92: 91 CE     265           STA  (IOBL),Y
ØE94: AØ ØC     266           LDY  #$ØC       ;Befehl
ØE96: A9 Ø1     267           LDA  #1         ;Read
ØE98: 91 CE     268           STA  (IOBL),Y
ØE9A: 2Ø CB ØE  269           JSR  PATCH1
ØE9D: DØ Ø6     27Ø           BNE  ORIGEXIT
ØE9F: AD E9 ØC  271           LDA  ORIGVOL
ØEA2: 8D 66 AA  272           STA  $AA66      ;Volume
ØEA5: 6Ø        273  ORIGEXIT RTS
                274  *
                275  * Duplikat-Volume: 2 Write
                276  *
ØEA6: AØ Ø1     277  DUPLIKAT LDY  #$Ø1       ;Slot
ØEA8: AD EA ØC  278           LDA  DUPLSLOT
ØEAB: 91 CE     279           STA  (IOBL),Y
ØEAD: C8        28Ø           INY             ;Drive
ØEAE: AD EB ØC  281           LDA  DUPLDRIV
ØEB1: 91 CE     282           STA  (IOBL),Y
ØEB3: C8        283           INY             ;Volume
ØEB4: AD EC ØC  284           LDA  DUPLVOL
ØEB7: 91 CE     285           STA  (IOBL),Y
ØEB9: AØ ØC     286           LDY  #$ØC       ;Befehl
ØEBB: A9 Ø2     287           LDA  #2         ;Write
ØEBD: 91 CE     288           STA  (IOBL),Y
ØEBF: 2Ø CB ØE  289           JSR  PATCH1
ØEC2: DØ Ø6     29Ø           BNE  DUPLEXIT
ØEC4: AD EC ØC  291           LDA  DUPLVOL
ØEC7: 8D 66 AA  292           STA  $AA66      ;Volume
ØECA: 6Ø        293  DUPLEXIT RTS
                294  *
                295  * Patch für MDB-Controller
                296  *
ØECB: A9 7Ø     297  PATCH1   LDA  #$7Ø       ;Slot 7
ØECD: CD E7 ØC  298           CMP  ORIGSLOT
ØED0: DØ Ø3     299           BNE  PATCH2
ØED2: CD EA ØC  3ØØ           CMP  DUPLSLOT
ØED5: 6Ø        3Ø1  PATCH2   RTS
```

Hinweis:
Zu den IOB-Zeigern (Input-Output-Block) siehe „Apple DOS 3.3“, S. 107.

MDB.KOPY

```
10 IF PEEK (116) < > 150 OR PEEK (104) < > 8 THEN
   PRINT "SPEICHERVERTEILUNG STIMMT NICHT!": END
15 ONERR GOTO 70
20 PRINT  CHR$ (4)"BLOAD MDB.KOPY.OBJ"
25 PRINT CHR$ (21): TEXT : HOME : INVERSE :
   PRINT "MDB.KOPIERER DOS 3.3": PRINT "VON U. STIEHL 1.4.86":
   NORMAL :S = 3300
30 VTAB 3: PRINT: PRINT "ORIGINAL-SLOT: ";: GOSUB 65:
   POKE S + 3,W * 16: PRINT "ORIGINAL-DRIVE: ";:
   GOSUB 65: POKE S + 4,W: PRINT "ORIGINAL-VOLUME:";:
   GOSUB 65: POKE S + 5,W
35 PRINT : PRINT "DUPLIKAT-SLOT: ";: GOSUB 65:
   POKE S + 6,W * 16: PRINT "DUPLIKAT-DRIVE: ";:
   GOSUB 65: POKE S + 7,W: PRINT "DUPLIKAT-VOLUME: ";:
   GOSUB 65: POKE S + 8,W
40 PRINT : PRINT "RICHTIG J/N ";
45 GET X$: ON X$ = "N" GOTO 25: IF X$ < > "J" GOTO 45
50 HOME : PRINT : PRINT CHR$ (4)"CATALOG,S";
   PEEK (S + 6) / 16;",D"; PEEK (S + 7);",V"; PEEK (S + 8):
   PRINT „DUPLIKATDISK"; CHR$ (7): PRINT "RICHTIG J/N ";
55 GET X$: ON X$ = "N" GOTO 25: IF X$ < > "J" GOTO 55
60 PRINT : CALL S: RUN 25
65 INPUT "";X$:W = VAL (X$): RETURN
70 POKE 216,0: HOME : PRINT "FEHLER": END
```

10 JAHRE apple computer

Ein Rückblick

von Dr. Jürgen B. Kehrel

Im April 1976 gründeten Steve Jobs und Steve Wozniak die „Garagenfirma" Apple Computer Inc., nachdem sie Ende 1975 den 6502-Computer „Apple I" in einigen Exemplaren gebaut und verkauft hatten. Dies war ein einfacher Einplatinen-Rechner gewesen, der sein Schwarz-Weiß-Bild auf einem Fernseher ausgab und einen Slot zum Anschluß eines Kassetteninterfaces besaß. Jobs hatte vorher bei Atari gearbeitet und dort u.a. das Spiel „Breakout" entworfen. Wozniak hatte sein College-Studium unterbrochen, um bei Hewlett-Packard in der Taschenrechner-Entwicklung zu jobben. In ihm reifte auch die Idee zum Bau eines kompakten und preiswerten Homecomputers. Da HP an dem Projekt nicht interessiert war, begann er selbst mit der Produktion in seiner Garage und bot die fertigen Einzelexemplare in seinem Computerclub an. Zu den beiden „Urvätern" von Apple kamen bald noch drei weitere hinzu: Rod Holt war Entwicklungsingenieur bei Atari gewesen und baute für Apple u.a. das Schaltnetzteil. Mike Markkula war Marketingleiter beim Halbleiter-Hersteller Intel gewesen und übernahm bei Apple die kaufmännische Seite. Mike Scott kam von National Semiconductor und wurde einer der Präsidenten von Apple.

Im Laufe des Jahres 1976 wurde der Apple II entwickelt, an dem auch noch Allen Baum (Monitor-ROM zusammen mit Wozniak, Slotkonzept) und Chris Espinoza (Manuals zusammen mit Mike Scott) beteiligt waren. Auf der West Coast Computer Faire am 5. April 1977 wurde der Apple II öffentlich vorgestellt, ab Juni 1977 wurde er ausgeliefert. Es war der erste Computer, der ein BASIC (Integer BASIC) im ROM hatte und auf nur einer Platine das bot, wozu andere Firmen einen ganzen Satz von Steckkarten benötigten. Der Hauptspeicher war 4K groß, konnte aber auf 16K und später auf 48K ausgebaut werden, als die neuen 16K-Speicherchips zur Verfügung standen (zum Vergleich: heute werden 256K-Chips in Serie gefertigt und 1M-Chips sind als Einzelexemplare erhältlich). Als Massenspeicher diente ein normaler Kassettenrecorder.

Thomas Munnecke führte Anfang 1977 mit Steve Jobs ein Interview, das erst 1981 in der Zeitschrift ELCOMP veröffentlicht wurde und aus dem wir auszugsweise mit freundlicher Genehmigung des Hofacker-Verlages zitieren:
Jobs: *Ich startete zusammen mit meinem Partner Steve Wozniak. Wir wollten eigent-*

lich einen Computer für uns selber bauen, konnten uns diesen jedoch finanziell nicht leisten. Deshalb hat Woz einen entwickelt. Dieses Ding war großartig, und auf einmal wollten alle unsere Freunde und die Leute, die mit den Ingenieuren arbeiteten, ein solches Gerät für sich selber haben. Dieses Gerät ... haben wir noch von Hand verdrahtet und es hat ca. 60 Stunden gedauert, bis wir es gebaut und die Fehler beseitigt hatten. ... Die Freunde haben uns dann gedrängt, eine Platine anzufertigen. Wir haben gesagt: „Ja, wir machen welche" und ich habe mein Auto verkauft, mein Partner hat seinen HP 65-Rechner verkauft. Wir haben damit eine Platine entwickeln und die Platinen anfertigen lassen. ... Das ganze Konzept hinter dem APPLE II ist es, zuerst mit einem Home-Type Personal Computer, welcher nicht ein Hobbycomputer ist, auf den Markt zu kommen. Er sollte komplett aufgebaut sein, er sollte Farbgrafik haben und soll sehr einfach für den Neuling zu verstehen und anzuwenden sein. Wir haben gesehen, daß der Markt sich in diese Richtung bewegt. ... Weg vom Hobbycomputer-Markt, in dem sich der Anwender seine Computer noch selbst zusammenbauen muß. Wir haben festgestellt, daß die meisten Hobbyisten sich ihren Computer gar nicht selbst zusammenbauen wollen. Sie wollen ihn, wenn der Preis stimmt, lieber kaufen. Dies war unser Ziel der Entwicklung.
... Wir fanden heraus, daß wir noch einige Dinge einbauen sollten, z.B. vier Analog/Digital-Wandler zum Anschluß von vier „Paddles". In BASIC wurde dann ein Befehl „Lese Paddle" eingefügt. Der Befehl ermöglichte es dann, die Werte vom Wandler direkt in den Rechner zu übernehmen. In gleicher Weise implementierten wir die Farbbefehle in BASIC. Im Gegensatz zu herkömmlichen Techniken, wo man zur Farbsteuerung zur Maschinenebene hinabsteigen muß, kann man hier einfach von BASIC aus die Farben steuern. ... Diese Möglichkeit hat es bis zu diesem Zeitpunkt noch in keiner Computergeneration gegeben.
... Alles ist in Software implementiert, es befindet sich alles im ROM und es ist sehr schnell. ... Wir haben bis heute noch kein schnelleres BASIC auf dem Markt gesehen. Einige Benchmarktests zeigten, daß wir mindestens zweimal schneller waren als der nächstfolgende Mitbewerber. ... Wir leisten Ingenieurarbeit und keine Hobbyarbeit. ... Sie sehen, wir haben für unseren Computer ein Plastikgehäuse, wir haben das erste BASIC im ROM, wir haben das erste System, welches die 16K RAMs verwendet. Auf jedem Gebiet im System sind wir um mindestens 9 Monate dem Rest der Mitbewerber voraus.
Mitte 1978 erscheint der Apple II Plus, der das um Fließkommaroutinen erweiterte Applesoft-BASIC im ROM enthält (von Microsoft und Randy Wiggington). Steve Wozniak und Randy Wiggington haben außerdem DOS 3.1 und den Controller entwickelt, um auf einfache Weise Shugart-Laufwerke als Diskettenstation an den Apple anzuschließen. Das Apple-Analog-Board besitzt nur 4 IC's, während im Shugart-Original noch 23 benötigt wurden. Eine S-100-Controller-Karte hat 30 IC's, der Apple-Controller dagegen nur 8.
Steve Jobs war nicht sehr an der Computerentwicklung interessiert. Ihn reizte von Anfang an mehr der Verkauf des Apple, und er schaffte es, genügend Kredite aufzutun und wichtige Leute in die Firma zu holen. Bis Dezember 1977 wurden alle Kredite wieder zurückgezahlt. Im September 1978 arbeiteten bereits 74 Leute bei Apple Computer Inc., und die Geräte waren zu diesem Zeitpunkt bei über 340 Vertragshändlern zu finden. 25% der Produktion wird exportiert, von denen einige wohl auch in Deutschland landeten.
Seit 1979 war BASIS-Computer in Münster der deutsche Apple-Generalimporteur, der den Apple II Plus mit 16K RAM für DM 3200,- zuzüglich Mehrwertsteuer anbot. Im September-Heft urteilte die Zeitschrift CHIP:

Gut
* Klein, leistungsfähig
* Umfangreiche Systemsoftware/Anwendersoftware
* Hochauflösende Farbgrafik
* Viele Erweiterungen auf dem Markt

Schlecht
* Relativ hoher Preis

Besonders beim letzten Punkt hat sich bis heute nicht viel geändert. ITT bietet mit dem ITT 2020 und Siemens mit dem PC 1000 jeweils eine „europäisierte" Version des Apple an, die sich aber beide nicht durchsetzen konnten und bald wieder verschwanden.
Im April 1979 gründeten ein paar der ersten deutschen Apple-Enthusiasten die Apple User Group Europe e.V. (AUGE), die durch ihre Zeitung (User Magazin) und durch die Treffen der Regionalgruppen ganz entscheidend zum Erfolg des Apple in Deutschland mit beitrugen. Da es praktisch keine deutschsprachigen Veröffentlichungen über den Apple gab und viele Fragen offen blieben, wurde die AUGE zu einem Kristallisationspunkt, an dem sich die Kenntnisse vieler engagierter Computernutzer trafen. Die bis heute auf mehrere Megabyte gewachsene Clubsoftware deckt ganz legal das Verlangen ab, dem Apple auch „Futter" zukommen zu lassen.
