Nr. 4/86 April
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

PEEKER

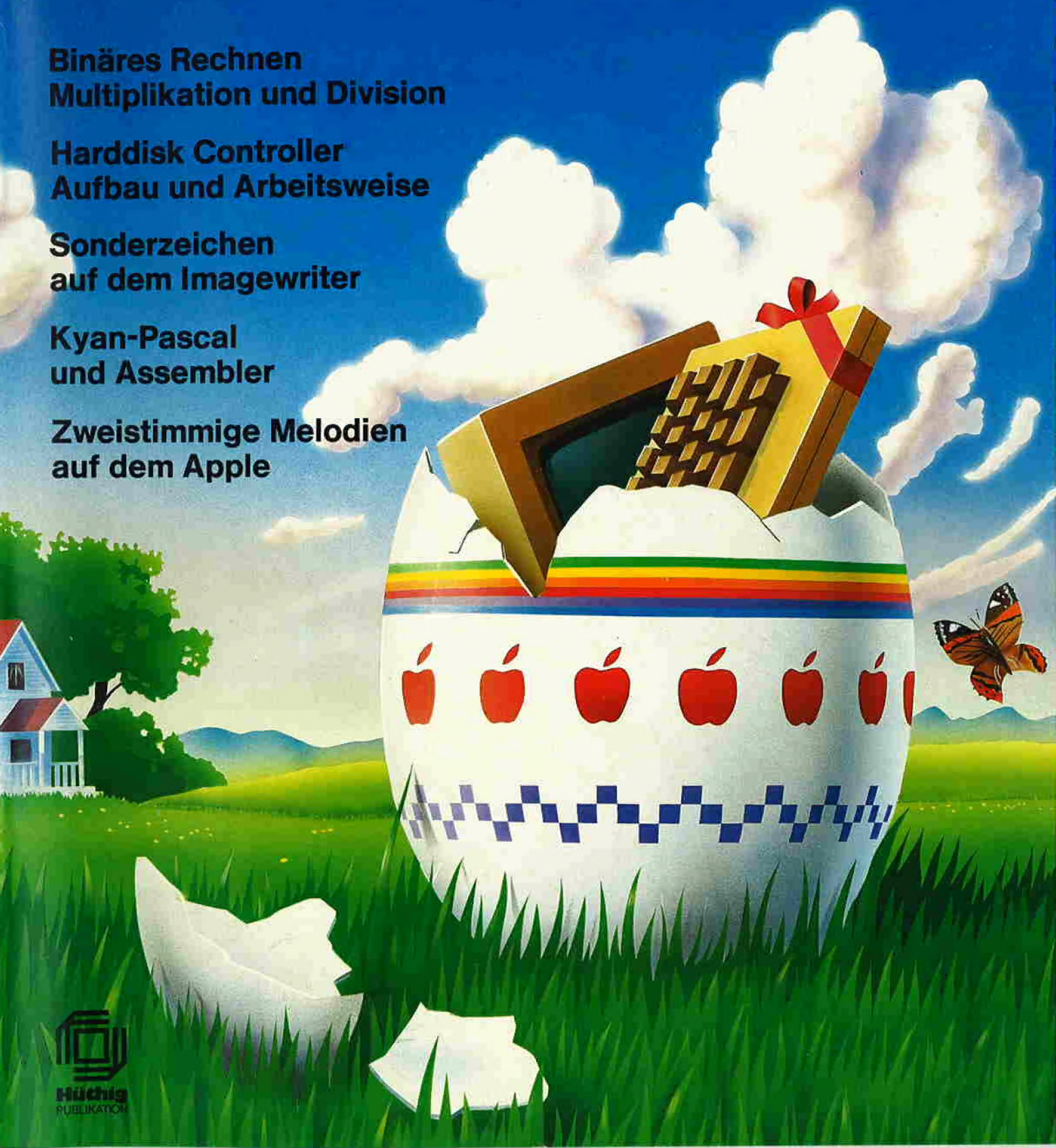
Binäres Rechnen
Multiplikation und Division

Harddisk Controller
Aufbau und Arbeitsweise

Sonderzeichen
auf dem Imagewriter

Kyan-Pascal
und Assembler

Zweistimmige Melodien
auf dem Apple



Hüthig
PUBLIKATION

DAS APPLE II HANDBUCH

Die rasche Orientierung für APPLE IIplus, IIe, IIc

Schnelle Antwort auf Alltagsfragen am APPLE II – leicht nachzuschlagen, praxisbezogen, für IIplus, IIe, IIc in nur 1 Buch!
Unterschiede IIplus/IIe, DOS 3.3/ProDOS, E/A-Interfacekarten/Ports, 40/80 Zeichendarstellung, US/DTS-Tastatur, 48K/128K Systeme etc.
Grafik/Soundmöglichkeiten, eine der APPLE-Stärken, in stark erweiterter Beschreibung.
Kurzführer „Steckkartenerweiterungen" mit Fotos; „Sofortbetrieb von Disketten/Cassettengeräten"; „Druckerbetrieb"; „Direktbefehle"; „Tastaturbedienung" etc.

Backgroundwissen: BASIC für Beginner/Professionelle; MC/BASIC-Kombination; MC-Entwicklung mit MONITOR/MINIASSEMBLER; APPLE-PASCAL-BS; Disketten/Plattenspeicherung; Dateiformate etc.
Ausführlicher Anhang zu Editor, Speicherbelegung, Codes des APPLESOFT-Interpreters etc.

DAS APPLE II HANDBUCH für IIplus, IIe, IIc, 472 Seiten, Softcover, DM 66,–

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

Weiterführende Literatur...

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

LOGO – Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den U.S.A. Für die Computer APPLE II, C-64, IBM PC, ATARI bis 520 ST., TI-99 und Schneider CPCs.
Hochwertiges Textbuch für Logo-Kurse für 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,–

Im 4439

NEU
APPLE WORKS + APPLE II, IIe, IIc = Elektronischer Schreibtischmanager
Dieses Programmsystem vereinigt die Funktion Texterstellung, Datenarchivierung, Formblattkalkulation, Datenfernübertragung. Das System mit den höchsten Verkaufsziffern. Sämtliche System/Anwendungsfragen in 2 Bänden. Beispiele aus der Wirtschaft u. v. m.
je 264 Seiten, je DM 49,–

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,–

Noch im Programm:
6502 – Programmieren in Assembler DM 59,–
VisiCalc, 50 Programme auf Diskette, DM 79,–
Computer für Kinder, APPLE II, DM 29,80

In Vorbereitung:
Macintosh Programmier-Handbuch mit Microsoft BASIC 2.0 DM 59,–

EDITORIAL

Die Revolution entläßt ihre Kinder

Nachdem Anfang dieses Jahres die Firma Osborne, die uns den ersten tragbaren Mikrocomputer beschert hatte, in den USA endgültig Konkurs anmelden mußte, brauen sich nunmehr auch über der Firma Commodore düstere Wolken zusammen. Wie in den einschlägigen Wirtschaftsblättern (z. B. ausführlich im „Manager-Magazin", 3/85) nachzulesen war, steht Commodore von einem Schuldenberg in Höhe von rund 192 Millionen Dollar oder umgerechnet fast einer halben Milliarde Mark. Wenn der Reinerlös (= Umsatz minus Selbstkosten) eines einzelnen Mikrocomputers im Durchschnitt aller Gerätetypen vom C64 bis zur Amiga etwa 400 Mark pro Gerät betragen würde, so müßten beispielsweise alle Einwohner Münchens, vom Baby bis zum Greis, je einen Commodore-Mikro erwerben, um den Schuldenberg abtragen zu helfen. Eine erschreckende Perspektive angesichts der allgemeinen Marktflaute in der PC-Branche, von der „durch die Bank" (Äquivokation beabsichtigt!) seit Anfang 1985 fast alle Firmen ergriffen worden sind. Nicht ohne Grund hat beispielsweise IBM jüngst die Preise für ihre PCs in den USA drastisch reduziert, um die Flut der Kombatiblen zu bremsen. Und was die Firma Apple anbelangt, so büßte sie in ihrem Wendejahr 1985 ca. 50 % ihres Marktanteils ein, der aufgrund unabhängiger Marktuntersuchungen in Deutschland von 16-18 % auf ca. 8-9 % geschrumpft sein soll. Dank der ungewöhnlich hohen Barreserven konnte Apple jedoch ohne Hilfe der Banken Umstrukturierungsmaßnahmen durchführen, während für Commodore der finanzielle Spielraum für derartige Reorganisationen gegenwärtig stark eingeschränkt ist. Wenn man bedenkt, daß die Pioniere Apple und Commodore einst den Markt beherrschten, so kann man sich nicht des Eindrucks erwehren, daß die Mikro-Revolution allmählich ihre Kinder entläßt.