Peeker-Leser Harry Fischer aus Frankfurt erwarb seinen Apple (A2S1-29311) im September 1979: *Fa. Basis baute damals ein neues Händlernetz auf. Anruf dort ergab, nächster Apple-Händler sitzt in Karlsruhe. Also am nächsten Sonnabend auf nach Karlsruhe zum dortigen Apple-Händler. Alle Apple ausverkauft. Anruf bei Filiale in Heilbronn. Auch dort kein Apple auf Lager. Der Techniker in Karlsruhe wußte einen Ausweg. Ein Apple II sei defekt zurückgekommen, er besitze schon die ROMs des Apple II+ und könne wohl in kurzer Zeit repariert werden (defekte Tastatur). Eine Taste wurde ausgewechselt, eine nachgelötet und dann konnte ich (zum Neupreis von DM 4000,-!) einen ge-*

Apple-Anlage von Herrn Eichhorn

brauchten und reparierten 48K-Apple II(+) heimfahren. Dieser Apple tut noch heute seinen Dienst, allerdings mit Unterstützung eines GEPARD mit 10 MHz-68000. Ewald Meyer ersteht bereits im Juni 1979 das Modell A2S1-14914. Die Schweizer Peeker-Leser scheinen sich noch etwas schneller für einen Apple entschieden zu haben. Edmar Eichhorn in Uster wartet gar mit der Serien-Nummer A2S1-2037 und einer Floppy aus dem Jahre 1978 auf: *Zu den technischen Details möchte ich erwähnen, daß das Gerät noch mit einem 115V Netzteil ausgerüstet war. Nach mehrmaligen Schwierigkeiten mit dem Netzteil und Trafo entschloß ich mich, ein neues einzubauen. Außer dieser Änderung ist mein Apple noch in der Originalausführung, d.h. aufgerüstet auf 48K. ... Der Rechner ist heute täglich mehrere Stunden im Einsatz. Technische Berechnungen, wie Typenauswahl, Schall- und Leistungsberechnungen und Ausgabe der Daten über einen Plotter von Mirwald Electronic sind das hauptsächliche Einsatzgebiet.*

Ebenfalls zu den „Applern" der ersten Stunde gehörte Herr Georg Lachenmeier in Zürich, dessen Apple II am 7.2.1979 geliefert wurde: 16K RAM, Integer-BASIC, alter Monitor für Fr. 3650,-. Die 16 zusätzlichen IC's zum Aufstocken auf 48K schlugen noch einmal mit Fr. 550,- zu Buche, während die Floppydisk mit Controller (13 Sektoren – 116K Kapazität) mit Fr. 1650,- berechnet wurde. Er schreibt: *Zuerst aber folgte ein Entscheid für ein Computermodell. Und ich entschied mich – nein, nicht für Apple, sondern für einen Commodore PET 2001. Der Hauptgrund war, daß Commodore kürzere Lieferzeiten versprach, und in meiner Naivität glaubte ich den Amerikanern. Sie kennen ja die Lieferversprechen. Zum PET (8K RAM, Fr. 3000,-), wovon wir zwei Stück kauften, kauften wir den billigsten Drucker für Fr. 3500,-.... Es war ein Centronics 779 Nadeldrucker, nur Großbuchstaben, ohne jede Intelligenz, ohne Traktorfeed. ... Nachdem Commodore seine Versprechungen bezüglich Diskdrives nicht hielt (Lieferfrist), und die Speichererweiterung astronomische Kosten verursacht hätte, entschied ich mich frustriert für den Apple II. Sofort wollte ich die zwei PET's verkaufen, was bei einem gelang, doch durch den Preisverfall blieb ich auf dem zweiten sitzen. ... Sofort versuchte ich, einen Apple II zu bekommen, und der Händler berücksichtigte mich als einen der ersten, doch warten mußten alle. Die Diskdrives waren rationiert, und jeder erhielt vorerst nur ein Stück.*

Im Jahre 1980 gab es nicht nur den Apple II Plus mit Disk II und DOS 3.2, sondern es waren ein Grafik-Tablett, der Thermodrukker Silentype und diverse Schnittstellen (parallel und seriell) dazugekommen. Apple Pascal wurde als „die zur Zeit beste verfügbare, strukturierte Programmiersprache" angeboten. Außerdem boten viele Fremdhersteller Erweiterungen an, von denen die Videx-Videoterm-80-Zeichenkarte wohl die größte Verbreitung fand, da sie den Apple zu professionellen Aufgaben befähigte. Die wichtigste Software wurde Visicalc, der Pionier der Tabellenkalkulation. Die Apple-Preisliste vom 1.1.1981 weist folgende Posten auf:

Apple II Plus 16K	DM 3396,65
Apple II Plus 32K	DM 3611,48
Apple II Plus 48K	DM 3827,31
Pascal Language System	DM 1381,99
Disk II mit Controller	DM 2013,66
Disk II als 2. Laufwerk	DM 1414,76
Silentype Thermodrucker	DM 1658,84
Centronics Interface	DM 636,19

In einem frühen Prospekt heißt es: *Apple hat sich einen Namen gemacht mit fortschrittlichem Design und Innovationen wie:*

* *Farbgrafiken*
* *hochauflösender Grafik*
* *Sprach- und Ton-Synthese*
* *Analog-Eingabe*

Apple versteht System-Unterstützung so: Dokumentation, Software, Zubehör, und die Leistungsfähigkeiit des Systems, um all dies zu bewältigen. ... Wir kümmern uns um unsere Kunden! Mehr als 90% von ihnen sagen, daß sie ihren Freunden den Apple II empfehlen würden.

Die technische Entwicklung des Apple II ist nicht stehengeblieben. Der IIe kam auf den Markt, der 128K Speicher und doppelthochauflösende Grafik besitzen kann, jedoch von der eingebauten ROM-Software keinerlei Unterstützung erhält. Die Manuals mit den nötigen Informationen sind heute nur separat und gegen gutes Geld zu erstehen. Der Apple IIc integrierte die wichtigsten Erweiterungen in einem handlichen Gehäuse, das wegen der fehlenden transportablen Stromversorgung und des Monitors aber nicht zu einem „Portable" reichte. Der angebotene Flachbildschirm bringt eher Augenärzten neue Kunden. Der IIc war zugleich auch der Rechner, der die neue Marktstrategie von Apple verkörperte: ein nettes geschlossenes System und dazu Handbücher mit Farbdruck. Beim Mac wurde das dann noch um eine Stufe potenziert geboten und brachte diesem Gerät den Ruf ein, der teuerste Spielcomputer der Welt zu sein. Laut „Byte" machte Apple 1985 noch 85% seines Umsatzes mit der IIer-Linie, in der Werbung ist der Apple II aber praktisch verschwunden, auf Messeständen dominiert der Mac. 1985 war für die Computerindustrie ein Krisenjahr, das Apple nur dank seines gesunden Finanzpolsters überstand. 1986 wird nicht weniger einfach werden, denn Apples Marktanteil hat sich innerhalb von zwei Jahren halbiert. Ein neuer Stern geht steil am Computerhimmel auf: Atari ST. Die Antwort von Apple läßt auf sich warten: Der IIx soll vielleicht im Herbst kommen, einige wollen ihn gar in Amerika schon gesehen haben. Aber kommt er wirklich? Apple sollte wissen: Äpfel, die über den Herbst hinaus am Baum hängen, fallen überreif herab und landen bestenfalls in der Mosterei.

Apple-Anlage von Herrn Fischer

Bücher

Das Macintosh Arbeitsbuch

20 betriebswirtschaftliche Beispiele aus der Praxis
von Dr. Dieter Nenner
1984, 189 S., kart., DM 58,--
Selbstverlag Dr. Dieter Nenner, München

Gliederung

Macintosh-Arbeitsweise mit Multiplan – Grafik erstellen mit Microsoft-Chart – Rechnungsstellung/Faktura – Automatische Faktura – Bankkredit mit Alternative – Textverarbeitung mit MacWrite – Kreditkundenbrief – Kleinlager, Inventur und Bestellwesen – Deckungsbeitrag-Vertrieb – Deckungsbeitragsrechnung-Marketing – Gebietsumsatz – Gewinn- und Verlustrechnung nach Großkunden oder Hauptumsatzträgern – Aktionsabsatzplan – Mediaplanung im Werbebereich – Jahresbudget mit Ergebnisrechnung – Aktions-Erfolgskontrolle – Statistische Auswertungen – Projektmanagement – Projektplan einer Gebäudeerrichtung: Neubau – Stückkostenkalkulation

Bemerkungen

Dieses im Selbstverlag erschienene Buch des ehemaligen Vertriebsleiters der Firma Apple enthält betriebswirtschaftliche Beispiele für die damals noch nicht eingedeutschten Programme Multiplan, MS-Chart, Mac-Project, Mac-Write und Lisa-Calc.

Das Floppybuch zum Apple II

von M. Czerwinski
1985, 264 S., kart., DM
Data Becker, Düsseldorf

Gliederung

Eigenschaften und Anwendungen einer Flopy – Unsortierte sequentielle Dateien – Sortierte Sequentielle Dateien – Random-Dateien – Such- und Sortierverfahren – Index-sequentielle Dateien – Verkettete Dateien – Programmierung eines Literaturverzeichnisses – Weitere DOS-Kommandos und Besonderheiten – Das System ProDOS

Bemerkungen

Dieses Buch ist im Grunde eine Einführung in DOS 3.3 für Applesoft-Progrmmierer mit einem kurzen Anhang zu ProDOS. Trotz des Titels „Floppybuch" wird auf die Hardware-Seite nicht eingegangen. Auch der maschinensprachliche RWTS-Zugriff wird nicht behandelt.

Apple IIc

Handbuch für Anwender und Programmierer
von Carl Ulrich Wassermann
1986, 324 S., kart., DM 35,--
Dr. Alfred Hüthig Verlag, Heidelberg

Gliederung

Der Apple IIc – Die Bedienung des Apple IIc – Applesoft BASIC – Andere Programmiersprachen – Anschluß externer Geräte – Weiterführende Informationen – Anhang

Bemerkungen

Dieses Buch enthält in den ersten 200 Seiten eine Einführung in Applesoft-BASIC unter besonderer Berücksichtigung des Apple IIc (Schnittstellen, Maus usw.). Im Anschluß daran wird in gedrängter Form auf den 65C02 und den IIc-Monitor eingegangen. Für IIc-Besitzer unentbehrlich.

Das ProDOS-Handbuch

von Karen Rice und Timothy Rice
1985, 272 S., kart., DM 48,--
Sybex-Verlag, Düsseldorf

Gliederung

Einführung in ProDOS – Die Dienstprogramme auf der ProDOS User's Disk – Die System-Dienstprogramm-Diskette für den Apple IIc – Verzeichnisse und ProDOS-Dateien – ProDOS-Befehle – ProDOS, Speicherbelegung und periphere Geräte – Textdateien und BASIC-Programmierung unter ProDOS – ProDOS und binäre Dateien – Grafik und Ton unter ProDOS – ProDOS und das Maschinesprache-Interfce – ProDos unter er Apple II – Anhang

Bemerkungen

Dieses Buch ist eine leichtverständliche, aber stark aufgeblähte und wenig Substanz bietende Einführung in ProDOS. Tips und Tricks sowie Utilities wird man vergeblich suchen. Vieles von dem, was die Autoren schreiben, haben sie offenbar blind den teilweise falschen Darstellungen in den ProDOS-Manuals der Firma Apple entnommen (z.B. INPUT von Strings mit Kommas; S. 115).

Programme zur Wahrscheinlichkeitsrechnung und Statistik

Apple-Version
von Heinrich Bockmeyer
1985, 125 S., kart., DM 34,--
Aulis Verlag Deubner, Köln

Gliederung

Datenerfassung, -verarbeitung und -darstellung – Binominalverteilung – Polynominalverteilung – Poisson-Verteilung – Hypergeometrische Verteilung – Normalverteilung – Verteilung mit Gauß-Kurvenanpassung – Unbestimmtheit (Entropie) von Zufallsexperienten – Regressionsanalyse – Liste der Programme

Bemerkungen

Dieses Buch enthält eine Vielzahl kurzer Applesoft-Programme mit HGR-Grafik für den Stochastik-Unterricht an Schulen, die den Apple II einsetzen.

Mein BASIC-Manual, Teil 1

Arbeitsunterlage für den Einsatz von Microrechnern, 1. Teil
von H. W. Papenkort
1985, 275 S., Loseblattsammlung, DM 24,80
Selbstverlag H.W. Papenkort, 6239 Eppstein 4

Bemerkungen

Dieses aus dem üblichen Rahmen fallende Loseblattwerk ist nicht leicht zu würdigen. Einerseits enthält es eine Fülle nützlicher Tips und Tricks zu Applesoft-Basic sowie allgemeinere Beiträge, die zum Schmökern einladen. Andererseits fehlt dem Buch eine sinnvolle Gliederung. Erheiternd ist die Typologie der Mikrocomputer-Freaks, die sich laut Papenkort in Löter, Knacker, Tüftler, Hacker, Perfektionisten, Avantgardisten u.a. einteilen lassen.

Arbeiten mit dem Macintosh

von Norbert Hesselmann
1984, 405 S., kart., DM 54,-
Sybex-Verlag, Düsseldorf

Gliederung

Die Macintosh-Story – Erste Kontakte – Der elektronische Schreibtisch des Macintosh – Arbeiten mit dem System – Der Macintosh intern – Arbeiten mit MacPaint – Arbeiten mit dem Textprogramm MacWrite – Programmieren in Microsoft-BASIC – Software für den Macintosh – Anhang

Bemerkungen

Eine klar und systematisch geschriebene Einführung in den Umgang mit dem Macintosh, die zugleich eine Kurzeinführung in MS-BASIC, Version 1.0, enthält. Dr. Hesselmann ist übrigens Lektor im Sybex-Verlag.

Das große BASIC-Lernbuch

BASIC lernen, verstehen, anwenden auf dem Personalcomputer
von W. Meyer und K. Schacht
1985, 489 S., kart., DM 36,--
Carl Hanser Verlag, München

Gliederung

Einführung in die Programmierung – Sprachelemente von BASIC – BASIC-Befehlsvorrat – Logische Operatoren – Großer BASIC-Test mit EDV-Auswertung – Programmierung allgemeiner und technischer Problemstellungen – Gestaltung der Ausgabe – Zeichenkettenverarbeitung – Unterprogramme – Programmierung komplexer Problemstellungen – Anhang

Bemerkungen

Eine von Studienräten für den Schulunterricht geschriebene, sehr umfangreiche, zweispaltig im Querformat gesetzte und mit zahlreichen Flußdiagrammen versehene Einführung in Minimal-BASIC. In Anbetracht des Umfangs und der Satzqualität eines der preiswertesten Computer-Bücher überhaupt!

Now That You Know Apple Assembly Language: What Can You Do With It?

von Jules H. Gilder
1985, 186 S., kart., $15,--
DataCraft, New York
In Deutschland Auslieferung über Hofacker-Verlag

Gliederung

Getting information out of your computer – Getting information into your computer – Stealing control of the output – Stealing control of the input – Using sound in your programs – Learning to use the ampersand – Expanding Applesoft Basic – ASCII Code – Applesoft token list – Shift key modification instructions – Adapting programs to work with ProDOS

Bemerkungen

Dieses englische Buch wendet sich an „fortgeschrittene Anfänger", die sich bereits mit den Grundlagen der 6502-Programmierung befaßt haben und nunmehr nützliche, aber noch relativ leichtverständliche Anwendungsprogramme suchen.

Spielprogramme für den Apple IIe

Spiele sowie Anleitungen, Techniken und Unterprogramme für die Eigenentwicklung von Spielen
von Howard Franklin, Joanne Koltnow und LeRoy Finkel
1984, 124 S., kart., DM
Vieweg-Verlag, Wiesbaden

Gliederung

Melodien und Klangeffekte – Die niederauflösende Grafik – Bilder in Lores-Grafik – Hochauflösende Grafik – Programme zur Dateneingabe – Wortspiele – Weitere Spiele – Renumber/Merge-Hilfsprogramm – Randbemerkungen zum Programmieren der Spiele – Hinweise zu den Programmlistings – Das Entwickeln von Programmen

Bemerkungen

Dieses Buch enthält zahlreiche Applesoft-Module für die Entwicklung einfacher Spielprogramme.

BASIC-Wegweiser für den Apple II

Datenverarbeitung mit Applesoft-BASIC für Apple II/IIe und kompatible Mikrocomputer
von Ekkehard Kaier
1984, 194 S., kart., DM 32,-
Vieweg-Verlag, Braunschweig

Gliederung

Computer allgemein – Arbeitsweise des Apple IIe und Apple II – Programmierung in Applesoft-Basic

Bemerkungen

Dieses Buch ist eine Einführung in Applesoft-BASIC unter besonderer Berücksichtigung der unterrichtlichen Belange von kaufmännischen Berufsschulen und Wirtschaftsgymnasien.

Programme in Microsoft-BASIC

Band 1: Mathematik
Fertige Programme, Anregungen und Erläuterungen in BASIC
von D. Herrmann und G. Schnellhardt
1985, 190 S., geb., DM
IWT Verlag, Vatterstetten
Gliederung
Mehr-Register-Arithmetik – Zahlentheorie – Kombinatorik – Algebra – Geometrie – Numerische Mathematik – Anhang
Bemerkungen
Dieses Buch ist inhaltlich weitgehend identisch mit dem bereits im Peeker 10/85, S. 67 besprochenen Titel „Mathematik auf dem Apple II" mit dem Unterschied, daß die Programme in dem vorliegenden Werk in MBASIC geschrieben wurden.

Apple Grafik

Anwenderhandbuch
von Harold J. Baily und J. Edward Kerlin
1985, 448 S., kart., DM 68,--
Carl Hanser Verlag, München
Gliederung
Niedrig auflösende Grafik – Hoch auflösende Grafik – Grafik für Fortgeschrittene – Zweidimensionale Grafik – Dreidimensionale Grafik – Anhang
Bemerkungen
Eine umfangreiche Einführung in die Lores- und Hires-Grafik für Applesoft-Programmierer mit zahlreichen Übungsaufgaben, die sich für den Schulunterricht eignen.

Apple II-DOS 3.3-Assembler-Listing

von Franz Engels
1985, 108 S., kart., DM 59,-
Röckrath Mikrocomputer, Aachen
Gliederung
DOS Assembler Listing – XREF Listing – DOS-Änderung/Lesefehlererzähler – DOS-Tip/Turnkey – Init ohne DOS – Noch ein zusätzliches Programm
Bemerkungen
Dieses mit dem Matrixdrucker in Großbuchstaben gesetzte Bändchen enthält den in englischer Sprache kommentierten Quellcode des Original-DOS-3.3. Der Autor entschuldigt sich im Vorwort, das bezeichnenderweise in *deutscher* Sprache geschrieben ist, daß er sich des Englischen bedienen mußte, um „schauderhaftes Kauderwelsch" in Deutsch zu vermeiden. Dazu hätte er allerdings erst einmal Englisch lernen müssen! Einige Stilblüten: „Several appendages later added" (S. 59); „If startmarker found fall through" (S. 67); „Save it around loading RAM-Applesoft" (S. 21). Daß manche Autoren die deutsche Sprache meiden, ist schon betrüblich genug. Aber daß sie uns dann Pidgin-Englisch vorsetzen, ist wohl der Gipfel.

Apple-Works

von V. Botta, Chr. Lange, K. Zimmermann 1985, 2 Bände je 264 S., kart., je DM 49,– Te-Wi Verlag, München
Mit dem Programm Appleworks ist es möglich, auf dem Apple IIe/c Textverarbeitung, Datenbank und Tabellenkalkulation in einem einzigen benutzerfreundlichen Paket anzuwenden, das für alle kleinen und mittleren Anwendungen ausreicht. Es war klar, daß bald Bücher über Appleworks erscheinen würden, obschon die Firma Apple ein brauchbares Handbuch und Übungsdisketten zu Appleworks mitliefert.
Die drei Autoren des Te-Wi-Verlags zeichnen noch einmal ausführlich das Original-Handbuch nach, wobei durch die Neuformulierung des Textes an der einen oder anderen Stelle zusätzliche Informationen überkommen. Das gilt besonders bei der Druckersteuerung, die mit ausführlichen Beispielen erklärt wird. Auf fast jeder Seite wird der Text durch eine Abbildung oder einen Bildschirmausdruck unterstützt, so daß hier wirklich ein anschauliches Werk entstanden ist, in dem man Dank des Stichwortverzeichnisses auch schnell etwas findet. Die beiden Bände werden wegen der vollständigen Befehls- und Installationsbeschreibung sicher auch viele Freunde bei den Besitzern zwielichtiger Kopien finden.
Zu allen Bereichen von Appleworks sind Anwenderbeispiele aufgeführt, die aber stets recht kurz ausfallen. Sie reichen aber aus, um eine Idee von der Leistungsfähigkeit des Programms zu erhalten und um z. B. schwierigere Programmierungen im Rechenblatt zu verstehen.
Zusammen haben beide Bände 528 Seiten dicken Papiers. Ganze 121 Seiten des 1. Bandes tauchen Wort für Wort identisch im 2. Band noch einmal auf. Weitere 56 Seiten behandeln die Datenfernübertragung mit Access II, das nicht zum Lieferumfang vom Appleworks gehört, sondern für über 250 DM separat bezogen werden muß. Was im Band 2 an direkter Information über Appleworks bleibt, sind 61 Seiten über die Textverarbeitung und 8 Seiten über Dateiumwandlungen. Alles andere ist entweder doppelt, gehört zu dem Programm Access II oder betrifft das Inhaltsverzeichnis. Nichts ist perfekt.

Peeker-Sammeldisk #17

(Kombinierte DOS-3.3- und ProDOS-Diskette; Heft 5/1986; Einzelpreis DM 28,-; Fortsetzungspreis DM 20,-)
Achtung: Solange Diskette schreibgeschützt bleibt, kann sie unter DOS *und* ProDOS benutzt werden.
(1) = Zweck; (2) = Heft/Seitenzahl; (3) = Gerätekonfiguration; (4) = Betriebssystem; (5) = Programmstart; (6) = Sonstiges

ProDOS-Teil

Warnung: Solange Dummy- oder Füller-Datei ED.FRAME.TXT.0 nicht überschrieben wird, bleiben DOS-3.3-Dateien auf den Spuren 17-34 erhalten. Daher vor Assemblierung von ED.FRAME.TXT Disk #17 kopieren.

ED.FRAME.TXT *
ED.DISKIO.TXT *
ED.FUNCS1.TXT *
ED.FUNCS2.TXT *
ED.CMDS.TXT *
ED.CMDS.ADD *
ED.FUNCS1.TXT1
ED.FUNCS2.TXT1
ED.FUNCS2.TXT2
ED.CMDS.TXT1
ED.CMDS.TXT2
ED.CMDS.TXT3
ED.FRAME.TXT.0 (später EDIT)
(1) Diskетteneditor für ProDOS; (2) Erster Teil in Heft 5/86, S. 8; es folgen noch zwei weitere Teile in den nachfolgenden Peeker-Ausgaben; (3) Apple II+/e/c; (4) ProDOS 1.0.1, 1.0.2, 1.1.1; (5) Objektcode EDIT ist aus Platzgründen auf Disk #18 enthalten. Wenn Sie jetzt schon mit dem Editor arbeiten wollen, müssen Sie den ProDOS-EDASM-Assembler besitzen und wie folgt verfahren: EDASM von Drive 1 laden und Kopie der Disk #17 in Drive 2 einlegen, dann Befehl ASM/PEEKER.DISK17/ED-.FRAME.TXT eingeben. Assemblierung erfolgt vollautomatisch und erzeugt ED.FRAME.TXT.0, das den kompletten Objektcode darstellt und in EDIT umbenannt werden kann. Später unter ProDOS (niemals unter DOS 3.3!) starten mit BRUN EDIT. (6) 1-Drive-Besitzer kopieren die mit * markierten Quelltexte auf ihre EDASM-Arbeitsdiskette und assemblieren dann mit ASM/ED-.FRAME.TXT. Die ohne *, aber mit Suffix „1", „2" oder „3" versehenen Zwischendateien beziehen sie sich die 3 Teile der Edit-Aufsatzes.

DOS-3.3-Teil

WM86.PAS
VORSPANN.PAS
ENTSCHEI.PAS

(1) Simulation der Fußball-Weltmeisterschaft 1986; (2) Heft 5/86, S. 6; (3) Apple II+/e/c mit CP/M- und 80-Z/Z-Karte; (4) CP/M 2.23 oder 2.26, d.h. mit mindestens 60K; Turbo-Pascal 3.0; (5) DOS-3.3-Dateien mit APDOS auf Ihre Turbo-Arbeitsdiskette konvertieren und compilieren. Wegen der Dateinamen-Änderung siehe Heft.

PRINT.ASS.TEXT
FASTPRINT.TEXT
DEMOPRINT.TEXT

(1) Schnelle Bildschirmausgabe in Apple-Pascal; (2) Heft 5/86, S. 29; (3) Apple IIe mit 80 Z/Z oder IIc (nicht II+!); (4) Pascal 1.1 oder 1.2; (5) DOS-3.3-Textfiles mit GETDOS von Sammeldisk #15 auf Ihre Pascal-Arbeitsdiskette konvertieren. Zur Assemblierung und Einbindung in Library siehe Heft.

LISTCODEF.TEXT
LEVEL0.TEXT
ASSEM.TEXT
(1) Analyse der Struktur von Apple-Pascal-Files; (2) Heft 5/86, S. 34; (3) Apple II+/e/c; (4) Pascal 1.1 oder 1.2; (5) DOS-3.3-Textfiles mit GETDOS von Sammeldisk #15 auf Ihre Pascal-Arbeitsdiskette konvertieren. Zur Assemblierung und Anwendung siehe Heft.

PI.START
T.PI
PI
(1) Berechnung der Zahl Pi (bis ca. 15.000 Stellen); (2) Heft 5/86, S. 42; (3) Apple II+/e/c; (4) DOS 3.3; (5) RUN PI.START, dann gewünschte Stellenanzahl eingeben.

PATCH.STARTER
SQ.PATCHER
(1) Modifikation des Kopierprogramm „Superquick" für 160-Spur-Erphi-Laufwerke für diejenigen, die „Superquick" vor dem 1.3.86 bestellt haben; (2) Heft 5/86, S. 59; (5) Näheres siehe Heft.

MDB.KOPY
Backup-Programm für DOS 3.3 für Megaboard-Festplatte: Befindet sich mit einem weiteren Backup-Programm für ProDOS auf Sammeldisk #18.

Hüthig Software Service
Postfach 10 28 69 Heidelberg

Dieter Geiß beantwortet UCSD-Pascal-Leserbriefe

Hinweis: Bitte beachten Sie, daß wir wegen der Fülle der Zuschriften nur einen kleinen Teil der Leserbriefe beantworten können. Ihre Briefe sollten sich auf die Aufsätze beziehen und nicht „globaler Natur" sein. Fragen in der Art „Wie kann ich Quickfile decompilieren?" (s.u.) sind bei nüchterner Betrachtung schlechthin eine Zumutung, denn dies würde Monate dauern. Schließlich müssen wir einem Spezialisten wie Dieter Geiß für die Beantwortung der Zuschriften ein Honorar zahlen. U.Stiehl

SYSTEM.SYNTAX eingedeutscht

Für die Schule habe ich versucht, die Fehlermeldungen im UCSD-Pascal ins Deutsche zu übersetzen. Dazu habe ich den File SYSTEM.SYNTAX von Apple-Pascal 1.1 vom Filer aus mit C)hange in einen Textfile umgewandelt und die Fehlermeldungen im Editor zeilenweise ins Deutsche übersetzt. Anschließend habe ich den geänderten File wieder zu SYSTEM.SYNTAX umbenannt. Leider erscheinen jetzt nicht mehr alle Fehlermeldungen in der Kopfzeile des Editors, sondern nur noch ihre Nummern. SYSTEM.SYNTAX ist laut Filer im Original ein File vom Typ .Data, während das geänderte SYSTEM.SYNTAX ein File vom Typ .Text ist. Meine ersten groben Nachforschungen am Original-SYSTEM.SYNAX ergaben allerdings, daß er zwar dem Namen nach vom Typ .Data ist, jedoch sieht der Anfang sehr nach dem typischen Beginn eines Textfiles aus. Könnten Sie mir bitte mitteilen, wie ich bei der Übersetzung von SYSTEM.SYNTAX vorgehen muß, damit alle Fehlermeldungen in der Kopfzeile erscheinen?
Wolfgang Kurz, Melle

Antwort:
Wie Sie richtig erkannt haben, ist der File SYSTEM.SYNTAX in der Tat ein Textfile, auch wenn er als Data-File auf dem Directory ausgewiesen ist. Nach Umbenennung, die aber nicht unbedingt notwendig ist, kann der File mit Hilfe des Editors geändert werden, also z.B. ins Deutsche übersetzt werden. Das Problem, daß der Editor dann nicht mehr alle Fehlermeldungen im File SYSTEM.SYNTAX korrekt findet, ist weniger auf den geänderten File zurückzuführen, sondern vielmehr auf die wirklich ungewöhnlich (schlechte) Weise, wie der Editor versucht, aus der Fehlernummer auf den Fehlertext zu schließen. Wie bewerkstelligt der Editor dies? Um das herauszufinden, muß man sich einen Teil des Editors (das Segment PUTSYNTAX) mit einem P-Code-Disassembler ansehen und zurückübersetzen. Dann erkennt man, daß der Editor feste Konstanten in seinem Code eingebaut hat. Diese Konstanten geben an, welche Fehler sich in welchen Blöcken von SYSTEM.SYNTAX befinden. Ändern sich die Längen der einzelnen Zeilen durch Übersetzung in eine andere Sprache, so erkennt der Editor nicht mehr, in welchen Blöcken sich welche Fehlermeldungen befinden. Folgende Voraussetzungen müssen erfüllt sein, damit der Editor alle Fehlermeldungen findet. Dabei gilt als Fileanfang von SYSTEM.SYNTAX der relative Block 0.

Pascal 1.1: Alle Fehler, für die gilt Fehlernummer <= 104, müssen in den Blöcken 2 und 3 zu finden sein, alle Fehler mit Nummer <= 126 in den Blöcken 4 und 5, mit Nummer <= 151 in den Blöcken 6 und 7, mit Nummer <= 185 in den Blöcken 8 und 9, mit Nummer <= 302 in den Blöcken 10 und 11, alle anderen Fehler in den Blöcken 12 und 13.
Pascal 1.2: Fehler <= 108 in den Böcken 2 und 3, Fehler <= 136 in den Blöcken 4 und 5, Fehler <= 169 in den Blöcken 6 und 7, Fehler <= 301 in den Blöcken 8 und 9, alle anderen Fehler in den Blöcken 10 und 11. Man beachte, daß bei Version 1.2 der File SYSTEM.SYNTAX nur 12 Blöcke lang ist.

Ein weiteres Problem ist das Vorhandensein der DLE-Codes in Textfiles. Sie dürfen im SYSTEM-SYNTAX nicht vorhanden sein.
Es gibt nun grundsätzlich drei Möglichkeiten, diesem Dilemma zu entgehen:
1. Man wählt die Übersetzung so geschickt, daß die oben angegeben Bedingungen für die Fehler erhalten bleiben. Nach der Übersetzung dürfen sich keine DLE-Codes mehr im Textfile befinden.
2. Man benutzt einen Editor, der keinen solchen unflexiblen Algorithmus verwendet, wie z.B. den ASE (Advanced System Editor), welcher immer alle Fehlermeldungen erkennt, egal in welchen Blökken sie sich befinden und ob ihnen ein DLE vorausgeht.
3. Man patcht den SYSTEM.EDITOR an den Stellen, wo die Konstanten für die Fehlermeldungen stehen (wenn man weiß, wo das ist). Außerdem dürfen sich keine DLEs im SYSTEM.SYNTAX befinden.