Fast-Writer

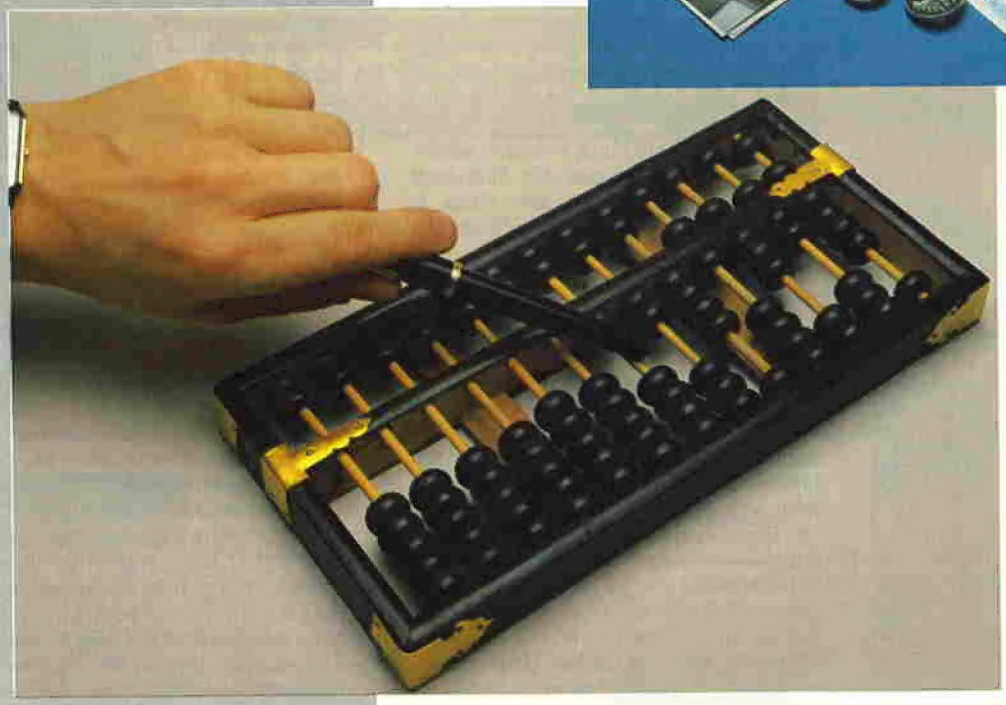
Unser ehemaliger Mitarbeiter Harald Grumser hat leider das Lager gewechselt und wird zukünftig Programme für den Großen Bruder HAL (vgl. „2001") entwickeln. Mit seinen superschnellen Programmen, etwa „Garbage Collection" (Heft 1/85) oder „Quicksort" (Heft 1/86), hatte er sich als „Fast-Program-Writer" profiliert, und so ist es mir ein besonderes Bedürfnis, als sein „Abschiedsgeschenk" für den Hüthig Software Service den „Fast-Writer" ankündigen zu können, der nach dem Druck des Manuals in etwa acht Wochen erscheinen wird. Es handelt sich dabei um ein professionelles Textverarbeitungsprogramm für den Apple IIe und IIc (in 40 Z/Z auch für den II+), dessen Verarbeitungsgeschwindigkeit im umgekehrten Verhältnis zu seinem Preis (DM 98,–) steht. Nach all meinen bisherigen Vergleichen ist der Fast-Writer mit Abstand das schnellste Textverarbeitungsprogramm, das jemals für den Apple II entwickelt worden ist. Das Wichtigste an einem Schreibprogramm ist, daß man schnell schreiben kann. Diese Forderung klingt zwar banal, doch wurde sie bislang von keinem Apple-Textprogramm erfüllt. Der Wordstar war bei etwa 100, der Applewriter bei etwa 200 Zeichen/Minute überfordert. Der Fast-Writer scrollt stufenlos, der Applewriter hüpfend und der Wordstar wortweise. Aber nicht nur das Scrollen ist beim Fast-Writer überragend schnell. Wenn man beispielsweise beim Fast-Writer (FW) und beim 64K-Applewriter (AW) den Speicher mit dem Muster „xxx xxx xxx ..." (27K) füllt, ergeben sich für das Ersetzen („xxx" durch „yyy" usw.) folgende Sekundenwerte:

xxx → yyy: AW: 89s, FW: 5s (Faktor 17)
xxx → xx: AW: 110s, FW: 5s (Faktor 22)
xx → yyy: AW: 118s, FW: 5s (Faktor 23)

Bei diesen Zahlen erübrigt sich jeder weitere Kommentar.

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

Heft 4/1986

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 #

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: Postgiro-konten: BRD: Karlsruhe 485 45-753; Österreich: Wien 755588; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 47718; Belgien: Brüssel 723026; Dänemark: Kopenhagen 349 69; Norwegen: Oslo 994 24; Schweden: Stockholm 5477 76-5
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 0 2165 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

Binäres Rech mit Papier ur

Teil 2: Multiplikation und Division

von Ulrich Stiehl

Dieser zweite Teil setzt voraus, daß Sie den Stoff aus dem ersten Teil (Peeker 2/86) bereits kennen.

Gliederung

5. Multiplikation

5.1. Dezimale Multiplikation

Multiplizieren heißt malnehmen oder vervielfachen. Der Ausdruck

$7 * 5 = 35$

im Sinne von „siebenmal die Fünf“ (und *nicht* „die Sieben fünfmal“) läßt sich in die Addition

$5 + 5 + 5 + 5 + 5 + 5 + 5 = 35$

überführen, wobei 7 als Multiplikator MR (Malzahl), 5 als Multiplikand MD (malgenommene Zahl) und 35 als Produkt P (Ergebnis) bezeichnet werden. Bei der mündlichen Multiplikation wird *zuerst* der

nen
d Bleistift

Multiplikator, bei der schriftlichen Multiplikation *zuerst* der Multiplikand genannt, also

```
mündlich:      MR      *  MD   =   P
               7       *  5    =   35
schriftlich:   MD      *  MR   =   P
               98765   *  7    =   691355
```

Dies steht im Widerspruch zu mir bekannten Schulbüchern und beruht offenbar auf einem tradierten Mißverständnis, über das in der Zwischenzeit keiner mehr nachgedacht hat. Rechnen Sie einmal „98765 * 7" schriftlich aus und murmeln Sie das, was Sie dabei denken, vor sich hin. Sie sagen „7 * 5, 7 * 6, 7 * 7" usw. und nicht „98765 * 7", denn sonst wären Sie Adam Ries!

Obwohl wir natürlich mit der schriftlichen Multiplikation vertraut sind, ist es nicht einfach, den sich dahinter verbergenden Algorithmus ins Bewußtsein zu heben. Gerade dies müssen wir jedoch tun, wenn wir die binäre Multiplikation im allgemeinen und die 6502-Multiplikation im besonderen verstehen wollen. Hierzu ist es erforderlich, die Phasen oder Komponenten der Multiplikation isoliert zu betrachten, wobei auch Dinge beachtet werden müssen, die man bei der konventionellen Papier- und Bleistiftmethode außer acht lassen würde.

Beispiel: Wenn Sie auf einem weißen Blatt nur eine einzige schriftliche Multiplikation durchzuführen hätten, stünde Ihnen genug „freier Raum" zur Verfügung. Wenn die Aufgabe jedoch lauten würde, auf einem karierten Papier bestimmter Größe möglichst viele 7-mal-5-stellige Multiplikationen unterzubringen, müßten Sie sich schon Gedanken über die Feldlängen der Teil- und Endprodukte machen, denn Sie würden z.B. links und rechts nicht genügend Platz haben, um die Teilprodukte „beliebig" zu verschieben.

5.1.1. Zerlegungsverfahren

Die schriftliche Multiplikation setzt das (auswendig gelernte) kleine Einmaleins voraus. Die Aufgabe
98765 * 7
ließe sich nicht lösen, wenn wir nicht bereits die Produkte von 7 * 5, 7 * 6, 7 * 7 usw. wüßten. Man kann diese Multiplikation aber auch in eine reine Addition umwandeln:

```
  98765 1mal
+ 98765 2mal
+ 98765 3mal
+ 98765 4mal
+ 98765 5mal
+ 98765 6mal
+ 98765 7mal
-------------
 691355 Summe = Produkt
```

Im Falle eines einstelligen Multiplikators wäre dieses Verfahren gerade noch angängig. Für die Aufgabe
77777 * 98765
würde man jedoch bereits einen Karton Papier benötigen. Also müssen wir uns bei mehrstelligen Zahlen Stelle für Stelle vorarbeiten, was als Zerlegung (in Einer, Zehner, Hunderter, Tausender usw.) bezeichnet wird. Beispiel:

```
     HZE
77 * 321
---------
          MR  MD       MR    MD
   77 =    1 * 77 oder 1 *   77
  154x =  20 * 77 oder 2 *  770
 231xx = 300 * 77 oder 3 * 7700
---------
  24717
```

Wie ersichtlich, gibt es zwei Interpretationen. So kann etwa die Multiplikation des Multiplikator-Zehners 2 mit dem Multiplikanden 77 als
2(0) * 77 oder als
2 * 77(0)
aufgefaßt werden. Für die Zehnerstelle können wir aber auch (2 * 77) * 10 schreiben, d.h. allgemein formuliert:

(MR-Stellenwert * MD) * 10 ↑ MR-Stelle.

Bei der Papier-und-Bleistift-Multiplikation geht man nun noch einen Schritt weiter und zerlegt nicht nur den Multiplikator, sondern auch den Multiplikanden, denn man rechnet nicht 2 * 77, sondern
2 * 7 = 4; Übertrag 1
2 * 7 = 14; 14 + 1 = 15.

Anstelle der Zehnerstelle-Multiplikation 2(0) * 77 können wir auch die 154 als Teilprodukt von 2 * 77 nach links schieben (154x), was durch das „x" in dem obigen Beispiel angedeutet wird. Jede Verschiebung um eine Stelle nach links entspricht einer Multiplikation mit 10 (oder allgemein einer Multiplikation mit der Basis des jeweiligen Zahlensystems):

```
  7 =   7 = 7 * (10 ↑ 0)
 7x =  70 = 7 * (10 ↑ 1)
7xx = 700 = 7 * (10 ↑ 2)
```

Wir merken uns:

1. Entweder schieben wir den Multiplikanden nach links und multiplizieren *danach* mit der Multiplikatorstelle (= **Multiplikandenverschiebung**),

2. oder wir multiplizieren den Multiplikanden mit der Multiplikatorstelle und schieben *danach* das Teilprodukt nach links (= **Teilproduktverschiebung**).