Die erste Möglichkeit scheitert eventuell daran, daß die Fehlermeldungen in anderen Sprachen als dem Englischen zu lang oder zu kurz werden, die zweite daran, daß kein solcher Editor zur Verfügung steht. Die dritte Möglichkeit müßte sehr ausführlich erklärt werden, dabei gibt es einige P-Code-spezifische Dinge zu beachten, bei denen ich Ihnen nicht zumuten möchte, sie zu verstehen. Aber ich mache Ihnen folgenden Vorschlag: Sie senden der Redaktion eine Diskette mit Ihrem geänderten SYSTEM.SYNTAX und – aus urheberrechtlichen Gründen – auch den SYSTEM.EDITOR von Pascal 1.1, 1.2 oder beide. Ich entferne durch ein Hilfsprogramm die überflüssigen DLE-Codes aus dem File SYSTEM.SYNTAX und patche den Editor so, daß er Ihren File ordnungsgemäß erkennt.

SCREENBIT-Funktion

Beim Versuch, mit Hilfe der UCSD-Funktion SCREENBIT eine Hardcopy zu erstellen, ergaben sich unerwartete Schwierigkeiten: War bei der Ausgabe an einen Epson FX-100 im Byte nur das Bit 2↑4 gesetzt (das unterste auf der Spalte habe den Wert 2↑0), so begann der Drucker „durchzudrehen". Wird im Ausgabewort irgendein anderes Bit zusätzlich gesetzt, gibt es einen korrekten Druck.
Die Hexdarstellung des Datenstromes zum Drucker zeigt für den kritischen Fall zusätzliche Leerzeichen (=$20), die mit einiger Sicherheit nicht auf das Druckerinterface oder den Drucker zurückzuführen sind. Mein Verdacht richtet sich auf das Pascal-System: Eine Codeprozedur zum Pascalprogramm, die die Grafikseite ausdruckt, lieferte das erwartete Ergebnis.
Zum Test der Ausgabe wurde ein Programm geschrieben, das bei einem Eingabewert von 16 den Fehler zuverlässig reproduziert. Verwendete Hardware: IIe, FX-100 mit Epson 8132W Druckerinterface.
Sollten Sie eine Lösung des Problems kennen und außerdem der Meinung sein, auch andere Anwender könnten daran interessiert sein, dann bitte ich um eine Antwort im Peeker.
Ludwig Kellner, Karlsruhe

Antwort:
Das Problem liegt darin, daß die Zahl 16 (ASCII DLE) eine Sonderstellung bei Textfiles einnimmt. Sie wird benutzt, um die Files komprimiert abzuspeichern, wenn es führende Leerzeichen in einer Zeile gibt (vgl. Apple Pascal Operating System Reference Manual, S. 266). Wird nun bei der Ausgabe ein DLE-Zeichen geschickt, so wird das nachfolgende Zeichen als Anzahl der Leerzeichen interpretiert, nachdem von diesem die Zahl 32 subtrahiert wurde. Eine Folge von 16, 38 entspricht also 6 Leerzeichen. Das Umwandeln der DLE-Codes kann aber verhindert werden, wenn man statt mit WRITE mit UNITWRITE arbeitet (vgl. Peeker 2/86, S. 54, Listing 2). In Ihrem Testprogramm müßten Sie das
WRITE(DR, CHR(X))
durch ein
UNITWRITE(6, X, 1, 0, 60)
ersetzen, damit Ihr Programm läuft. Für eine schnelle und in vielen Formaten mögliche Hardcopy möchte ich auf die Pascal-Version von SUPERDUMP verweisen, die in Vorbereitung ist und im Peeker veröffentlicht werden wird.

Gemini-Grafik-Ausdruck

Zunächst einmal darf ich Ihnen versichern, daß das Lob für Ihre Zeitschrift aus zahlreichen Leserbriefen auch durch mich nicht abreißen soll. Auch ich kann mir keine wertvollere Literatur zur Programmierung des Apple II vorstellen.
Der Artikel über Assembler-Ein/Ausgage in Peeker, Heft 2, S. 47 hat ein Problem in meiner Programmierung aufgeworfen, bei dessen Lösung Sie mir hoffentlich helfen können: An meinem II+ habe ich über eine parallele Schnittstelle einen Drucker vom Typ Gemini 10x angeschlossen. Unter DOS und im Apple Monitor habe ich die Möglichkeit, nach PR#1 (Printer Karte in Slot 1) die HGR-Seiten mit CTRL-Q auf dem Drukker ausgeben zu lassen. Dabei kann ich die Form des Ausdrucks durch einen Poke auf 1912 + Slotnr. variieren. Diese Möglichkeit wird von der Karte eröffnet. Unter UCSD-Pascal habe ich bisher kein

Glück gehabt, diesen Befehl zu übergeben. Versuche mit BLOCKWRITE und UNWRITE sind gescheitert, und auch die in Ihrem Artikel beschriebene Assemblerroutine brachte keinen Erfolg. Wahrscheinich wird vom Pascal-Druckertreiber das CTRL-Q uminterpretiert oder abgefangen.
Bisher lese ich also die HGR-Seite Bit für Bit aus und schicke die entsprechenden Bytes an den Drukker (es gibt Druckbefehle, mit denen ich einzelne Nadeln ansprechen kann). Das dauert allerdings ziemlich lange.
Detlef Zieleznik, Ratingen

Antwort:
Der Gemini 10x ist meines Wissens Epson-kompatibel. Um die Grafik von Pascal aus auf den Drucker zu bringen, benutzt man am besten ein schnelles Maschinenprogramm, wie es von SUPERDUMP (Pascal) zur Verfügung gestellt wird (s. o.). SUPERDUMP läuft unter anderem auch mit Epson-Druckern mit Parallel-Interface, so daß ich große Hoffnungen hege, daß es auch mit Ihrem Gemini 10x einwandfrei funktioniert.

Quickfile decompilieren?
Ihre Erläuterungen zum P-Code-Disassembler im Peeker 2/86 veranlassen mich zur Frage, ob mit Ihrer Hilfe ein bestehendes Problem beseitigt werden kann.
Ich benutze das in Pascal 1.1 geschriebene Programm „Quickfile IIe" zusammen mit dem Imagewriter und der Super-Serial-Card. Leider hat dieses Programm einen Fehler bei der Ausgabe im Listen-Format. Zwischen den Datenzeilen erscheinen immer unschöne Leerzeilen – auch wenn diese kürzer sind als der Seiten-Kopf – und der Druck-Parameter für den Zeilenabstand arbeitet unkorrekt. Da diese Leerzeilen nicht im Seiten-Kopf erscheinen, liegt es wohl nicht an der Drucker-Konfiguration. Trotz mehrfacher Reklamation bei Apple in München kann oder will man mir keine fehlerfreie Disketten-Version zur Verfügung stellen.
Hier drängt sich nun die Frage auf, ob Sie eine Möglichkeit sehen, mit Hilfe Ihres P-Code-Disassemblers ein änderbares Quell-Programm zu erzeugen. Ich denke hier an die Überlassung des Programms gegen zu vereinbarendes Entgelt oder aber an einen Besuch in Ihrem Hause. Da ich inzwischen selbst über das Pascal-Betriebssystem 1.2 verfüge, kann die anschließende Korrektur selbst vorgenommen werden. Evtl. versuche ich dann noch, das Programm so zu ändern, daß im Etiketten-Format mehrbahnige Aufkleber gedruckt werden können.
Gerhard Bletzer, Hirschberg

Antwort:
Ich besitze zwar Quickfile IIe nicht, nehme aber an, daß ein Programm zu diesem Preis (279.- DM) professionell genug ist, daß solche Fehler, wie Sie sie beschreiben, normalerweise nicht auftreten dürfen. Quickfile ist mit Sicherheit von den Programm-Autoren durchgetetstet worden. Vielleicht ist Ihre Version defekt.
Mein P-Code-Disassembler erzeugt keine Quellprogramme, sondern listet den P-Code mit den mnemotechnischen Abkürzungen auf und zeigt noch einige andere Dinge an. Quellprogramme kann ich nur durch mühsame Kleinarbeit von Hand erzeugen. Ein Programm, das dieses automatisch durchführt, ist sehr kompliziert und wurde zwar von mir in Angriff genommen, aber nach 8000 Zeilen kann es bis jetzt „nur" die Struktur und die Datentypen eines Programms herausfinden, wobei es bei geschachtelten Records mit Case-Varianten Schwierigkeiten gibt, da die Datenstrukturen verschiedene Gestalt annehmen können. Ich werde dieses Programm nicht weiter entwickeln, da mir dazu einfach die Zeit fehlt. Aus demselben Grund kann ich mich nicht darauf einlassen, irgendwelche längeren Codefiles zurückzuübersetzen. (Rechnen Sie mit einem Monat für 4K).
Ein Programm wie Ihre Quickfile-Version, das nicht ordnungsgemäß funktioniert, würde ich zurückgeben. Vielleicht gibt es aber auch eine Option oder ein Flag, das man setzen kann, so daß die zusätzlichen Leerzeilen im Ausdruck erscheinen. In diesem Fall sollten Sie das Handbuch noch einmal genau studieren.

Diverse Fragen
Ihre im Peeker begonnene und – hoffentlich – fortzusetzende Serie „Tips und Tricks in Pascal" bietet, wie Ihnen schon in zahlreichen Leserbriefen bestätigt wurde, eine Menge Informationen, aber, was ich vor allem begrüße, Erklärungen. Die Teile 2, 4 und 5 möchte ich hier besonders hervorheben, die über die normalen Abhandlungen in Handbüchern oder anderweitig erschienenen Veröffentlichungen hinausgehen. Die ergänzenden Programmbeispiele – es könnten von mir aus mehr sein – erhellen die Hinweise, so daß diese Form der Präsentation Ihrer Studien beizubehalten wäre.
Zum Teil 5 – Assembler-Ein/Ausgabe in UCSD-Pascal – kamen mir ein oder zwei Gedanken, die ich an Sie mit der Bitte der Erklärung weiterleiten möchte:
1. S. 48: **File of Integer** benötigt mindestens 300 Words plus 1 für die Größe des Datentyps. Wieso passen nur 256 Integer in einen Block? Da jede Integer nur 1 Word beansprucht, berechne ich 300:1 = 300 Integer. Was übersehe ich?
2. Wie und wo brauche ich die **Unitbefehle** bei der Assemblerprogrammierung?
3. Schon seit längerem benutze ich die Sprungadressen **$FF00 und $FF03** in Assemblerprozeduren, aber ohne vorher das X-Register mit den entsprechenden Parametern, 0 = read usw., zu belegen (Unwissenheit). Bisher traten jedoch keine Fehler in den Programmen auf. Ist das Zufall oder ist die Belegung des Registers nicht von Bedeutung? Wenn der Belegung Beachtung zu schenken ist, was bewirkt die Initalisierung? Wieso sind bei mir bislang keine Fehler aufgetaucht?
4. Vielleicht könnten Sie an der einen oder anderen Stelle Ihrer Artikel typische praktische Anwendungsbeispiele einbringen. Im Teil 5 würde sich die Beantwortung der Frage aufdrängen, wozu soll ich in der Programmiersprache Pascal überhaupt **Assemblerprozeduren** schreiben? Wie lauten typische, charakteristische Anwendungsbeispiele? Programmlistings wären hier natürlich auch schön und könnten der Erklärung und Handhabung dienen.
Sollten Sie in Ihrer Serie hierauf nicht mehr eingehen, so würde ich mich freuen, wenn Sie mir solche vielleicht aus Ihrer Praxis mitteilen, um mein bescheidenes Reservoir aufzufüllen (entsprechende Unkosten ersetze ich natürlich).
5. Wo liegen und was sind **Screenholes**? (Video-80-Zeichenkarte)
6. Themen, die mich weiterhin interessieren würden:
i) Wie bringe ich in UCSD-Pascal die **Graphic** auf den Drucker?
ii) Wie schreibe ich in UCSD-Pascal 1.2 einen Treiber einer 256K **RAM-Disk**?
iii) Software-Problem: Wie programmiere ich einen **Compiler** für eine selbstdefinierte Sprache in Pascal?
Ich hoffe, ich habe Ihre Zeit nicht zu sehr strapaziert und Sie mit meinen Problemen gelangweilt; sehen Sie meinen Brief als Rückmeldung von der Basis. Froh wäre ich, wenn Sie auf die ein oder andere Frage eine Antwort wüßten.
Udo Memmert, Köln

Antworten:
Zu Frage 1: Der File Information Block belegt insgesamt 300 Words. In ihm ist auch der Puffer, der 512 Bytes = 256 Words belegt, in welchen 256 Integer-Zahlen passen. Genaue Informationen entnehmen Sie dem Teil 6 meiner Serie.
Zu Frage 2: Zum besseren Verständnis der drei I/O-Ebenen.
Zu Frage 3: Die Belegung des X-Registers ist nicht nur guter Programmierstil, sondern gehört zu den Konventionen des BIOS. Als Beispiel wollen wir die Ausgabe an den Drucker betrachten. Normalerweise braucht das X-Register nicht gesetzt zu werden, wenn man über PRINTERWRITE springt. Nun ist es aber möglich, daß ohne das Wissen des Assemblerprogrammierers ein neuer Treiber für einen Drucker angeschlossen wurde oder einfach ein Druckerpuffer installiert wurde, wie das z.B. bei mir der Fall ist. Manche Drucker-Treiber bzw. der Lader des Druckertreibers ersetzt dann die Einsprungadressen von PRINTERINIT, PRINTERREAD, PRINTERWRITE und PRINTERSTATUS, wobei alle vier Adressen zu *derselben* Speicherzelle zeigen. Dort liegt im allgemeinen ein Programm, welches zuerst das X-Register inspiziert und aufgrund seines Inhalts zur entsprechenden Routine für Read, Write, Init oder Status verzweigt.
Wenn bei Ihnen keine Fehler auftreten, weil Sie das X-Register nicht setzen, so liegt das einfach daran, daß bei Ihnen keine speziellen Treiber für Bildschirm oder Drucker etc. angeschlossen sind, sonst würde Ihre Assemblerroutine auf die Nase fallen, wenn nicht zufällig im X-Register der richtige Wert stehen würde.
Zu Frage 4: Die Lösung von so manchen Problemen läßt sich oft nur in Assembler noch effizient verwirklichen. Als Paradebeispiel läßt sich immer wieder die Grafik heranziehen. Die TurtleGraphics-Unit besteht zu einem Großteil aus Assemblerprozeduren, ebenso die Applestuff-Unit. Versuchen Sie einmal, eine Linie von der linken unteren zur rechten oberen Ecke in der Hires-Grafik zu zeichnen, wobei Sie nur Pascal-Befehle verwenden. Messen Sie dann die Zeit und vergleichen Sie sie mit einer Befehlssequenz aus der TurtleGraphics. Auch die Ausgabe von

Hires-Bildern auf den Drucker kann nur in Assembler noch einigermaßen schnell vonstatten gehen. Auch die Note-Prozedur in der Applestuff-Unit könnte nicht in Pascal geschrieben werden, sonst käme man über ein dumpfes Brummen nicht hinaus. In Teil 8 meiner Serie werde ich eine Unit vorstellen, die einige Assemblerprozeduren aufweist, von denen eine (Addr) in Pascal gar nicht zu realisieren wäre.