5.1.2. Links- und Rechtsverfahren

Man kann beim Multiplikator mit der links stehenden, höchsten Stelle (z.B. hier mit der Tausenderstelle) beginnen. Dies nennen wir **Linksverfahren** oder genauer Von-links-nach-rechts-Verfahren. Mit jeder weiteren Multiplikatorstelle, die wir von links wegnehmen, müssen wir das entsprechende Teilprodukt um eine Stelle nach rechts einrücken. Die Teilprodukte sind damit treppenförmig von links nach rechts eingerückt.

Bei der Assemblerprogrammierung kann man das Wegnehmen einer Multiplikatorstelle durch Verschiebebefehle realisieren. Wenn man eine Linksverschiebung (ASL oder ROL) vornimmt, wird das jeweils höchste Bit, also die höchste Multiplikatorstelle „weggenommen" (= in das Carry-Flag übertragen).

Linksverfahren-Beispiel

```
MD      MR
1111 * 4321
-----------
   x4444xxx T   ZW: 04444000 T
   xx3333xx H   ZW: 04777300 T+H
   xxx2222x Z   ZW: 04799520 T+H+Z
   xxxx1111 E   ZW: 04800631 T+H+Z+E
-----------
   x4800631 P
```

Umgekehrt kann man beim Multiplikator mit der rechts stehenden, niedrigsten Stelle (regelmäßig mit der Einerstelle) beginnen. Dies nennen wir **Rechtsverfahren** oder genauer Von-rechts-nach-links-Verfahren. Mit jeder weiteren Multiplikatorstelle, die wir von rechts wegnehmen, müssen wir das entsprechende Teilprodukt um eine Stelle nach links einrücken. Die Teilprodukte sind damit treppenförmig von rechts nach links eingerückt.

Bei der Assemblerprogrammierung kann man das Wegnehmen einer Multiplikatorstelle durch Verschiebebefehle realisieren. Wenn man eine Rechtsverschiebung (LSR oder ROR) vornimmt, wird das jeweils niedrigste Bit, also die niedrigste Multiplikatorstelle „weggenommen" (= in das Carry-Flag übertragen).

Rechtsverfahren-Beispiel

```
MD      MR
1111 * 4321
-----------
   xxxx1111 E   ZW: 0000111 1 E
   xxx2222x Z   ZW: 000233 31 E+Z
   xx3333xx H   ZW: 00356 631 E+Z+H
   x4444xxx T   ZW: 0480 0631 E+Z+H+T
-----------
   x4800631 P
```

Unter **Teilprodukt** versteht man das Produkt einer Multiplikatorstelle mit dem Multiplikanden (unter Berücksichtigung der Stellenverschiebung!). Unter **Zwischensumme** versteht man die Summe der bislang errechneten Teilprodukte. Unter Endprodukt oder kurz Produkt (P) versteht man die Summe aller Teilprodukte oder die letzte Zwischensumme. Wir haben hier im Vorgriff auf den nächsten Abschnitt zusätzlich die Zwischensummen (ZW) ausgewiesen.

Bei dem Rechtsverfahren bemerken wir übrigens, daß jede weitere Zwischensumme eine weitere gültige Endziffer aufweist.

Um zu einem Algorithmus zu gelangen, muß man sich Gedanken über die Stellenanzahl oder **Zahlenfeldlänge** machen. Betrachten wir hierzu die Multiplikation:

```
4321   4321   87654321 Stellen
9999 * 9999 = 99980001 Zahlen
```

Wir ersehen, daß die 4-mal-4-stellige Multiplikation höchstens zu einem 8stelligen Produkt führt. Allgemein gilt, daß bei einer x-mal-x-stelligen Multiplikation ein x-plus-x-stelliges Produktfeld immer ausreicht. Wenn wir die Teilprodukte der E-, Z-, H- und T-Multiplikationen in einem 8stelligen Feld ausrichten und dabei das Verschieben der Teilprodukte analysieren, so können wir zwei Phänomene erkennen:

1 Bei dem Rechtsverfahren wird das erste Teilprodukt (für die E-Multiplikation) zunächst *ganz rechtsbündig* ausgerichtet; danach wird mit jeder weiteren Multiplikatorstelle jedes weitere Teilprodukt um eine Stelle nach links gerückt. Demgegenüber wird bei dem Linksverfahren das erste Teilprodukt (hier für die T-Multiplikation) *nicht ganz linksbündig*, d.h. um eine Stelle von der linken Kante entfernt, ausgerichtet; danach wird mit jeder weiteren Multiplikatorstelle jedes weitere Teilprodukt um eine Stelle nach rechts gerückt.

2 Bei einem 4stelligen oder allgemein x-stelligen Multiplikator muß das Teilprodukt 3mal oder allgemein (x-1)-mal nach links oder rechts geschoben werden.

5.1.3. Normalverfahren

Bei dem obigen Beispiel (5.1.2) wurden die E-, Z-, H- und T-Teilprodukte *direkt* untereinandergeschrieben. Allgemein führt ein x-stelliger Multiplikator zu x Teilprodukten. Bei der computermäßigen Multiplikation muß man hingegen *zusätzlich* Zwischensummen bilden, weil sonst zuviel Speicherplatz verschwendet wird – eine 24-mal-24-Bit-Multiplikation benötigt 72 Bytes für die Teilprodukte –, zumal das Ablegen der Teilprodukte im Speicher (fast) solange dauert wie das Aufaddieren zur Zwischensumme.

Bei dem Rechtsverfahren würde man folgende Zwischensummen ZW bilden:

```
1111 * 4321
-----------
   00000000 0. ZW
 + 00001111 E
-----------
   00001111 1. ZW
 + 0002222x Z
-----------
   00023331 2. ZW
 + 003333xx H
-----------
   00356631 3. ZW
 + 04444xxx T
-----------
   04800631 4. ZW = P
===========
```

Für das Rechtsverfahren gilt:
0. ZW = 0
1. ZW = E-P (Einerprodukt)
2. ZW = E-P + Z-P
3. ZW = E-P + Z-P + H-P
4. ZW = E-P + Z-P + H-P + T-P
Die letzte Zwischensumme ist gleichzeitig das Gesamtprodukt der Multiplikation.

Für das Linksverfahren gilt:
0. ZW = 0
1. ZW = T-P (Tausenderprodukt)
2. ZW = T-P + H-P
3. ZW = T-P + H-P + Z-P
4. ZW = T-P + H-P + Z-P + E-P

Das soeben skizzierte Verfahren bezeichnen wir als **Normalverfahren** oder genauer als normales Zwischensummenverfahren. Das besondere Merkmal dieser Methode ist, daß die Position der Zwischensummen unverändert bleibt und nur das jeweilige Teilprodukt nach rechts (= normales Rechts[zwischensummen]verfahren) oder nach links (normales Links[zwischensummen]verfahren) verschoben wird. Die Teilproduktverschiebung kann auch als Multiplikandenverschiebung gedeutet werden (s. 5.1.1.). Wir merken uns deshalb:
Beim Normalverfahren wird das Teilprodukt oder der Multiplikand, aber niemals die Zwischensumme selbst verschoben.

5.1.4. Tausendeinsverfahren

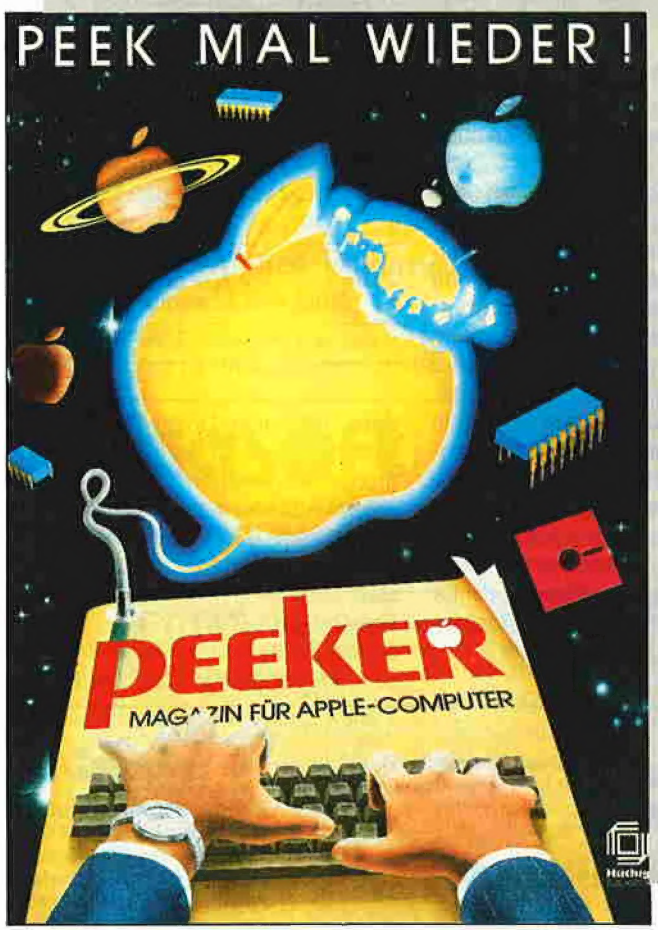
Diesen Abschnitt sollten Sie ggf. überspringen, da jetzt eine Methode vorgestellt wird, die mit „Papier und Bleistift" nichts zu tun hat.
Wie bereits eingangs bemerkt, sind die bekannten 6502-Einführungen bezüglich der Binärmathematik völlig undidaktisch aufgebaut. Dies wird an den dort präsentierten Beispielen für die 8-mal-8-Bit-Multiplikation besonders deutlich. In dem Buch „Kehrel: Apple Assembler lernen" findet sich der weiter unten als Tausendeins-Linksverfahren bezeichnete Algorithmus, der in identischer Form in „Inman: Apple Machine Language", in „Leventhal: 6502 Assembly Language Programming" und ähnlichen Einführungen auftaucht. In all diesen Lehrbüchern wird dieser exotische Algorithmus kommentarlos in den Raum gestellt – wahrlich eine didaktische Zumutung, zumal es sich nicht einmal um den schnellsten Algorithmus handelt!
Was hat es nun mit dem Tausendeinsverfahren auf sich? Betrachten wir zu diesem Zweck den Anfang der folgenden Multiplikation nach dem „anomalen" Links[zwischensummen]verfahren (= Tausendeins-Linksverfahren):

```
0001 * 1000
-----------
   00000000 0. ZW
 + 00000001 1000 * 1!
-----------
   00000001 ???
```

Ist 1000 * 1 = 1? Dies kann doch wohl nicht stimmen! Was ist geschehen? Während wir beim normalen Links[zwischensummen]verfahren den Multiplikanden oder das Teilprodukt verschieben, wird

GRATIS

FÜR ABONNENTEN



Dieses 4-farbige Poster im Format 420 mm × 580 mm erhalten »**peeker**«-Abonnenten kostenlos. Es liegt der Ausgabe 5/86 bei.

Wenn Sie kein »**peeker**«-Abonnement haben, das Poster mit 4 nützlichen Tabellen auf der Rückseite (Applesoft, DOS und ProDOS, Pascal-Befehle 6502/65C02) trotzdem haben möchten, bestellen Sie bitte mit Coupon und DM 5,– in Briefmarken.

Wenn Sie kein »**peeker**«-Abonnent sind, den »**peeker**« aber schon immer abonnieren wollten, tun Sie's jetzt. Denn dann bekommen auch Sie Ihr »**peeker**«-Poster gratis! Einfach den Bestellschein ausfüllen und ab zur Post.

peeker 4/86

Bestellcoupon

☐ **Ja**, schicken Sie mir das »**peeker**«-Poster, DM 5,– in Briefmarken anbei.

☐ **Ja**, ich abonniere **jetzt** den »**peeker**« und erhalte das Poster gratis.

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, 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

Bitte lesen!

Coupon ausschneiden oder kopieren und einsenden an:

»peeker«
Abonnementservice
Im Weiher 10
6900 Heidelberg 1

Hüthig
PUBLIKATION

beim anomalen Tausendeins-Linksverfahren die Zwischensumme selbst verschoben. *Deshalb ist jede Zwischensumme bis auf die letzte falsch.* Um dies begreifbar zu machen, stellen wir Tausendeins- und Normalverfahren einander gegenüber. Zur Verdeutlichung fügen wir eine zusätzliche Verschiebungszeile ein („Shift"), die beim 1001-Verfahren die Verschiebung der Zwischensumme und beim Normalverfahren die Verschiebung des Multiplikanden anzeigt. Beim 1001-Verfahren muß man deshalb jeweils die 2. und 3. und beim Normalverfahren die 1. und 3. Zeile addieren:

Rechtsverfahren

```
anomal 1001     normal

2345 * 6789     2345 * 6789
-----------     -----------
  0000 0000       0000 0000   0. ZW
  0000 0000       xxxx 2345   kein Shift
+21105          +    2 1105   MD * 9
-----------     -----------
 21105 0000       0002 1105   1. ZW
> 2110 5000     < xxx2 345x   Shift
+18760          +   18 7600   MD * 8(0)
-----------     -----------
 20870 5000       0020 8705   2. ZW
> 2087 0500     < xx23 45xx   Shift
+16415          +  164l 500   MD * 7(00)
-----------     -----------
 18502 0500       0185 0205   3. ZW
> 1850 2050     < x234 5xxx   Shift
+14070          + 1407 0000   MD * 6(000)
-----------     -----------
 15920 2050       1592 0205   4. ZW
> 1592 0205     ===========   Shift
===========
```

Zur Verdeutlichung haben wir 2 Vierergruppen gebildet, die linke und die rechte Vierergruppe.
Beim *normalen Rechtsverfahren* wird der Multiplikand zunächst in der rechten Vierergruppe eingesetzt. *Nach* jeder Addition – mit Ausnahme der letzten – wird der Multiplikand um 1 Stelle nach links verschoben [<–]. Es finden damit 4 Additionen und 3 Linksverschiebungen des Multiplikanden statt:
A-L-A-L-A-L-A
Da der Multiplikand infolge der sukzessiven Linksverschiebung die beiden Vierergruppen überlappt, muß die Addition stets beide Vierergruppen erfassen.
Beim *anomalen Rechtsverfahren* (1001-Methode) wird der unverschobene Multiplikand mit der jeweiligen Multiplikatorstelle malgenommen und das sich ergebende Teilprodukt stets in der linken Vierergruppe eingetragen. Da die 4 Stellen der linken Vierergruppe meist nicht ausreichen, wird eine zusätzliche Übertrag-Stelle eingerichtet. Die linke Vierergruppe ist mithin eigentlich eine „Fünfergruppe". *Nach* jeder Addition – einschließlich der letzten – wird die sich ergebende Zwischensumme um 1 Stelle nach rechts verschoben [–>], womit der Übertrag wieder in die linke Vierergruppe hineingeschoben wird. Es finden damit 4 Additionen und 4 Rechtsverschiebungen statt:
A-R-A-R-A-R-A-R

Da der Multiplikand nicht verschoben wird, muß die Addition nur die beiden linken Vierergruppen erfassen, wobei jedoch meist ein Übertrag zu berücksichtigen ist.

Linksverfahren

```
anomal 1001     normal

2345 * 6789     2345 * 6789
-----------     -----------
  0000 0000       0000 0000   0. ZW
  0000 0000     > x234 5xxx   Shift
+    1 4070     + 1407 0000   MD * 6(000)
-----------     -----------
  0001 4070       1407 0000   1. ZW
< 0014 0700     > xx23 45xx   Shift
+    1 6415     +  164 150    MD * 7(00)
-----------     -----------
  0015 7115       1571 1500   2. ZW
< 0157 1150     > xxx2 345x   Shift
+    1 8760     +   18 7600   MD * 8(0)
-----------     -----------
  0158 9910       1589 9100   3. ZW
< 1589 9100     > xxxx 2345   Shift
+    2 1105     +    2 1105   MD * 9(0)
-----------     -----------
  1592 0205       1592 0205   4. ZW
===========     ===========
```

Beim *normalen Linksverfahren* wird der Multiplikand zunächst in der linken Vierergruppe eingetragen und bereits *vor* der ersten Addition um 1 Stelle nach rechts verschoben [–>]. Es finden damit 4 Rechtsverschiebungen und 4 Additionen statt:
R-A-R-A-R-A-R-A
Da der Multiplikand infolge der sukzessiven Rechtsverschiebung die beiden Vie-

rergruppen überlappt, muß die Addition stets beide Vierergruppen erfassen.
Beim *anomalen Linksverfahren* wird der unverschobene Multiplikand mit der jeweiligen Multiplikatorstelle malgenommen und stets in der rechten Vierergruppe eingetragen. Da die 4 Stellen der rechten Vierergruppe meist nicht ausreichen, erfolgt ein Übertrag in die erste Stelle der linken Vierergruppe. Dies ist normal, denn anders als beim anomalen Rechtsverfahren braucht keine zusätzliche Übertrag-Stelle eingerichtet zu werden. Nach jeder Addition – mit Ausnahme der letzten – erfolgt eine Linksverschiebung [<–]. Es finden damit 4 Additionen und 3 Linksverschiebungen statt:
A-L-A-L-A-L-A
Da der Multiplikand nicht verschoben wird, muß die Addition nur die beiden rechten Vierergruppen erfassen, wobei jedoch meist ein Übertrag zu berücksichtigen ist.

5.2. Binäre Multiplikation

Für die binäre Multiplikation gelten folgende Grundregeln:

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

5.2.1. Zerlegungsverfahren

Während bei der dezimalen Multiplikation das kleine Einmaleins beherrscht werden muß, reduziert sich das binäre Zerlegungsverfahren auf die Frage, ob der Multiplikand mit 1 oder mit 0 multipliziert werden soll, oder anders formuliert, ob der *unveränderte*, wenngleich stellenverschobene Multiplikand oder Null als Teilprodukt notiert werden soll. Fazit: Die Multiplikation des Multiplikanden mit einem Einserbit des Multiplikators läuft auf das stellenverschobene *Abschreiben* des Multiplikanden hinaus. Beispiele:

```
1111 * 11
---------
   1111
  1111

1111 * 00
---------
   0000
  0000
```

5.2.2. Links- und Rechtsverfahren

Die Links- und Rechtsverfahren sind analog anwendbar. Bei den nachfolgenden Beispielen sind wieder die Überträge als Additionshilfen vermerkt:

Linksverfahren-Beispiel

```
MD      MR
1011 * 1110   (11 * 14)
-------------
   x1011000   1 * 1011000
   xx101100   1 *  101100
   xxx10110   1 *   10110
   xxxx0000   0 *    1011
-------------
    11111     Übertrag
-------------
   10011010   (154)
=============
```

Rechtsverfahren-Beispiel

```
MD      MR
1111 * 1011   (15 * 11)
-------------
   xxxx1111   1 *    1111
   xxx11110   1 *   11110
   xx000000   0 *  111100
   x1111000   1 * 1111000
-------------
     1111     Übertrag
     1111     Übertrag
-------------
   10100101   (165)
=============
```

Wie ersichtlich, führt eine 4-mal-4-Bit-Multiplikation höchstens zu einem 8stelligen Produkt. Allgemein gilt, daß eine x-plus-x-stellige Zahlenfeldlänge für ein x-mal-x-Bit-Produkt stets ausreichend ist.