Zu Frage 5: Screenholes sind Speicherzellen, die zwar im RAM-Bereich des Bildschirms liegen ($400-$7FF), aber dennoch nicht benötigt werden, da der Bildschirm nicht 1024, sondern nur 960 Zeichen darstellt. Diese Speicherzellen werden durch ein Löschen des Bildschirms nicht berührt und dienen den Zusatzkarten als sichere Speicherplätze für ihre Variablen. Welche Variablen in den Screenholes untergebracht sind, hängt von der Karte ab, die in dem jeweiligen Slot steckt. Für jeden Slot sind genau 8 Bytes reserviert, die sich in den Speicherzellen 478+n, 4F8+n, 578+n, 5F8+n, 678+n, 6F8+n, 778+n und 7F8+n befinden, wobei n die Nummer des Slots ist.

Zu Frage 6:
i) Die Antwort ist die gleiche wie bei zwei der oben abgedruckten Leserbriefe: Mit der SUPERDUMP-Version für Pascal.
ii) Die Antwort lautet: Genauso wie in Pascal 1.1. Und für diese Version wurde bereits ein Treiber im Peeker 6/85 vorgestellt. Eine 256K-RAM-Karte hat selbstverständlich mehr Blöcke (512), und auch der Zugriff auf die Karte hängt von der Herstellerfirma ab. Deswegen ist die Frage nicht allgemein zu beantworten. In dem o.g. Peeker-Heft finden Sie allerdings alle Hinweise, die Sie brauchen, um einen RAM-Disk-Treiber zu installieren.
iii) Die Antwort auf diese Frage ist wohl nicht in wenigen Sätzen zu behandeln. Ich bin mir auch nicht sicher, ob Sie nun wissen wollen, wie man überhaupt programmiert (vor allem längere Programme), wie man eine Sprache definiert, oder wie man Compiler baut. Die Antwort wäre vielleicht folgende: Sie studieren einige Semester Informatik. In den ersten beiden Semestern lernen Sie, wie man Algorithmen entwirft, sie verifiziert und implementiert, im dritten Semester hören Sie dann etwas von Erzeugung, Akzeption und syntaktischer Analyse formaler Sprachen, und mit diesem Wissen ausgerüstet sind Sie ab dem fünften Semester bereit, die Vorlesung „Übersetzerbau" und im sechsten Semester „Ausgewählte Kapitel aus dem Übersetzerbau" zu hören. Ein Übersetzerbaupraktikum und ein Besuch der Vorlesungen „Syntaxanalyse" und „Semantik" wären natürlich ebenfalls von Vorteil. Falls Sie das nicht wollen, so sollten Sie sich zumindest ein erstaunlich dünnes Büchlein von Nikolaus Wirth, dem Erfinder von Pascal und Modula zulegen, das den Namen „Compilerbau" trägt. Dort sind die wichtigsten Dinge zu finden, wie Definieren einer Sprache mit Hilfe der Backus-Naur-Form, Parser, Syntaxanalyse und sogar Codeerzeugung.
Ich hoffe, auf Ihre Fragen ausreichende Antwort gegeben zu haben.

Fotofiles
Ich bitte um Information, ob mit Hilfe von Apple-Business-Graphics erstellte Fotofiles vom TurtleGraphics-Library-Paket geladen und auf dem Imagewriter ausgedruckt werden können (auf Apple IIc).
Dieter Hinderberger, Salach

Antwort:
Das Programm Apple Business-Graphics kenne ich nicht, so daß ich nicht weiß, welches Format die dort abgespeicherten Fotofiles haben. Da ich aber annehme, daß es sich um reine Bit-Images der Hires-Page handelt, müßte es sich mit der neuen TurtleGraphics durch einen Aufruf von LoadHires laden lassen. Um die Anhängung des Suffix „.GRAF" zu verhindern, hängen Sie bitte an den Namen des zu ladenden Files einen Punkt an. Zum Ausdrucken der Hires-Page benutzen Sie bitte die Pascal-Version von SUPERDUMP, die im Peeker erscheinen wird.

Kommunikationsprogramm
In der Textverarbeitung, die ich mit Hilfe meines Apple IIc betreibe, arbeite ich in der Regel so, daß ich unterwegs Texte in meinen „handheld"-electronic-printer eingebe und zu Hause über das Kommunikationsprogramm Access II von dessen Speicher auf Diskette (des Apple) überspiele. Da ich zur Weiterverarbeitung ein Pascal-Programm benutze (TEDI von ADIMENS), weil es entsprechend gute Programme in ProDOS noch nicht gibt, besteht nun die Notwendigkeit, die unter ProDOS gespeicherte ASCII-Textdatei in eine Pascal-Datei umzuwandeln, was ich dank des Peeker-Wettbewerbs nun problemlos kann.
Für den Ausdruck der fertig bearbeiteten Textdatei kann ich leider nicht das Pascal-Programm benutzen (dort fehlt die Möglichkeit, Einzelblattstopps automatisch einzufügen), sondern muß das Ganze wieder zurücktransferieren, um dann mit Appleworks unter ProDOS auszudrucken. Eine aufwendige Sache...
Zur Vereinfachung gäbe es m.E. nun zwei Möglichkeiten:
a) Die Überspielung von Handheld müßte schon unter Pascal erfolgen, so daß wenigstens eine Transferierung entfiele. Meine eigenen Versuche, (als Anfänger) ein Pascal-Kommunikationsprogramm zu schreiben, scheiterten an der Abspeicherungsmöglichkeit der über REMIN problemlos empfangenen Daten unter einer bestimmten Datei auf Diskette. Ich suche deshalb jetzt ein schon bestehendes Pascal-Programm für diesen Zweck.
b) Die Verwendung von Kyan-Pascal würde mir die Möglichkeit geben (?), auf ProDOS-Dateien auch unter Pascal zuzugreifen. Voraussetzung dafür wäre aber, daß mein (Pascal 1.2)-Textprogramm auch unter Kyan-Pascal laufen könnte. Dies ist nach Auskunft von Herrn Stiehl nicht der Fall, da ich von diesem Programm selbstverständlich keine Source-Files besitze, die neu compiliert werden könnten. Meine Frage: Werden Sie aufgrund Ihrer Fähigkeit, beliebige Codefiles zurückzuübersetzen, ein entsprechendes Programm veröffentlichen, das dies tun kann, oder wären Sie ggf. in meinem Einzelfall bereit, dies für mich zu tun, wenn ich dann bei den Textfiles im ProDOS-Format bleiben kann?
Simon Dach, Blaubeuren

Antwort:
Ein bestehendes Pascal-Kommunikationsprogramm besitze ich leider nicht, so daß ich Ihnen hier keine Empfehlung geben kann. Wenn Sie allerdings schon Daten über REMIN problemlos empfangen, so speichern Sie diese doch in einen großen Array, sagen wir mal mit dem Namen A, und speichern ihn dann folgendermaßen auf Diskette ab:

```
var F: file;
begin
rewrite (F, 'TESTFILE');
if blockwrite (F, A, Blks)
<> Blks
then exit (program);
close (F, lock)
end;
```

Dabei enthält Blks die Anzahl der abzuspeichernden Blöcke, die die Anzahl der abzuspeichernden Bytes div 512 ist, und falls Bytes mod 512 nicht Null ist, muß noch ein Block dazuaddiert werden.
Das Zurückübersetzen des Pascal 1.2-Textprogrammes wäre ein Aufwand, dem ich mich nicht unterziehen möchte, weil er Monate oder gar Jahre dauern würde (je nach Größe des TEDI). Ein Programm, welches dies automatisch macht, ist sehr kompliziert und würde eine Länge von 10000 bis 20000 Zeilen Pascal haben (s. auch Antwort zum Leserbrief von Gerhard Bletzer). Außerdem würde es in Ihrem Fall keine Abhilfe schaffen, weil UCSD-Pascal Textfiles nicht mit dem Kyan-Pascal Compiler compiliert werden können, wenn darin Befehle vorkommen, die es in Kyan-Pascal nicht gibt. Gehen Sie davon aus, daß im TEDI tausende solcher Befehle vorkommen.

Brief von Edgar Fuss in 10/85
Die Richtigstellung der Abfrage für die Gültigkeit eines Pascal-Directory in meinem CopyDupDir-Programm ist nicht korrekt. Wenn man einmal von der fehlenden Klammer vor „dFKind" absieht, hat Herr Fuss wohl nicht beachtet, daß das CopyDupDir-Programm (s.Heft 1/85) kein System-, sondern ein Userprogramm ist. Das bedeutet, daß es keinen Zugriff auf eine Variable MiscInfo.UserKind geben kann; diese Variable existiert im Pascalsystem (äußerste Ebene). Insofern würde eine Abfrage, wie sie Herr Fuss vorschlägt, in meinem Programm beim Übersetzen lediglich einen Syntaxfehler 104 (Undeclared Identifier) hervorrufen, da MiscInfo nirgends definiert wird. Auch das Einlesen des Files SYSTEM.MISCINFO würde nichts bringen, da der UserKind geändert worden sein könnte. Außerdem soll mein Programm ein Directory reparieren. Dabei spielt es keine Rolle, in welchem UserKind man sich befindet. Im übrigen ist es, wie in meinem Artikel dargelegt, praktisch unmöglich, einen anderen UserKind einzustellen, so daß eine Abfrage wie in meinem Programm durchaus genügt.

Diverse Leseranfragen

PAL-Karte
Der Apple IIe bietet eine eingebaute PAL-Karte. An den Übergängen zweier Farben flimmern die Farben. Gibt es eine Möglichkeit, diesen Effekt auszuschalten? Dieser Effekt tritt sowohl bei einem Fernseher als auch bei einem Farbmonitor auf.
Martin Dobbert, Berlin
Antwort: Beachten Sie bitte den technischen Erfahrungsbericht zu einer PAL-Karte im Heft 6/86. us

Siemens-PT88-Drucker
Gibt es den „Superdump" (Peeker 6/85, S. 22) auch für den Siemens Drucker PT88, bzw. wer kann mir evtl. weiterhelfen?
Dieter Jörg, Gerlingen
Antwort: Zumindest in der Apple-Gemeinde scheint den PT88 keiner zu benutzen. Oder doch? us

LOAD
Ich suche ein Maschinenprogramm, welches den LOAD-Befehl als RUN-Befehl ausführt (Apple IIc).
Horst Figgener, Kamen-Methler
Antwort: LOAD/RUN sind sowohl Applesoft-Interpreter- als auch DOS/ProDOS-Befehle. Im ersteren Fall müßte man das Interpreter-ROM ändern, was Sie wahrscheinlich nicht wollen. Im letzteren Fall müßte man die Befehlsworttabelle im RAM ändern. Für DOS 3.3 gibt es z.B. das Programm DOS-BOS, mit dessen Hilfe diese Modifikation menügesteuert vorgenommen werden kann. Die 48K-DOS-3.3-Befehlsnamen befinden sich ab $A884 im RAM. Im übrigen führt nach POKE 214, 128 *jeder* Direktbefehl zum Zwangs-RUN (gilt für Applesoft und DOS 3.3 gleichermaßen). us

Apple-II-Nachfolger
In Peeker 2/86 schlägt Stefan Niedergesäs einen Ideenwettbewerb für den neuen Apple IIe-Nachfolger vor, worin Peeker-Leser Vorschläge machen sollten, wie der Nachfolger, basierend auf einem 16-Bit-Prozessor (wahrscheinlich dem 65816), ihrer Meinung nach von der Firma Apple gestaltet werden solle, insbesondere im Hinblick auf den Atari 520 ST und die Commodore Amiga. Gemäß meinem Erfahrungen stellen diese beiden Computer keine Konkurrenz für den Apple dar, denn deren Kundenkreis ist ein anderer. Viel bedrohlicher dagegen ist das andere Steinzeitprodukt namens IBM-PC. Dieser erfreut sich nämlich gerade auf Kosten „Old Apple's" besonders großer Beliebtheit und Verbreitung in genau den Kundenkreisen, die auch von Apple anvisiert werden. Ich persönlich verstehe dies nicht, denn meine Uni-Erfahrungen an IBM-PCs sind weitaus weniger positiv als an meinem Apple IIe, wobei ich zugeben muß, daß auch mich einst das IBM-Fieber gepackt hatte (bevor ich auf IBM-PCs programmierte). Auch laufen meine Applesoft-Programme auf meinem knapp 1MHz lahmen 8-Bit-Rechner mit seinem sehr beschränkten Befehlssatz (prozessorseitig) zum Teil wesentlich schneller als adäquate Basic-Programme auf der 16-Bit-Maschine von IBM mit über 4MHz und viel mächtigeren Befehlen. Außerdem stürzt der IBM-Rechner relativ häufig ab, etwas, das mir in Applesoft-Basic noch nie passiert ist! Dennoch erfreut sich dieses Gerät größter und immer größer werdender Beliebtheit. An fast jeder Ecke hängt Charlie und streckt einem irgendein IBM-Produkt entgegen. Auch die Software-Industrie macht fast nur mit den auf dem IBM-PC lauffähigen Programmen Reklame und wirbt mit der IBM-Kompatibilität für Ihre Produkte. Daß diese zum Teil auch in einer Apple-Version erhältlich sind, wird verschwiegen. Selbst wenn dieses Informationsdefizit seitens der Firma Apple nun endlich beseitigt würde, so ließe sich damit zwar vielleicht die Abwanderung von Apple-Besitzern in das IBM-Lager verringern, aber sicher nicht umgekehrt eine Abwerbung von IBM-Kunden oder potentiellen IBM-Kunden erreichen. Dazu ist etwas wesentlich Neues, etwas wesentlich Besseres nötig als das, was IBM zur Zeit zu bieten hat. So etwas könnte z.B. ein 128-Bit-Arithmetikprozessor mit in Nibble-Schritten variabler Daten- und Adreßbreite sein. Ob dieses Variieren der Daten- und Adreßbreite wie beim 65816 über ein zusätzliches Flag gelöst wird oder einfach durch einen anderen Befehlscode (beim 6502 wurde diese Möglichkeit für die Zero-Page-Adressen gewählt), ist im Prinzip egal. Die Vorteile der variablen Datenbreite machen sich sehr deutlich bei Grafik mit flexiblen Shapes bemerkbar. Auch bei vielen Rechenoperationen kann dies nützlich sein. Als Arithmetik-Funktionen wären zusätzlich zu den 6502-Fähigkeiten denkbar: ADD (inklusive CLC), SUB (inklusive SEC), MULT, DIV, SERI (Entwicklung einer Potenzreihe), was für viele transzendente Funktionen benötigt wird, und natürlich so manches mehr, was beim 6502 zu kurz kommt. Da die Entwicklung eines solchen Prozessors ein paar Jahre Zeit kostet, halte ich es für das Beste, wenn man auf der Basis des 65816-Prozessors eine 16 Megabyte-Maschine entwickelt, auf welcher bereits ein freier Sokkel für den späteren 128-Bit-Prozessor installiert ist.