5.2.3. Normalverfahren

Auch das normale Zwischensummenverfahren ist analog anwendbar. Bei dem nachfolgenden Beispiel für eine 8-mal-8-Bit-Multiplikation im Rechtsverfahren stehen jeweils Zwischensumme (ZW) und Teilprodukt als 16-Bit-Zahlen untereinander, wobei neben dem Teilprodukt der 8-Bit-Multiplikator (MR) mit der rechts herausgeschobenen Multiplikatorstelle notiert wird.

```
MD       * MR
00001111 * 10101010
15       * 170

 00000000 00000000   0. ZW
+00000000 00000000   x1010101-0 MR
-------------------
 00000000 00000000   1. ZW
+00000000 00011110   xx101010-1 MR
-------------------
 00000000 00011110   2. ZW.
+00000000 00000000   xxx10101-0 MR
-------------------
 00000000 00011110   3. ZW
+00000000 01111000   xxxx1010-1 MR
-------------------
 00000000 10010110   4. ZW
+00000000 00000000   xxxxx101-0 MR
-------------------
 00000000 10010110   5. ZW
+00000001 11100000   xxxxxx10-1 MR
-------------------
 00000010 01110110   6. ZW
+00000000 00000000   xxxxxxx1-0 MR
-------------------
 00000010 01110110   7. ZW
+00000111 10000000   xxxxxxxx-1 MR
-------------------
 00001001 11110110   8. ZW
===================
```

5.2.4. Tausendeinsverfahren

Das Tausendeinsverfahren ist aus programmiertechnischer Sicht – nicht aus der Papier-und-Bleistift-Sicht! – effizienter als das Normalverfahren. Der Abschnitt über das dezimale Tausendeinsverfahren (5.1.4) wird hier vorausgesetzt. Zunächst ein Beispiel:

```
Rechtsverf.          Linksverf.

1111 * 1011          1111 * 1011
-----------          -----------
  0000 0000            0000 0000
  0000 0000          < 0000 0000
+ 1111               +      1111   MD
  0                          0     Übertrag
-----------          -----------
 01111 0000            0000 1111
> 0111 1000          < 0001 1110
+ 1111               +      0000   MD
  1                          0     Übertrag
-----------          -----------
 10110 1000            0001 1110
> 1011 0100          < 0011 1100
+ 0000               +      1111   MD
  0                          1     Übertrag
-----------          -----------
 01011 0100            0100 1011
> 0101 1010          < 1001 0110
+ 1111               +      1111   MD * 0
  1                          1     Übertrag
-----------          -----------
 10100 1010            1010 0101
> 1010 0101          ===========
===========
```

In Abweichung zum dezimalen Tausendeinsverfahren können wir folgendes konstatieren:
1. Beim binären Rechts/Linksverfahren paßt das Teilprodukt exakt in die linke/rechte Vierergruppe, während beim dezimalen Rechts/Linksverfahren eine „Fünfergruppe", d.h. ein 5stelliges Feld vorgesehen werden muß.
2. Erst wenn die Zwischensumme gebildet wird, ist beim binären Tausendeinsverfahren ein Übertrag zu erwarten. Beim Rechtsverfahren geht der Übertrag in die 9. Stelle (externer Übertrag) und beim Linksverfahren in die 5. Stelle (interner Übertrag).
3. Beim Rechtsverfahren kann sich die Addition auf die linke Vierergruppe beschränken. Ein ggf. entstehender externer Übertrag wird durch die anschließende Rechtsverschiebung wieder in die linke Vierergruppe hineingeschoben.
4. Beim Linksverfahren kann sich die Addition auf die rechte Vierergruppe beschränken. Ein ggf. entstehender interner Übertrag führt bei der linken Vierergruppe höchstens zu einer Addition von 1 (wozu programmtechnisch ein Inkrement-Befehl ausreicht).

Wenn wir das Beispiel generalisieren und anstelle der Vierergruppen von „Achtergruppen" (= Bytes) sprechen, haben wir den Schlüssel zur binären Multiplikation nach dem Tausendeinsverfahren gefunden.

Treue wird belohnt!

Applesoft-Editor

Mit der Sammeldiskette #16, die zum Heft 4/1986 verschickt wird, werden die Fortsetzungsbezieher der Peeker-Sammeldisketten *zusätzlich* einen vollständigen Applesoft-Editor *kostenlos* erhalten.
Mit dieser Zusatzleistung belohnen wir unsere treuen Stammkunden. Der Applesoft-Editor wird nur an diejenigen verschickt, die bereits vor dem 1. April 1986 Fortsetzungsbezieher der Sammeldisketten gewesen sind. Wer also bis zum 31. März 1986 noch einen Forsetzungsauftrag für mindestens 6 Sammeldisketten erteilt, ist mit dabei!

Bei dem Applesoft-Editor handelt es sich um einen von U. Stiehl entwickelten Makroeditor, der unter der Bezeichnung MUM (Macro Utilities Master) von der Firma Heyden Datasystems in Großbritannien und den USA vertrieben wird. Wie heißt es doch so schön in der englischen Anleitung: „You have chosen the right MUM and like every other mum this one will look after your special needs for a lifetime."

Kurzanleitung

Der Applesoft-Editor setzt einen Apple II+/e/c (40 Z/Z) mit einem Laufwerk voraus und ist für 48K-DOS-3.3 (oder Diversi-DOS usw.), nicht jedoch für ProDOS gedacht. (Hierfür gibt es den analogen PRODOS.EDITOR über den Hüthig Software Service.) Nach dem Booten von DOS 3.3 oder Diversi-DOS legen Sie die Sammeldiskette #16 ein und starten Sie den Editor mit
BRUN MACROEDITOR
Eine spezielle Schnell-Lade-Routine stellt selbst unter dem alten DOS 3.3 sicher, daß der Editor in maximal 2-3 Sekunden im Speicher installiert ist. Der Editor belegt den Bereich $7D00-$95FF ($7D00 = dezimal 32000) und kann auch gestartet werden, wenn sich ein Applesoft-Programm bereits im Speicher befindet. Und nun die Befehle im einzelnen:

&U
Damit läßt sich der Editor vorübergehend deaktivieren, um z.B. ein Applesoft-Programm mit RUN auszutesten, das jedoch nicht HIMEM höher als 32000 oder MAXFILES auf einen anderen Wert als 3 setzen sollte, weil sonst der Editor überschrieben werden würde. Das gleiche gilt für FP; dagegen ist NEW natürlich erlaubt.

&
Damit oder auch mit CALL 32000 wird der Editor wieder aktiviert. Nunmehr kann man alle nachfolgenden Befehle eingeben.

Z
Es erscheint das Wort „LINE", und man muß jetzt die Nummer der zu ändernden Programmzeile eingeben, die dann auf den letzten 6 Zeilen des Bildschirms angezeigt wird. Als spezielle Korrigierbefehle sind implementiert:
Ctrl-B
Sprung zum Beginn der Zeile.
Ctrl-E
Sprung zum Ende der Zeile.
Ctrl-Z
Eine Zeile tiefer.
Ctrl-T
Tabulieren, d.h. 10 Zeichen vorwärts.
Linkspfeil
Ein Zeichen nach links.
Rechtspfeil
Ein Zeichen nach rechts.
Ctrl-D
Delete bzw. Löschen des Zeichens, auf dem der Cursor gerade ruht.
Ctrl-I
Insert bzw. Einschalten des Einfüge-Modus ab der momentanen Cursor-Position. Jedes nunmehr eingegebene Zeichen schiebt die Applesoft-Zeile entsprechend auseinander. Außerdem können im Einfüge-Modus die meisten Ctrl-Zeichen eingegeben werden außer
Rechtspfeil, Linkspfeil, Return,
mit denen man den Einfüge-Modus wieder aufhebt.
Return
Übernahme der Applesoft-Zeile, so wie man sie am Bildschirm sieht, in den Programmspeicher.
Ctrl-P
Abhacken der Applesoft-Zeile ab der momentanen Cursor-Position und Übernahme des Zeilenanfangs in den Programmspeicher.
ESC
Verlassen des Editier-Modus, *ohne* daß die (veränderte) Zeile in den Programmspeicher übernommen wird. Ctrl-X und Ctrl-C bewirken das gleiche.

K
Wie **Z**, doch wird die Applesoft-Zeile in gepackter Form, d.h. ohne überflüssige Leertasten, angezeigt. Nützlich bei überlangen Programmzeilen.

Ctrl-K
Ausgabe des Makros „CATALOG".
Ctrl-L
Ausgabe des Makros „LIST".
Diese beiden Makros sind vordefiniert.
Ctrl-D
Es erscheint das Wort „Macro", und man kann nunmehr ein eigenes Tastatur-Makro definierten. Dazu tippt man zunächst Ctrl-Q oder Ctrl-W oder Ctrl-E oder Ctrl-R oder Ctrl-T oder Ctrl-P (alle in der dritten Reihe der Tastatur) und daraufhin das Makro (z.B. HOME: LIST usw.). Ein Makro kann 100 Zeichen lang sein. Das definierte Makro kann dann mit Ctrl-Q usw. aufgerufen werden. Wenn man bestimmte Makros permanent benötigt, so speichere man den ganzen Editor mit
BSAVE EDITOR.NEU, A32000, L6399

M + Monitorbefehl
führt einen Monitorbefehl von BASIC aus durch, z.B. M300L entspricht CALL -151, 300L, Ctrl-C.

&D + Dezimalzahl
verwandelt eine Dezimalzahl in eine Hexzahl, z.B. &D16 ergibt $0010.
&H + Hexzahl
verwandelt eine Hexzahl in eine Dezimalzahl, z.B. &H10 ergibt 16.

&L
zeigt die Länge des Applesoft-Programms in Bytes an.
&F
zeigt die Anzahl der freien Sektoren auf der Diskette an, auf die man zuvor mit Ctrl-K o.ä. zugreifen sollte; funktioniert nicht korrekt bei bestimmten, für 40 oder 80 Spuren modifizierten DOS-Varianten.

&V
listet alle Variablen des Applesoft-Programms in alphabetischer Reihenfolge. Mit POKE 38395, A (z.B. POKE 38395, 5) kann man die Anzahl der Verweise pro Zeile für den Ausdruck festlegen.
&R z,i
„renumbert" das gesamte Applesoft-Programm ab Zeile z im Intervall i, z.B. &R100,5

&K
entfernt bzw. „kruncht" alle REMs; vorher kommentiertes Programm abspeichern!

&S
Dieser Befehl dient in der Form
&S: Token, z.B. &S:GOTO
zum Suchen von Tokens und in der Form
&S"String, z.B. &S"HALLO
zum Suchen von Strings. Bei gemischten Ausdrücken muß man im Zweifelsfall beide Varianten durchprobieren.

& + Nummer
zeigt eine Applesoft-Zeile in der internen Form als ASCII-Hexdump an, z.B.
&100

Hüthig Software Service · 6900 Heidelberg · Postfach 102869

5.3. 6502-Multiplikation

Während die meisten 16-Bit-Prozessoren über mehr oder weniger komfortable Multiplikationsbefehle verfügen, muß bei 8-Bit-Prozessoren die Multiplikation auf Additions- und Schiebebefehle reduziert werden. Dies kostet natürlich Prozessortakte. Beispielsweise benötigt beim 68000 eine 16-mal-16-Bit-Multiplikation unter Verwendung des MULU-Befehls ca. 100 Takte, aber ohne Verwendung des MULU-Befehls, d.h. konventionell mit ADD-Befehlen usw., ca. 550 Takte. Ebenfalls ca. 550 Takte werden bei der 6502-16-mal-16-Bit-Multiplikation im optimierten Tausendeinsverfahren benötigt. Eine präzise Bestimmung der Takte ist indes nicht möglich, weil die Multiplikationsdauer u.a. von der Anzahl der Einserbits des Multiplikators abhängt.

Es gibt prinzipiell vier Multiplikationsalgorithmen:
1. MRLN = Von-rechts-nach-links-Normalverfahren.
2. MLRN = Von-links-nach-rechts-Normalverfahren
3. MRLT = Von-rechts-nach-links-Tausendeinsverfahren
4. MLRT = Von-links-nach-rechts-Tausendeinsverfahren

Nachfolgend stellen wir alle vier Varianten für die 8-mal-8-Bit-Multiplikation vor. Die Programme setzen die Beherrschung der 6502-Befehle ASL, LSR, ROL und ROR voraus, die aus Platzgründen an dieser Stelle nicht erläutert werden können. Es lassen sich folgende Schritte unterscheiden:

Schritt 1: Zunächst muß das spätere Produkt auf 0 gesetzt werden. Falls Multiplikator und Multiplikand erhalten bleiben sollen, müssen Shift-Multiplikator (und beim Normalverfahren Shift-Multiplikand) initialisiert werden. Man beachte, daß beim normalen Linksverfahren MDL in SMDH eingetragen wird (s. Listing).
Schritt 2: Als nächstes muß der Bit-Zähler für die (hier 8) Multiplikator-Bits gesetzt werden.
Schritt 3: Bei den Rechtsverfahren wird jetzt der Multiplikator rechtsverschoben. Bei den Linksverfahren wird jetzt entweder der Multiplikand rechtsverschoben (Normalverfahren) oder das Produkt (= momentane Zwischensumme und späteres Endprodukt) linksverschoben (Tausendeinsverfahren).
Schritt 4: Bei den Rechtsverfahren wird jetzt im Falle eines gesetzten Carry-Flags die Addition durchgeführt; man beachte die unterschiedlichen Additionen! Bei den Linksverfahren wird jetzt der Multiplikator linksverschoben.
Schritt 5: Bei den Rechtsverfahren wird jetzt der Multiplikand linksverschoben (Normalverfahren) oder das Produkt rechtsverschoben (Tausendeinsverfahren). Bei den Linksverfahren wird jetzt im Falle eines gesetzten Carry-Flags die Addition durchgeführt; man beachte die unterschiedlichen Additionen!
Schritt 6: Jetzt wird geprüft, ob alle (hier alle 8) Bits des Multiplikators erledigt sind. Wenn nein, dann zurück zu Schritt 3. Wenn ja, dann Ende.

Bemerkungen zu M8RLN/M8LRN

Wenn Sie M8RLN genau untersuchen, werden Sie feststellen, daß die *letzte* (hier 8.) Linksverschiebung des Shift-Multiplikanden (SMD) in Zeile 44 unnötig ist. Derartige „Leerverschiebungen" am Anfang oder Ende der Schleife nimmt man wegen der Kürze des Programms oft in Kauf.
Wie ersichtlich, muß hier bei M8RLN und M8LRN die Addition Low- und High-Bytes gleichermaßen einbeziehen. Deshalb dauert das Normalverfahren länger als das Tausendeinsverfahren.

Bemerkungen zu M8RLT/M8LRT

Wenn Sie M8LRT genau untersuchen, werden Sie feststellen, daß die erste Linksverschiebung des Produkts in Zeile 29 unnötig ist („Leerverschiebung").
Bei M8LRT tritt in Zeile 37 der BCC-INC-Befehl auf, der bei M8RLT fehlt. Deshalb ist M8RLT nicht nur der kürzeste, sondern auch der schnellste Algorithmus.

Erkennungsmerkmale

Rechtsverfahren erkennen Sie daran, daß der Multiplikator mit LSR oder ROR nach *rechts* verschoben wird. Sinngemäß erkennen Sie Linksverfahren daran, daß der Multiplikator mit ASL oder ROL nach *links* verschoben wird.
Normalverfahren erkennen Sie daran, daß die *Rechts*verschiebung des Multiplikators mit der *Links*verschiebung des Multiplikanden gekoppelt ist und umgekehrt. Sinngemäß erkennen Sie Tausendeinsverfahren daran, daß die *Rechts*verschiebung des Multiplikators mit der *Rechts*verschiebung des Produkts gekoppelt ist und umgekehrt.

M8RLN
Normales Rechtsverfahren (8x8)

```
1              ORG  $0300
2    * M8RLN
3    * ======
4              JMP  S1
5    * Multiplikand Low
6    MDL       HEX  00
7    * Multiplikator Low
8    MRL       HEX  00
9    * Produkt Low-High
10   PL        HEX  00
11   PH        HEX  00
12   * Shift-MD Low-High
13   SMDL      HEX  00
14   SMDH      HEX  00
15   * Shift-MR Low
16   SMRL      HEX  00
17   * Bit-Zähler
18   BITS      HEX  00
19   * 1. Werte initialisieren
20   S1        LDA  #0
21             STA  PL
22             STA  PH
23             LDA  MDL
24             STA  SMDL
25             LDA  #0
26             STA  SMDH
27             LDA  MRL
28             STA  SMRL
29   * 2. Bit-Zähler setzen
30   S2        LDA  #8
31             STA  BITS
32   * 3. SMR rechtsverschieben
33   S3        LSR  SMRL
34             BCC  S5
35   * 4. Wenn C=1, P = P + SMD
36   S4        CLC
37             LDA  PL
38             ADC  SMDL
39             STA  PL
40             LDA  PH
41             ADC  SMDH
42             STA  PH
43   * 5. SMD linksverschieben
44   S5        ASL  SMDL
45             ROL  SMDH
46   * 6. Alle Bits erledigt?
47   S6        DEC  BITS
48             BNE  S3
49             RTS
```

M8LRN
Normales Linksverfahren (8x8)

```
1              ORG  $0300
2    * M8LRN
3    * ======
4              JMP  S1
5    * Multiplikand Low
6    MDL       HEX  00
7    * Multiplikator Low
8    MRL       HEX  00
9    * Produkt Low-High
10   PL        HEX  00
11   PH        HEX  00
12   * Shift-MD Low-High
13   SMDL      HEX  00
14   SMDH      HEX  00
15   * Shift-MR Low
16   SMRL      HEX  00
17   * Bit-Zähler
18   BITS      HEX  00
19   * 1. Werte initialisieren
20   S1        LDA  #0
21             STA  PL
22             STA  PH
23             LDA  MDL
24             STA  SMDH
25             LDA  #0
26             STA  SMDL
27             LDA  MRL
28             STA  SMRL
29   * 2. Bit-Zähler setzen
30   S2        LDA  #8
31             STA  BITS
32   * 3. SMD rechtsverschieben
33   S3        LSR  SMDH
34             ROR  SMDL
35   * 4. SMR linksverschieben
36   S4        ASL  SMRL
37             BCC  S6
38   * 5. Wenn C=1, P = P + SMD
39   S5        CLC
40             LDA  PL
41             ADC  SMDL
42             STA  PL
43             LDA  PH
44             ADC  SMDH
45             STA  PH
46   * 6. Alle Bits erledigt?
47   S6        DEC  BITS
48             BNE  S3
49             RTS
```