(Wenn ich das Handbuch meines Merlin-Pro richtig deute, hat der 65816 eine variable Adreßbreite von 16 bzw. 24 Bit und eine variable Datenbreite von 8 bzw. 16 Bit; der 65802 hat ebenfalls eine variable Datenbreite von 8 bzw. 16 Bit, aber nur eine Adreßbreite von 16 Bit, weshalb es bei ihm möglich ist, ihn pinkompatibel zum 6502 zu gestalten. Beide haben im Modus für 16-Bit-Adressen auch noch die Möglichkeit der 8-Bit-Zeropage-Adressierung. Sobald aber auf 16 Bit Datenbreite – und beim 65816 damit auch auf 24 Bit Adreßbreite – umgeschaltet wird, entfällt die Möglichkeit der Zeropage-Adressierung über eine 8-Bit-Kurzadresse.)
Dieser Computer könnte dann wahlweise ausgeliefert werden mit Tastatur oder einem Kabel mit Interface für den Direktanschluß des alten Apple II. Dies ist meines Erachtens notwendig, damit man die Stammkunden nicht für ihre Treue vor den Kopf stößt und sagt: „Werft euren alten Apple weg, kauft den neuen." Auf diese Weise könnte man sich vorübergehend Luft verschaffen, um in Ruhe den neuen 128-Bit-Prozessor zu entwickeln, um dann wieder konkurrenzfähig zu sein, besonders im Hinblick auf den Riesen IBM. Es wäre jedenfalls sehr schade, wenn IBM den ganzen Markt an sich reißen könnte. Innerhalb welcher Zeit Apple jedoch in der Lage wäre, Ideen von Peeker-Lesern aufzugreifen und gegebenenfalls gar zu verwirklichen, entzieht sich meiner Kenntnis. Ich weiß auch nicht, ob ein solches Maß an Flexibilität für eine Computer-Firma überhaupt möglich ist. Ich jedenfalls bemühe mich schon seit einer Weile um einen 65802-Prozessor, habe jedoch noch keinen gesehen und auch noch keine Bezugsquelle gefunden. Sollte diesbezüglich jemand mehr wissen als ich, so kann er dies ja der Redaktion oder mir mitteilen.
Georg Maaß, Karlsruhe
Antwort: Der Apple-II-Nachfolger, der wahrscheinlich noch im vierten Quartal 1986 erscheinen wird, wird den 65816-Prozessor haben und insofern im Emulationsmodus zumindest weitgehend mit dem alten Apple II softwarekompatibel sein. Während der Prozessor bereits öffentlich (in Deutschland anläßlich der Mac-Plus-Pressekonferenz Anfang 1986) bestätigt wurde, dürfen wir andere technische Details noch nicht bekanntgeben. Anrufen zwecklos!
Für den IIe und IIc gibt es inzwischen von der Firma Checkmate 65816-Prozessorkarten. Darüber hinaus kann man den 65802 auch selbst in den Apple IIe einbauen. Wenden Sie sich an die Firma Unitronic in Düsseldorf, die den 65816 bzw. 65802 als Distributor von Western Design, Arizona (USA) vertreibt. Ein Erfahrungsbericht eines „Selbsteinbauers" erscheint im Peeker. us

HP-Floppy und -Drucker
Hiermit möchte ich höflichst anfragen, ob Ihnen irgendeine Möglichkeit bekannt ist, von einem Apple IIe aus die Doppelfloppy HP-9121D und den Drucker HP-82906A anzusprechen, die über eine HP-IB-Schnittstelle verfügen. Es handelt sich hierbei intern um 3.5"-SONY-Laufwerke bzw. einen Epson FX-80 mit geändertem Zeichensatz. Die Geräte sollten auch weiterhin an einem HP-86B eingesetzt werden können. Außerdem wäre ich für die Nennung empfehlenswerter Apple IIe-Kopien dankbar, die vor allem eine größtmögliche Software-Kompatibilität und eine gute Tastatur aufweisen sollten.
Thomas Fütterer, Triberg

TA 80008
Ich suche schon seit längerer Zeit vergeblich nach einer Apple-Interface-Schnittstelle zu einer TA 80008 Schreibmaschine. Könnten Sie mir in dieser Angelegenheit mit Tips bzw. Adressen oder mit Hinweisen auf Veröffentlichungen helfen. Für Ihre Bemühungen sage ich Ihnen jetzt schon herzlichen Dank.
Reiner Sokolowski, Flensburg

Kann irgendein Peeker-Leser helfen? us

Korrekturen und Errata zu früheren Peeker-Heften

Designer, 1/86
Im Listing auf S. 44 wurden zwei Textblöcke wegen eines Montagefehlers vertauscht. Siehe Korrekturhinweis in Heft 2/86, S. 65. Die Sammeldiskette ist korrekt. us

Quicksort, 1/86
In der Tabelle auf S. 8 ist in der 39. Zeile, 16. Spalte statt „83" nur „3" abgedruckt; Belichtungsfehler. us

AGE, 1/86
Auf S. 35 müssen die letzten 7 Zeilen (ab $9340...) an das Ende der mittleren Spalte, also ab $9338..., angehängt werden. Montagefehler. Die Sammeldiskette ist korrekt.

DMP-Charger, 9/85
Daß Sie Testberichte haben, finde ich gut. Zum Testbericht von Harald Grumser im September-Heft möchte ich aber Stellung nehmen. Ich habe mir den DMP-Charger gekauft. Appleworks hat ein scheußliches ß und Ü. Der Bericht ist mir nicht kritisch genug! Beim längeren Umgang mit dem Charger merkt man einige Schwachpunkte. So gibt es keine Löschmöglichkeit eines einzelnen Zeichens, wenn alle Tasten umdefiniert sind. Bei der Zeichensatz-Erstellung wird nicht angezeigt, welche Taste mit welchem Zeichen belegt ist. Besser und praktischer wäre da eine ausdruckbare Tastaturschablone sowie eine Anzeige beim Editieren. So behält man doch Überblick. Oder wer mal versucht hat, zwei Zeichensätze zu mischen, erlebt eine Ochsentour. Da fehlt die Möglichkeit für Sammeleingaben der zu mischenden Zeichen. Auch das im Artikel gebrachte Beispiel hätte erläutert werden sollen. Es handelt sich dabei um einem Mac-like-Zeichensatz. Zu diesem wäre eine nicht im Handbuch dargestellte Macke erwähnenswert. Bevor man zwei Zeilen „zusammenklebt", muß erst eine Leerzeile bestehen. Ansonsten gibt es eine sichtbare Naht zwischen den Zeilen. Da hat der Druckertreiber von Appleworks nach Auskunft von Herrn Rautenstrauch, der das Programm geschrieben hat, eine Macke. Das hätte Herr Grumser doch schon beim Probieren merken müssen!

Dazzle Draw, 9/85
Ich möchte hier gerne einmal zu einem Leserbrief Stellung nehmen, der in dem letzten Peeker (2/86, S. 64) abgedruckt war. Dort schrieb ein Leser, daß das Programm Dazzle Draw nicht fähig sei, die 16 Farben der Double-Hires Grafik auf seinem Taxan-Farbmonitor darzustellen. Da ich selbst viel mit Dazzle Draw arbeite und auf meinem Taxan Super-Vision III seltsamerweise alle 16 Farben einwandfrei zu sehen sind, liegt der Fehler wohl nicht am Programm. Der angesprochene „Grün-Violett-Effekt" tritt auf, wenn man die Double-Hires Grafik einschaltet, ohne die 80-Zeichen-Karte aktiviert zu haben. Ich kann hier natürlich keine Ferndiagnose stellen, aber muß sagen, daß es vielleicht angebracht wäre, das Programm einmal auf anderen Apple's zu testen.
Holger Göritz, Grünendeich

Trickfilmgrafik, 11/85
Bei der Überprüfung meines Programms habe ich folgenden Fehler gefunden: Beim Erhöhen des Zeigers auf das Bitsprite (in der Routine SZEICH, Label MOD) habe ich vergessen, den Überlauf zu berücksichtigen. Der Fehler ist einfach durch Einfügen von zwei Befehlen zu beheben (s. Heft 11/85, S. 12, 2. Spalte, letzte Zeile):

```
248 SZEICH4 INX
            BNE SZEICH5
            INC MOD + 2
249         INY
250 SZEICH5 DEC XCOUNT
```

Dadurch wird das Programm um 5 Bytes länger, und es verschiebt sich die Startadresse von POSINIT (nun $85C0). Dies ist bei Demo-Programmen entsprechend zu berücksichtigen.
Bernd Klawonn, Bochum

EPROM-Gerät, 10/85
Ich habe mich sehr darüber gefreut, daß Sie jetzt auch Hardware-Schaltungen veröffentlichen. Da ich schon länger eine Anleitung für einen Eprommer suche, habe ich mich entschlossen, dieses Gerät zu bauen.
Beim näheren Betrachten der Schaltung sind jedoch noch einige Fragen aufgetaucht. Meine erste Frage bezieht sich auf das CS1 Signal des VIA 6522. Aus dem Schaltbild ist nicht zu entnehmen, womit er verbunden werden soll. Gehe ich recht in der Annahme, daß dieses Signal an +5V gehört? Die zweite Frage bezieht sich auf das Netzteil. Es ist leider nicht angegeben, wie der Netztrafo dimensioniert sein muß. Nach meinen Informationen muß die Sekundärspannung in diesem Fall zwischen 27 und 30 Volt liegen. Nun müßte man nur noch die Stromaufnahme wissen. Unklar ist auch die Handhabung der unterschiedlichen Epromtypen. Wenn ich die Abb. 1 (Heft 10/85, S. 41) richtig verstanden habe, beziehen sich beim Programmiersockel die Angaben in Klammern auf eine 28-polige Fassung und die ohne Klammern auf eine 24-polige. Die kleinere Fassung eignet sich nur für die Epromtypen 2716 und 2732. Mit der größeren müßten dann auch die Typen 2764 und 27128 bearbeitet werden können. In der 28-poligen Fassung müßten dann die 16- und 32iger Typen so eingesetzt werden, daß die 4 freien Plätze der Fassung auf der Seite der Einkerbung des Eproms liegen.
Im übrigen vermisse ich die Anschlußbelegungen der ICs CD4050 und SN7404. Ich habe in meinen Unterlagen die Anschlußbelegungen nachgesehen:
4050: +5V an Pin 1, Masse an Pin 8. Weitere Pinbelegung der Treiber mit Reihenfolge Eingang/Ausgang: 3/2, 5/4, 7/6, 9/10, 11/12, 14/15.
7404: +5V an Pin 14, Masse an Pin 7. Weitere Pinbelegung der Inverter mit Reihenfolge Eingang/Ausgang: 1/2, 3/4, 5/6, 9/8, 11/10, 13/12.
Der Ausgang eines Treibers/Inverters befindet sich am spitzen Ende des Schaltsymbols.
Ich hoffe, daß ich nicht der einzige bin, der zu der Schaltung Fragen und Ergänzungen hat, so daß diese Punkte möglichst bald klargestellt werden können.
Trotz der Unklarheiten finde ich die Schaltung gut, da sie eine Alternative zu einem teuren Eprommer bietet.
Mark Liebrand, Bocholt

Antwort von Dr. Schulé: Herr Liebrand hat recht: In die Abbildung des EPROM-Programmiergeräts haben sich einige kleine Unterlassungen hineingemogelt: CS1 ist mit +5V verbunden, die Pin-Numerierung bezieht sich je nach Wunsch auf einen 24-poligen (Pin-Nummern nicht eingeklammert) oder auf einen 28-poligen Sockel (Pin-Nummern in Klammern). Auch die Inverter und Treiber kann man sich nach der Liste der Anschlüsse der IC aussuchen. Den Trafo wählt man nach der Verfügbarkeit, wobei ein gebräuchlicher Wert z.B. 2x15V, 3,5VA ist. Die beiden Sekundärwindungen werden dann in Serie geschaltet. Wir hoffen, daß mit den Ergänzungen von Herrn Liebrand der Nachbau auf keine Schwierigkeiten stößt.

READPAS, 1/86
Bei UCSD-Pascal wird die aktuelle Anzahl Files über einen Zähler im Directory erkannt. Ungültige File-Einträge werden physikalisch nicht gelöscht, sondern nur nicht mehr berücksichtigt. Der Filetyp beansprucht nur 4 Bit Platz im Directory und nicht ein ganzes Byte. Die zwei genannten Fehler sind in untenstehender Prozedur GETFILES beseitigt (ersetzt die veröffentlichte Prozedur).

```
PROCEDURE GETFILES;
TYPE
UCSDDIR = ARRAY [0..77]
OF RECORD CASE INTEGER OF
0:(VSTB,VNEB,VTYP:IN-
TEGER;
VVNA:STRING[7];
VKBN,VFNU,VTIM,VDAT:IN-
TEGER);
1:(DSTB,DENB,DTYP:IN-
TEGER;
DFNA:STRING[15];
DLEN,DDAT:INTEGER);
END;
VAR
I:INTEGER; DIRECTORY:
UCSDDIR ABSOLUTE DIRBLOCK;
BEGIN
FILEZAHL:=DIRECTO-
RY[0].VFNU;
{Anzahl gültiger Files}
FOR I:=1 TO FILEZAHL DO
WITH  DIR[I],DIRECTORY[I]
DO
BEGIN
VON:=DSTB; BIS:=DENB;
EOFBYTE:=DLEN-1;
FTYP:='DATA'
{Data,Bad,Info,Graf,Fo-
to,Dir}
CASE (DTYP AND 15) OF
{Nur 4 Bit sind massgebend;
TURBO}
2:FTYP:='CODE';
{'packt' Bit-Arrays nicht
gleich}
3:FTYP:='TEXT';
{wie UCSD Pascal}
END;
FNAME:=DFNA;
END;
END;
```

Bernard Condrau, CH-Dübendorf

DOS-Mover, 2/86
In dem obengenannten Artikel weisen Sie darauf hin, daß weder FID noch RENUMBER unter gemovtem DOS laufen. Durch kleine Änderungen werden die Programme auch unter gemovtem DOS lauffähig:
Änderungen für FID:
BLOAD FID
CALL-151
0854: 90

BSAVE FID, A$803, L$124F
Änderung für RENUMBER:
LOAD RENUMBER
CALL-151
0B7C: 69 80 30 09 C9 08 10 05
109C: 4C 6C D6
SAVE RENUMBER
Achtung: Die Adressen $0B7C und $109C gelten für RENUMBER von der System-Master 1982. Für die ältere Version von 1980 sind diese durch $0B63 und $1083 zu ersetzen.
Harald Nitsche, Gummersbach

Makros für 65C02, 4/85

Vielen Dank für den Artikel, in dem sich m.E. einige Fehler befinden (Heft 4/85, S. 31 ff.). Alte Zeile durch folgende ersetzen:

```
97 DS 2
117 DS 2
154 * BRA
299 DS-2
303 RMB1 EQU Ü1*$10.$07
327 DS-2
331 SMB1 EQU 8+Ü1*$10.$07
```

Ansonsten ein Lob für Inhalt und Aufmachung des Peeker.