M8RLT
Anomales Rechtsverfahren (8x8)

```
1             ORG  $0300
2    * M8RLT
3    * =====
4             JMP  S1
5    * Multiplikand Low
6    MDL      HEX  00
7    * Multiplikator Low
8    MRL      HEX  00
9    * Produkt Low-High
10   PL       HEX  00
11   PH       HEX  00
12   * Shift-MR Low
13   SMRL     HEX  00
14   * Bit-Zähler
15   BITS     HEX  00
16   * 1. Werte initialisieren
17   S1       LDA  #0
18            STA  PL
19            STA  PH
20            LDA  MRL
21            STA  SMRL
22   * 2. Bit-Zähler setzen
23   S2       LDA  #8
24            STA  BITS
25   * 3. SMR rechtsverschieben
26   S3       LSR  SMRL
27            BCC  S5
28   * 4. Wenn C=1, PH = PH + MDL
29   S4       CLC
30            LDA  PH
31            ADC  MDL
32            STA  PH
33   * 5. P rechtsverschieben
34   S5       ROR  PH
35            ROR  PL
36   * 6. Alle Bits erledigt?
37   S6       DEC  BITS
38            BNE  S3
39            RTS
```

M8LRT
Anomales Linksverfahren (8x8)

```
1             ORG  $0300
2    * M8LRT
3    * =====
4             JMP  S1
5    * Multiplikand Low
6    MDL      HEX  00
7    * Multiplikator Low
8    MRL      HEX  00
9    * Produkt Low-High
10   PL       HEX  00
11   PH       HEX  00
12   * Shift-MR Low
13   SMRL     HEX  00
14   * Bit-Zähler
15   BITS     HEX  00
16   * 1. Werte initialisieren
17   S1       LDA  #0
18            STA  PL
19            STA  PH
20            LDA  MRL
21            STA  SMRL
22   * 2. Bit-Zähler setzen
23   S2       LDA  #8
24            STA  BITS
25   * 3. P linksverschieben
26   S3       ASL  PL
27            ROL  PH
28   * 4. SMR linksverschieben
29   S4       ASL  SMRL
30            BCC  S6
31   * 5. Wenn C=1, P = P + MDL
32   S5       CLC
33            LDA  PL
34            ADC  MDL
35            STA  PL
36            BCC  S6
37            INC  PH
38   * 6. Alle Bits erledigt?
39   S6       DEC  BITS
40            BNE  S3
41            RTS
```

Nachfolgend bringen wir noch für die 16-mal-16-Bit-Multiplikation Algorithmen für das normale und das anomale Rechtsverfahren. Die entsprechenden Linksverfahren befinden sich auf der Peeker-Sammeldisk.

M16RLN
Normales Rechtsverfahren (16x16)

```
1             ORG  $0300
2    * M16RLN
3    * ======
4             JMP  S1
5    * Multiplikand High/Low
6    MDH      HEX  00
7    MDL      HEX  00
8    * Multiplikator High/Low
9    MRH      HEX  00
10   MRL      HEX  00
11   * Produkt High bis Low
12   PH       HEX  00
13   PN       HEX  00
14   PM       HEX  00
15   PL       HEX  00
16   * Shift-Multiplikand
17   SMDH     EQU  $00F9
18   SMDN     EQU  $00FA
19   SMDM     EQU  $00FB
20   SMDL     EQU  $00FC
21   * Shift-Multiplikator
22   SMRH     EQU  $00FD
23   SMRL     EQU  $00FE
24   * Bit-Zähler
25   BITS     EQU  $00FF
26   * 1. Werte initialisieren
27   S1       LDY  #3
28            LDA  #0
29   LOESCHE  STA  PH,Y
30            STA  SMDH,Y
31            DEY
32            BPL  LOESCHE
33            LDY  #1
34   KOPIERE  LDA  MRH,Y
35            STA  SMRH,Y
36   * MD.HL nach SMD.ML
37            LDA  MDH,Y
38            STA  SMDM,Y
39            DEY
40            BPL  KOPIERE
41   * 2. Bit-Zähler initialisieren
42   S2       LDA  #16
43            STA  BITS
44   * 3. SMR rechtsverschieben
45   S3       LSR  SMRH
46            ROR  SMRL
47            BCC  S5
48   * 4. Wenn C=1, P = P + SMD
49   S4       CLC
50            LDY  #3
51   ADDIERE  LDA  PH,Y
52            ADC  SMDH,Y
53            STA  PH,Y
54            DEY
55            BPL  ADDIERE
56   * 5. SMD linksverschieben
57   S5       ASL  SMDL
58            ROL  SMDM
59            ROL  SMDN
60            ROL  SMDH
61   * 6. Alle Bits erledigt?
62   S6       DEC  BITS
63            BNE  S3
64            RTS
```

M16RLT
Anomales Rechtsverfahren (16x16)

```
1             ORG  $0300
2    * M16RLT
3    * ======
4             JMP  S1
5    * Multiplikand High/Low
6    MDH      HEX  00
7    MDL      HEX  00
8    * Multiplikator High/Low
9    MRH      HEX  00
10   MRL      HEX  00
11   * Produkt High bis Low
12   PH       HEX  00
13   PN       HEX  00
14   PM       HEX  00
15   PL       HEX  00
16   * Shift-Multiplikator
17   SMRH     EQU  $00FD
18   SMRL     EQU  $00FE
19   * Bit-Zähler
20   BITS     EQU  $00FF
21   * 1. Werte initialisieren
22   S1       LDY  #3
23            LDA  #0
24   LOESCHE  STA  PH,Y
25            DEY
26            BPL  LOESCHE
27            LDY  #1
28   KOPIERE  LDA  MRH,Y
29            STA  SMRH,Y
30            DEY
31            BPL  KOPIERE
32   * 2. Bit-Zähler initialisieren
33   S2       LDA  #16
34            STA  BITS
35   * 3. SMR rechtsverschieben
36   S3       LSR  SMRH
37            ROR  SMRL
38            BCC  S5
39   * 4. Wenn C=1, P.HN = P.HN + MD.HL
40   S4       CLC
41            LDY  #1
42   ADDIERE  LDA  PH,Y
43            ADC  MDH,Y
44            STA  PH,Y
45            DEY
46            BPL  ADDIERE
47   * 5. P rechtsverschieben
48   S5       ROR  PH
49            ROR  PN
50            ROR  PM
51            ROR  PL
52   * 6. Alle Bits erledigt?
53   S6       DEC  BITS
54            BNE  S3
55            RTS
```

Geschwindigkeitsoptimierte 8-mal-8-, 16-mal-8- und 16-mal-16-Bit-Multiplikationsroutinen – allesamt Tausendeins-Rechtsverfahren – finden Sie in dem Aufsatz „Fast 6502 Multiplication“ in „Call A.P.P.L.E.“, Heft 6/1983. Eine Variante der schnellen 16-mal-16-Bit-Multiplikation ist außerdem in meinem „Apple Assembler“, S. 65, abgedruckt.
Die Peeker-Sammeldisk enthält zudem die Programme MULT8RLN.DEMO und MULT16RLN.DEMO, die den Multiplikationsvorgang Schritt für Schritt in Form von Binärzahlen anzeigen und sich deshalb besonders zum Üben eignen.

6. Division

Die Division ist leichter verständlich als die Multiplikation, da exotische Algorithmen in der Art des Tausendeinsverfahrens nicht möglich sind.

6.1. Dezimale Division

Dividieren heißt teilen. Bei dem Ausdruck
22 : 7 = 3 Rest 1
ist die 21 der Dividend DD (= die zu teilende Zahl), die 7 der Divisor DR (= Teiler), die 3 der Quotient Q (= das ganz-

zahlige Ergebnis) und die 1 der Rest R, der bei natürlichen Zahlen dann entsteht, wenn die Division nicht „aufgeht".
Da die Division als Umkehroperation zur Multiplikation verstanden wird, kann man sich von der Richtigkeit einer Division überzeugen, indem man den errechneten Quotienten mit dem Divisor multipliziert und den eventuellen Rest hinzuzählt, also
(7 * 3) + 1 = 22.
Wie bei der Subtraktion können auch bei der Division die Operanden nicht vertauscht werden, denn beispielsweise führt
7 : 2
zu einem anderen Ergebnis als
2 : 7.
Ferner wird in der traditionellen Mathematik die Division durch 0 ausgeschlossen. Es gelten dann – im Hinblick auf die binäre Division – folgende Divisionsregeln:

```
DD : DR = Q Rest R
1  : 1  = 1 Rest 0  (Q = DD)
0  : 1  = 0 Rest 0  (R = DD)
1  : 0  = verboten
0  : 0  = verboten
```

Die Division kann man in eine reine Subtraktion überführen, z.B.
369 : 123 = 3 Rest 0:

```
 369
-123 1mal
-123 2mal
-123 3mal
-----
 000
```

Wenn Dividend und Divisor die gleiche Anzahl der Stellen haben, ist dieses Verfahren empfehlenswert. Für die Aufgabe
98765432 : 2
würde man jedoch bereits einen Karton Papier benötigen.