Jürgen Treumer, München

Frühere Korrekturen

Turtle Graphic 1/84/19, Ergänzung 2/84/10
ProDOS-Patch für F8-ROMs 1/84/54, Ergänzung 2/84/10
Poor Man's RAM-Disk 1-2/85/8, Ergänzung 6/85/69
Garbage-Collection 1-2/85/32, Berichtigung 8/85/70
Pascal 1.2 3/85/65, Ergänzung 10/85/60
Terminal-Programm 4/85/34, Ergänzung 6/85/72
INALL 4/85/70, Berichtigung 5/85/50
PLOT 2.0 5/85/17, Berichtigung 9/85/69
Superdump 6/85/22, Ergänzung 8/85/50
Diversi-DOS 2C 6/85/72, Ergänzung 10/85/69
Graf-quattro-Cursoren 6/85/6, Ergänzung 10/85/63
Fourier-Analyse 6/85/34, Berichtigung 9/85/69
Pascal-RAM-Disk 6/85/48, Berichtigung 7/85/52
Format-Programm 7/85/20, Ergänzung 9/85/69
Adreßverwaltung in dBase 8/85/40, Ergänzung 10/85/21

Cirtech-CP/M-Karte, 4/86

Wir haben das Programm „SUPERQUICK" ausführlich getestet und sind zu dem Ergebnis gekommen, daß es hierbei um ein Software-Problem des Programmes „SUPERQUICK" handelt und nicht um ein Problem unserer CP/M-PLUS-Hardware. Bitte entnehmen Sie den folgenden Punkten weitere Einzelheiten:

1. Alle kommerziellen Apple-//e,c-Programme laufen auf dem Apple // e,c mit dem eingebauten CIRTECH-CP/M-PLUS-SYSTEM fehlerfrei, sofern diese richtig installiert werden. Dies bezieht sich auf alle Apple-//e,c-Programme, so z.B. WORDSTAR, TURBO PASCAL, SUPERQUICK, LOCKSMITH 5.0, APPLEWORKS, VISICALC etc.
2. Bei einigen Programmen müssen vor dem Ersteinsatz verschiedene Parameter auf die speziellen Bedürfnisse des Apple //e,c angepaßt werden. Dies trifft auch auf das Programm „SUPERQUICK" zu.
3. Das Programm „SUPERQUICK" läuft fehlerfrei – Kaltstart sowie Warmstart – sofern die Suchroutine nach RAM-Erweiterungskarten ausgeschaltet ist. Dies ist notwendig, da das Programm „SUPERQUICK" beim Suchen nach weiteren RAM-Karten (IBS AP 17) in den ROM-Speicherbereich $C400-$C7FF schreibt und dadurch die CIRTECH-Z80-Karte einschaltete. Dies führt zum Abstürzen des Programmes.
4. Das bekannte Kopierprogramm LOCKSMITH 5,0, welches ebenfalls den Apple //e,c nach RAM-Erweiterungskarten absucht, bewältigt dieses Problem ohne in den ROM-Speicherbereich $C400-$C7FF zu schreiben. Dadurch treten keinerlei Probleme bei dem Einsatz von dem Programm LOCKSMITH 5.0 auf einem Apple //e,c mit eingebautem CIRTECH-CP/M-PLUS-SYSTEM auf.
5. Wir empfehlen zur Beseitigung des Problemes, das Programm „SUPERQUICK" in einer Version auszuliefern, bei der die Suchroutine nach RAM-Erweiterungskarten ausgeschaltet ist und des weiteren den End-User detaillierter auf mögliche Probleme in dem Kapitel 4 des SUPERQUICK-Handbuches hinzuweisen.

Bei der von uns angegebenen Taktfrequenz von ca. 4 bis 5,7 MHz handelt es sich um einen statistischen Mittelwert und nicht um einen Absolutwert!
Das von uns gelieferte Handbuch ist in erster Linie für den kommerziellen Anwender gedacht und enthält für diesen alle notwendigen Informationen zum Einsatz kommerzieller CP/M-Programme. Für den „Tüftler" bieten wir das von Ihnen erwähnte CP/M-PLUS-Programmier-Paket an.
M. Semjan, Frankfurt

Vokabeltrainer, 1/86

In dem Applesoft-Modul VOK. TRAINER (Sammeldisk #13) sind folgende Zeilen zu berichtigen:

```
1790 IF C$ <> "G" THEN GOTO
1770
(statt IF C$ = <> "G" ...)
2355 IF F$ = "N"
     OR F1$ = "N" THEN GOTO
     2020
(statt nur IF F$ = "N" THEN ...)
```

Computerbücher
die gehen, für Computer die kommen.

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

Arne Schäpers
Bewegte Apple-Grafik
1985, 305 S., 6 Abb., kart.,
DM 58,—
ISBN 3-7785-1150-5

Frank Bühler
Applesoft Basic
Tips und Tricks
1985, 241 S., 40 Abb., kart.,
DM 38,—
ISBN 3-7785-1094-0

Jürgen Kehrel
Assembler lernen
Band 1: Einführung in die Assembler-Programmierung des 6502
1985, 235 S., kart.,
DM 38,–
ISBN 3-7785-1151-3

Ulrich Stiehl
Apple Assembler
1984, 200 S., 3 Abb., kart.,
DM 34,—
ISBN 3-7785-1047-9

Ulrich Stiehl
Apple DOS 3.3
Tips und Tricks
3., völlig überarb. Aufl. 1986
X, 203 S., kart.,
DM 28,–
ISBN 3-7785-1298-6

Ulrich Stiehl
ProDOS für Aufsteiger
Band 1
2., geänderte Auflage 1985,
208 S., kart., DM 28,—
ISBN 3-7785-1098-3

Ulrich Stiehl
ProDOS für Aufsteiger
Band 2
1985, 207 S., kart., DM 30,—
ISBN 3-7785-1036-3

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

Hüthig

Flipper--Karte
Die Flipper-Karte dient der Speichererweiterung des Apple II. Damit kann man den Apple II+/e zu einem Computer mit bis zu 6M RAM Speicher ausbauen. Die Karte wird einfach in den Apple II+/e gesteckt und ist sofort einsatzfähig. Sie ist kompatibel zu DOS 3.3, CP/M, Pascal und ProDOS sowie auch Appleworks.
Die wichtigsten Eigenschaften der Flipper-Karte:
– Die Karte wird mit 1M RAM geliefert.
– Auf max. 6M RAM erweiterbar.
– Kein „Patchen" notwendig.
– Wird automatisch vom Cirtech-CP/M-Plus-System aktiviert.
– Einsatz unter verschiedenen Betriebssystemen möglich, z.B. CP/M, Pascal und ProDOS.
– Sehr schneller Datenzugriff.
– In jedem Slot einsatzfähig.
– Eingebauter Fehler-Selbsttest.
Bezugsquelle: M. Semjan Computer Systeme, Frankfurt

Europrint FT 80 X
ist ein leistungsfähiger Nadeldrukker mit folgenden Merkmalen:
– Epson-, IBM-, Schneider- oder C64-Zeichensätze und Steuercodes per DIP-Schalter einstellbar
– Near Letter Quality (NLQ)
– Prüfzertifikation gemäß GS und FTZ
– 100 Zeichen/s.
– niedriger Geräuschpegel
– robuste Ausführung für harten Dauereinsatz
– viele Anschlußsätze wie C64, Apple, IBM, CPC 464 etc.
Der Grundpreis für Europrint ohne Interface beträgt DM 695,-. Die Interface-Karten sind ab DM 95,- lieferbar.
Europrint wird mit reichhaltigem Zubehör geliefert:
– Farbbandrollenaufnahme
– Einzelblattschacht
– Staubschutzhaube
– Papierrollenhalter
Bezugsquelle: Unitronic GmbH, Düsseldorf

Jazz 1A für Mac Plus
Die neue Jazz-Version 1A unterstützt folgende Erweiterungen des Macintosh Plus:
– Installation auf der Festplatte, die eine Programm-Diskette im Laufwerk überflüssig macht.
– Schnellere Zugriffszeiten (indem das interne doppelseitige Diskettenlaufwerk ausgenutzt wird, paßt das Betriebssystem und das komplette Jazz-Programm auf eine Diskette).
– 1 M RAM, das dem Anwender erlaubt, größere Jazz-Dokumente anzulegen.
– Unterstützung von neuer Mac-Plus-Tastatur.
– Schnelleres Ausdrucken über den LaserWriter.
– Unterstützung der hierarchischen Dateien-Struktur.
Bezugsquelle: Lotus, München

Champion-Drucker-Karte
Das neue Drucker-Interface Champion von Cirtech mit 16K RAM bzw. 64K RAM Puffer für den Apple II+/e wird in Deutschland über die Firma Semjan vertrieben. Es druckt sowohl die Textseiten (40/80 Zeichen) als auch die Grafikseiten aus und ist mit den meisten Druckern mit Centronics-Parallel-Schnittstelle kompatibel.

Eine serielle Ausgabe und Eingabe ist mit dem seriellen Adapter möglich und verwandelt die Karte in eine vollwertige parallele und serielle Schnittstelle (RS 232C) für den Apple II+/e. Das Champion-Drucker-Interface ist voll DOS-, ProDOS-, Pascal-, CP/M- und Appleworks-kompatibel.

Bezugsquelle: M. Semjan Computer Systeme, Frankfurt

Grafiktablett für Mac
Das Mactablet von Summagraphics mit einer aktiven grafischen Fläche von 152 x 227 mm und einer Auflösung von 20 l/mm versorgt den Macintosh-Computer mit einem idealen Interface für grafische Anwendungen. Mit dem Mactablet erhalten Sie zusätzliche Möglichkeiten, wie das Übertragen und Ändern von schon vorhandenen Zeichungen oder das Entwerfen von eigener Grafik oder von Konstruktionsplänen. Die maximale Vorlagestärke beträgt 12 mm. Durch den universellen Aufbau der Treiber-Software ist das Mactablet kompatibel mit Macdraw, Macpaint und allen Software-Paketen für den Apple-Macintosh. Nachdem die Treiber-Software installiert ist, erscheint das Mactablet als Bestandteil im Desktop-Accessory. Sie können wählen zwischen Apple-Mouse und Mactablet. Beide Eingabesysteme können aktiviert sein und wahlweise je nach Applikation eingesetzt werden. Der attraktive Preis von DM 1.653,- schließt die Stromversorgung, die Treiber-Software und einen grafischen Stift ein, so daß Sie sofort mit Ihrer grafischen Arbeit beginnen können.
Bezugsquelle: Summagraphics Corporation, München

Fakturierprogramm Faktufix
Das Programm FAKTUFIX gibt dem Kaufmann laufend einen aktuellen Überblick über Kunden, Artikel und offene Posten (Rechnungen). Man kann
– 480 Kunden- bzw. Lieferantenadressen je Laufwerk und Diskette speichern,
– 720 Artikel je Laufwerk und Diskette erfassen,
– 480 offene Posten je Laufwerk und Diskette aufnehmen,
– Auftragsbestätigungen, Lieferscheine, Rechnungen, Gutschriften und Bestellungen drucken,
– die 6-stelligen Artikelnummern frei festlegen, wobei eine Doppelvergabe ausgeschlossen ist,
– Kundenlisten, Artikellisten und Rechnungslisten nach mehreren Kriterien sortiert ausdrucken,
– Zahlungseingänge mit Buchungssatz für BUFIX tätigen,
– die Eingabe durch Makro-Befehle erleichtern.

Dieses Programm zeichnet sich durch leichte Erlernbarkeit, anwenderfreundliche Eingabe mit Korrekturmöglichkeit und hohe Verarbeitungsgeschwindigkeit aus.

Bezugsquelle: Herr Rötering, Soest (keine eingetragene Firma)

Drucker-Treiber
Wohl jeder, der ein Computersystem zusammenstellt, steht vor dem Problem des Druckeranschlusses, wobei hier nicht die Hard-, sondern einmal die Softwareseite beleuchtet werden soll.
Angenommen, ein Anwender hat sich für ein Computersystem entschieden. Leistungsfähigkeit und Kompatibiltät wurden ebenso beachtet wie ein umfangreiches Software-Angebot. Natürlich sollen dann die mit dem Computer erarbeiteten Ergebnisse auch zu Papier gebracht werden.
Nehmen wir weiter an, die Entscheidung fällt auf einen Punktmatrixdrucker. Eigentlich logisch. Denn dieser Druckertyp erledigt vielfältigste Aufgaben in vernünftigem Kosten/Leistungsverhältnis. Apropos Preis: Angeblich preisgünstige Geräte gibt es wie Sand am Meer, aber der Freude an der gesparten Mark folgt schon bald bittere Enttäuschung:
– Die Druckgeschwindigkeit ist zu langsam.
– Korrespondenzqualität wird nicht erreicht.
– Geringe Auflösung bei Grafiken.
Kurzum: Der Drucker ist professionellen Ansprüchen nicht gewachsen. Denn die auf dem Bildschirm zu sehenden Texte und Grafiken sind einfach nicht qualitativ hochwertig auf Papier zu übertragen.
Der Grund für die Enttäuschung liegt klar auf der Hand: Drucker der ersten Generation mit 9-Nadel-Druckkopf sind den gestiegenen Ansprüchen niemals gewachsen. Ausschließlich Punktmatrixdrucker mit 18- bzw. 24-Nadel-Druckkopf erfüllen professionelle Anforderungen.
Aber gehen wir noch einmal einen Schritt weiter. Der Anwender ist jetzt von Druckern der 18- und 24-

Nadel-Kategorie überzeugt. Die technischen Daten sprechen für sich, und die Qualität des bei Einsatz von Programm A gemachten Ausdrucks ist vorbildlich. Der Ausdruck von Programm B ist jedoch, auch mit dem professionellen Drucker, kaum besser als mit dem „alten" 9-Nadel-Drucker. Wie ist so etwas möglich?

Nicht nur die Leistungsfähigkeit der Software und der Drucker ist ausschlaggebend. Es kommt noch ein dritter Faktor hinzu: Qualität und Leistungsspektrum des sogenannten „Treibers", der Verbindung zwischen Software und Drucker.

Der Treiber wandelt die Druckausgabedaten eines Programms in eine für den Drucker verständliche Reihe von Kommandos und Druckdaten um und bestimmt dadurch die Art und Form des Ausdrucks.

Dabei unterscheiden sich Drucker übrigens auch in der Kommandosprache. Zwar gibt es einige Standards, die aber bei weitem nicht ausreichen, die Möglichkeiten, die so mancher Drucker bietet, voll auszuschöpfen. Standards und Kompatibilität können demnach, wollte man einmal etwas böswillig schlußfolgern, eher hemmend als förderlich sein.

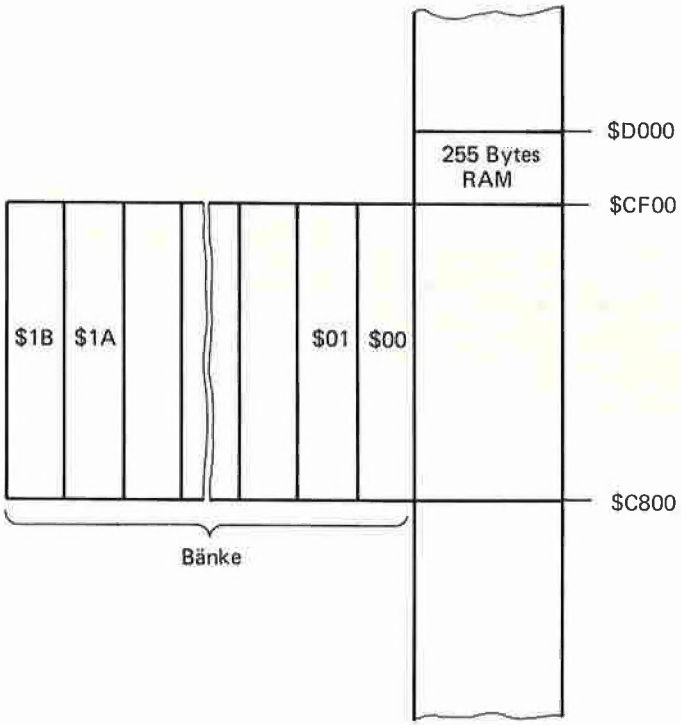
Dieses Problem ist prinzipiell allerdings recht einfach zu lösen. Jeder Drucker sollte seinen spezifischen Treiber haben. Aber dies ist wesentlich einfacher hingeschrieben als praktisch umgesetzt. Denn eine Vielzahl der momentan angebotenen Software – hauptsächlich jedoch die Programme älteren Datums – verfügt lediglich über einen standardisierten Treiber, der Drukker auf einem Niveau unterstützt, das vor 5 Jahren akzeptabel, heute jedoch völlig unzureichend ist. Erfreulicherweise zeichnet sich allerdings auch hier langsam aber sicher eine Wende ab. Denn die Software-Produzenten haben diesen Mißstand erkannt und bieten bei Ihren neuen Software-Paketen die Möglichkeit, einen dem jeweiligen Drucker entsprechenden Treiber in das Programm zu integrieren.

NEC hat vor allem in den letzten Monaten den Schwerpunkt auf die Entwicklung und Anpassung von Druckertreibern für NEC Spinwriter und Pinwriter gelegt.

Bezugsquelle: NEC Business Systems, München

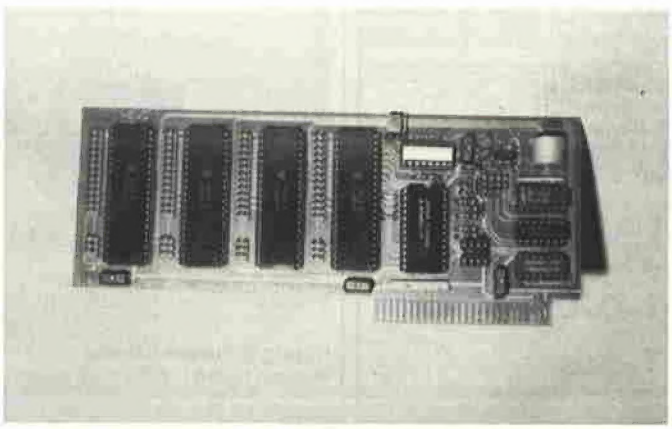
A/D-Wandlerkarte Softcard II

Das Aufbauprinzip besteht in der parallelen Anordnung von RAM bzw. EPROM im slotunabhängigen Teil des I/O-Bereichs ($C800-$CFFE). Dieser 2K große Adreßumfang wird von der Softcard II in zwei unabhängige Bereiche unterteilt: In den Speicher von $C800 bis $CEFF werden die diversen EPROM- bzw. RAM-Bänke je nach Wunsch des Benutzers eingeblendet. Die oberen 255 Bytes von $CF00-$CFFE stehen hingegen allen Bänken als gemeinsames RAM zur Verfügung. Durch dieses Konzept ist es möglich, einen kleinen gemeinsamen Datenbereich für alle Programmteile zu haben. Diese Aufteilung erlaubt es auch, Programme mit privaten Daten und großer Gesamtlänge (Parameterübergabe zwischen einzelnen Programmteilen auf den verschiedenen Speicherbänken) zu entwickeln. Die Anwendungsmöglichkeiten der Softcard II sind sehr vielseitig. Neben einer Entlastung des Apple-Hauptspeichers ermöglicht diese Zusatzkarte dem Programmierer, temporäre Variablen in gesonderten Bereichen unterzubringen. Die einfachste Einsatzmöglichkeit ist für den Assemblerprogrammierer gegeben. Er kann häufig benötigte Unterprogramme im EPROM ablegen und ist dadurch in der Lage, unterschiedliche Anwendungen ständig parallel zur Verfügung zu haben, ohne sie immer wieder neu von Diskette laden zu müssen. Der BASIC-Programmierer hat durch die Softcard II die Möglichkeit, nützliche Ampersand-Routinen dort abzulegen und so den Befehlssatz nach den eigenen Bedürfnissen beliebig zu erweitern. Als Option ist von WINTEX Instruments ein kompletter Ampersand-Handler verfügbar, so daß nur noch die eigentlichen Routinen vom Anwender erstellt werden müssen. Mit der variablen Bestückung der Softcard II ist eine bedarfsgerechte Anpassung der Hardware an den jeweiligen Einsatz möglich: Es sind sechs universell einsetzbare Stecksockel vorhanden, die sowohl EPROMs (Kapazitäten von 2K bis 8K) als auch RAMs (Kapazitäten von 2K und 8K) aufnehmen können. Zusätzlich ist je ein weiterer Sockel für den notwendigen RAM-Baustein (256 Bytes gemeinsames RAM für alle Bänke) und das slotabhängige EPROM vorhanden. Der Speicherumfang kann beliebig zwischen maximal 42K EPROM und rund 50K RAM variiert werden, wobei die unterschiedlichsten Kombinationen von RAM und EPROM auf der Softcard II zulässig sind. Die Minimalausstattung besteht in der Bestückung mit einem RAM. Der variable Einsatz von RAM und/oder EPROM erlaubt es, bis zur entgültigen Fertigstellung der Routinen mit einem RAM zu arbeiten und dann nur einmal ein EPROM zu brennen.



PIA-Karte für Apple II

Die PIA-Karte dient zur Steuerung paralleler Prozesse durch einen Apple II+ oder IIe. Sie kann, je nach Ausbaustufe, z.B. zum Steuern einer Alarmanlage, zum Einschalten bestimmter Geräte zu bestimmten Zeitpunkten oder als einfache Uhr verwendet werden. Außerdem lassen sich fast alle parallelen Schnittstellen (IEEE, Centronics) simulieren und konvertieren. Die PIA-Karte ist mit bis zu 4 PIAs, einem RAM oder EPROM sowie einer Real-Time-Clock bestückt. Im einzelnen sind folgende Bauteile vorgesehen:



– PIA: 6821
– Uhr: RTC 58321
– Speicher: 6116 oder 2716 oder 2732 sowie kompatible Typen.

Bezugsquelle: Conrad Helmcke, Berlin

Gepard-Computer
Die Firma Gepard Computer ist ein deutsches High-Tech-Unternehmen. Ziel der Firma ist es, technologisch herausragende Microcomputersysteme für den industriellen, wissenschaftlichen und semi-professionellen Bereich zu entwikkeln. Der Gepard-Computer ist ein modulares System auf Europakarten in einem 19"-Tischgehäuse oder einem 19"-Einschub eingebaut. Der System-Bus weist 16 völlig gleichberechtigte Steckplätze mit 96poligen, vergoldeten Steckverbindungen auf. Das System basiert auf der schnellen 16-Bit-CPU 68000 (68020 in Vorbereitung). Es zeichnet sich durch große Modularität auf der Hard- und Software-Ebene aus. Anwenderspezifische Problemlösungen sind somit kostengünstig und schnell zu realisieren. Weiterhin ist ein preiswerter Einstieg in die 16-Bit-Technologie durch Gepard-Minimalpakete, z.B. für Apple-II- und Commodore-C64-Besitzer möglich. Die Software basiert auf der Sprache Modula-2 und ist ebenso modular aufgebaut wie die Hardware. Eine systematische Syntax ermöglicht das einfache Erlernen dieser zukunftsträchtigen Hochsprache. Zur Grundausstattung gehört folgende Software: Betriebssystem GDOS, Modula-2-Compiler, 2 Programm-Editoren, Debugging-Tool und Symbolischer 68000-Assembler. Einige professionelle Programme wird es in Kürze geben. Die klare, übersichtliche, deutsche Dokumentation (über 350 Seiten) erleichtert die Einarbeitung wesentlich. Ein „Update-Service" hält jeden Anwender auf dem aktuellen Stand. Außerdem bringt die Hauszeitschrift „68000 News" alle neuen und wichtigen Informationen an einen großen Kreis von Benutzern und Interessenten.
Bezugsquelle: Gepard Computer GmbH & Co. KG, Oldenburg

Der nächste Peeker
Heft 6/1986
erscheint am 6.6.1986

Preiswerte Begleitdisketten

Bd. 1: DM 28,–; Bd. 2: DM 28,–

DM 28,–

DM 28,– (Neue Diskette für 3. Aufl.)

Hüthig Software Service
Postfach 10 28 69 · 6900 Heidelberg

Apple ProDOS für Aufsteiger

Band 1, 2. Aufl., 203 S., DM 28,–
Band 2, 208 S., DM 30,–

von Ulrich Stiehl

Dr. Alfred Hüthig Verlag
Postf. 10 28 69 · 6900 Heidelberg

Apple Assembler
Tips und Tricks

von Ulrich Stiehl
2. Aufl., 226 S., 3 Abb., kart., DM 34,–
ISBN 3-7785-1047-9

Dr. Alfred Hüthig Verlag
Postf. 10 28 69 · 6900 Heidelberg

Band 1:
1. Einleitung
2. Was Sie benötigen
3. Starten von APPLE WORKS
4. Der Schreibtischmanager
5. Datenbank
6. Rechenblatt

A1 Anschluß der Festplatte ProFile
A2 APPLE II Easy Pieces Referenz
A3 Druckeranpassungen
A4 DOS 3.3 Konvertierungen
A5 APPLE WORKS Disketten sichern/kopieren
A6 Hilfsfunktionen nach Programmteilen

Band 2:
1. Einleitung
2. Was Sie benötigen
3. Starten von APPLE WORKS
4. Der Schreibtischmanager
5. Textbearbeitung
6. Datenfernübertragung

A1 ... A6 wie Band 1, dazu:
A7 Modemkabel für APPLE IIe, IIc
A8 SuperSerialCard: Einstellung
A9 ASCII-Textdateien aus anderen Dateiformaten für ACCESS II
A10 DATEX-P20 F Verzeichnis
A11 Deutsche/Englische Menübilder von ACCESS II

APPLE WORKS auf APPLE II, IIe, IIc:
verwandelt APPLE-II-Computer in einen Elektronischen Schreibtischmanager mit:
Texterstellung ... Edition, Briefarchiv, Ausdruck etc. Datenarchivierung ... Kontoführung, Buchhaltung etc. Formblattkalkulation ... Bilanzen, VisiCalc-Dateien etc. Datenfernübertragung ... Mailbox, Rechnerkopplung etc.

- ist ein erfolgreicheres Integrationspaket als LOTUS auf IBM PC!
- ist auf 1 MByte Speichererweiterungen Ihres APPLE II vorbereitet!
- erschließt Ihnen die Zukunftstechnik MAILBOX!
- ist ebenso einfach zu bedienen wie APPLE WRITER:
  Kein Befehlsstudium ... Einfachste Menüführung ... Sofortige Anwendbarkeit

te-wi's APPLE WORKS SYSTEMBÜCHER 1+2 zeigen Ihnen:
- Sämtliche APPLE WORKS Funktionen an Beispielen aus der Wirtschaft
- Das Wechseln zwischen Text/Rechenblatt/Datenarchiv/Dfü
- Umfassende Systeminformationen zu Dateikonvertierung, Druckeranpassung etc.

Von Botta/Lange/Zimmermann je 264 Seiten, Softcover, je DM 49,–

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weitere te-wi-Bücher

NEU

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

LOGO – Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den U.SA. Für die Computer APPLE II, C-64, IBM PC, ATARI bis 520 ST, TI-99 und Schneider CPCs.
Hochwertiges Textbuch für Logo-Kurse zu Hause und im Lehrbereich.
384 Seiten, A4, DM 59,–

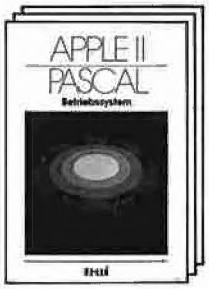
NEU

APPLE II – Bewegte 3D-Graphik
(Phil Cohen)
Selbstentworfene Graphiken und Diagramme – animiert oder als Standbilder – eben oder räumlich: alle erforderlichen BASIC-Programme mit Erklärung finden Sie in diesem Buch.
200 Seiten, Softcover, DM 49,–

Apple Maschinensprache
Für BASIC-Programmierer der einfachste Zugang zur Muttersprache des APPLE. Wesentlich schnellere Maschinenprogramme, direkte Manipulation des Mikroprozessors 6502 im APPLE – als Brücke dorthin benötigt dieses Buch nur die drei BASIC-Befehle POKE, CALL, PEEK: D. Inman/K. Inman. DM 49,–

NEU

Reparaturanleitung Computer: Apple II, IIplus
Einzigartige Serviceunterlage für Reparaturen und Entwicklungsarbeiten am Apple II. Enthält Schaltpläne, Bauteile- und Vergleichstypenliste; Prüfpunkte mit Oszillogrammen der Signalformen, Logiktabellen, Spannungsangaben; schnelle Servicetests; Anleitung zur systematischen Fehlersuche.
In A4-Mappe, DM 29,80



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

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

tm 4422

Noch im Programm:
Computer für Kinder, APPLE II, DM 29,80
6502 Programmieren in Assembler, DM 59,–
Umweltdynamik (Prospekt anfordern), DM 66,– (NEU)

Macintosh Programmierhandbuch mit MSBASIC 2.0 (Ende '85), DM 59,–
Einführung in die Mikrocomputer-Technik, DM 66,–
M68000-Familie, 2 Bände, DM 79,– und DM 69,–

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. Auslieferung ab 1.6.86
Normalpreis DM 128,–
Sonderpreis für Peeker-Abonnenten DM 98,–

ProDOS-Version. Auslieferung ab 1.9.86
Normalpreis DM 128,–
Sonderpreis für Peeker-Abonnenten DM 98,–
Kombinationspreis für Bezieher der DOS-3.3-Version DM 28,–

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