Intuitives „Herunterholen"

Bei der schriftlichen Division wendet man üblicherweise das „Herunterhol"-Verfahren an:

```
 3150 : 25 = 126
-25''
 ---
  65' (5 herunter)
 -50
 ---
  150' (0 herunter)
 -150
 ----
    0
```

Typisch für dieses Verfahren ist,
a) daß der Divisor zunächst linksbündig unter den Dividenden gesetzt wird und
b) daß im Falle einer nicht erfolgreichen Subtraktion die nächste Stelle *oder auch mehrere gleichzeitig* „heruntergeholt" werden. Dies setzt jedoch voraus, daß man entweder ein guter Kopfrechner ist oder gut „raten" kann.
Wenn man als schlechter Kopfrechner die 24-durch-8-stellige Division
99786968678569686869784 5 : 12367578
nach diesem intuitiven Verfahren zu lösen versucht, begreift man die Unzulänglichkeit des Algorithmus.

Partialdivisionsverfahren

Das „Herunterholen" läßt sich durch die Stellen- oder Partialdivision beweisen. Das Verfahren ist jedoch derart umständlich, daß es als maschinensprachlicher Algorithmus ausscheidet:

```
  (3000 + 0100 + 0050 + 0000)     (3150)
: (0020 + 0005)                 : (  25)
= (0100 + 0020 + 0006)          = ( 126)
=============================================

  (3000 + 0100)  [0100 heruntergeholt]
- (0400 + 0100)  = (0020 + 0005) * 0020
= (2600 + 0000)
---------------------------------------------

  (2600 + 0050)  [0050 heruntergeholt]
- (0200 + 0050)  = (0020 + 0005) * 0010
= (2400 + 0000)
---------------------------------------------

  (2000 + 0400)  [2400 neu zerlegt]
- (1600 + 0400)  = (0020 + 0005) * 0080
= (0400 + 0000)
---------------------------------------------

  (0300 + 0100)  [0400 neu zerlegt]
- (0300 + 0075)  = (0020 + 0005) * 0015
= (0000 + 0025)
---------------------------------------------

  (0020 + 0005)  [0025 neu zerlegt]
- (0020 + 0005)  = (0020 + 0005) * 0001
---------------------------------------------
                                 = 0126
=============================================
```

Genormtes „Herunterholen"

Wenn Dividend und Divisor beide x-stellig sind, genügen jeweils x-stellige Zahlenfelder für den Quotienten und den Rest. Beispiele:
999 : 001 ergibt Q = 999, R = 000.
998 : 999 ergibt Q = 000, R = 998.
Wenn der Dividend (2 * x)-stellig und der Divisor x-stellig sind, genügen ein (2 * x)-stelliges Feld für den Quotienten und ein x-stelliges Feld für den Rest. Beispiele:
9999 : 01 ergibt Q = 9999, R = 00.
0098 : 99 ergibt Q = 0000, R = 98.
Im folgenden beschränken wir uns auf Rechenbeispiele mit 4stelligem Dividenden und Divisor und somit 4stelligem Quotienten und Rest. Bei unserem Einführungsbeispiel müssen wir mithin mit Nullen auffüllen:
3150 : 0025.

Wenn wir vom Kopfrechnen überhaupt keine Ahnung hätten, würden wir den „Herunterhol"-Algorithmus unter Berücksichtigung genormter Zahlenfelder etwa so implementieren:

```
    DD      DR      Q
    3150  :  0025  = xxxx
-------------------------
0000'''          = xxxx
 0003' (3)
-0025
 ----
 0003            = 0xxx
  0031' (1)
 -0025
 ----
 0006            = 01xx
  0065' (5)
 -0025
 -0025
 ----
 0015            = 012x
  0150' (0)
 -0025
 -0025
 -0025
 -0025
 -0025
 -0025
 ----
 0000            = 0126
```

Bei diesem Algorithmus gibt es zwei Besonderheiten:
a) Der Dividend hat eine feste Position, und der Divisor wurde von links nach rechts *unter* dem Dividenden verschoben. Alternative: Der Divisor erhält eine feste Position, und der Dividend wird von rechts nach links *über* dem Divisor verschoben, was auf dasselbe hinausläuft. Der Dividend fungiert dann programmtechnisch als „Shift-Dividend" (Linksverschiebung).
b) Das Einfügen des jeweils errechneten Stellenergebnisses im Quotientenfeld ist algorithmisch* ungeschickt. Alternative: Die Stellenergebnisse werden von rechts nach links in das Quotientenfeld hineingeschoben, was auf dasselbe hinausläuft. Der Quotient fungiert dann programmtechnisch als „Shift-Quotient" (Linksverschiebung).

Genormtes Schiebeverfahren

Unter Berücksichtigung der obigen Modifikationen gelangen wir nunmehr zu einem genormten Schiebeverfahren:

```
  1.Sp  2.Sp  3.Sp
-------------------------------
  R-DR  DD    Q
  0000  3150  xxxx   0.
-------------------------------
< 0003  150x
- 0025        xxx0   1.
= 0003
-------------------------------
< 0031  50xx        2.
- 0025        xx01
= 0006
-------------------------------
< 0065  5xxx        3.
- 0025        x011
- 0025        x012
= 0015
-------------------------------
< 0150  xxxx        4.
- 0025        0121
- 0025        0122
- 0025        0123
- 0025        0124
- 0025        0125
- 0025        0126
= 0000  Rest
```

Spalte 1 enthält den jeweiligen Rest (R) und den Divisor (DR), Spalte 2. den Shift-

* Algorithmisch ungeschickt in bezug auf die 6502-Schiebebefehle, die keine Bit-Einfügung zulassen. Wenn wir mit ROL das Carry-Flag in den Quotienten hineinschieben, führen wir quasi durch die Hintertür das 1001-Verfahren bei der Division ein.

INTUS-Lern- und Anwenderprogramme

für Apple II - Computer

- Rechtschreibtrainer DM 120,–
- Business-English, über 2 500 Wörter DM 120,–
- Maschinenschreiben wie der Blitz DM 175,–
- Basic-Lernprogramm, sehr umfangreich DM 238,–
- Kinderschule, Lernen für Vorschulkinder DM 49,–
- AppleGraph, Erstellen von Kreis- und Balken-Graphiken DM 120,–
- Rechenmodelle für AppleWorks-Rechenblatt DM 195,–
- **NEU: MailWorks für AppleWorks-Serienbriefe, für alle Drucker geeignet. DM 168,–**
- PriBu-Privatbuchhaltung gibt den Überblick DM 195,–
- DMP-Charger zur Gestaltung eigener Zeichensätze auf dem Matrix-Drucker. DM 198,–
- und über 200 weitere Programme. Katalog gratis
- Demo-Disketten mit 5-9 Teilprogrammen DM 10,–
- **6000 Frei-Programme (fast) gratis Programm-Liste (Vorkasse) DM 10,–**

INTUS SOFTWARE

Kaiserstr. 21, 7890 Waldshut,
Tel. 07751-7920

Ausgabe und **Eingabe** mit

TYPETERM®

im Slot Ihres

APPLE II/IIe

Das bedeutet: Computer-textverarbeitung von der Schreibmaschinentastatur! Steckerfertig ohne Umbau.

TYPETERM-Interface **DM 479,–**
für alle BROTHER-Typenrad-schreibmaschinen ab CE-51

Paketpreis: **DM 1887,–**
Schreibmaschine EM-80 mit TYPETERM
CE-68 mit TYPETERM ..... DM 2143,–
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

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

Ausgabe mit

TYPETERM® JUNIOR

im Slot Ihres

APPLE II/IIe

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

brother
QUALITÄT AUS ERSTER HAND

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

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

TYPETERM JUNIOR – ein Produkt von

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

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

Floppy-Controller

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

Erphi-Controller 298,-

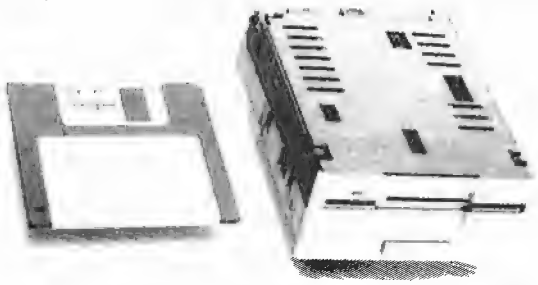
Disketten 1D, 48 tpi ..... 10 St. 29,-
Disketten 2D, 48 tpi ..... 10 St. 36,-

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 498,-

Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur .....



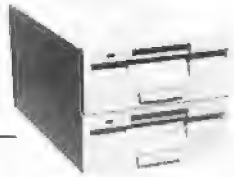
TEAC FD 55 AV 1 x 40 Track	395,-	**TEAC FD 55 BV** 2 x 40 Track	460,-
TEAC FD 55 EV 1 x 80 Track	445,-	**TEAC FD 55 FV** 2 x 80 Track	398,-

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

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

Dividenden (DD) und Spalte 3. den Shift-Quotienten (Q).

0. Block: Wir initialisieren R auf 0000, tragen DD ein und löschen Q. Damit die Schiebetechnik besser auffällt, schreiben wir x statt 0.

1. Block: Wir schieben DD um eine Stelle nach links in R hinein [<–]. Damit kommt die 3 als höchste Stelle von 3150 in die niedrigste Stelle von R. Nunmehr subtrahieren wir
0003 - 0025 = 0 R 0003,
schieben das *Stellen*ergebnis 0 (daher nur 0 und nicht 0000) von rechts in Q hinein und übernehmen 0003 als R in die nächste Zeile.

Block 2: Wir schieben wieder DD um eine Stelle nach links in R hinein. Nunmehr subtrahieren wir
0031 - 0025 = 1 R 0006,
schieben das Stellenergebnis 1 von rechts in Q hinein und übernehmen 0006 als R in die nächste Zeile.

Block 3: Wir schieben wieder DD um eine Stelle nach links in R hinein. Nunmehr subtrahieren wir
0065 - 0025 - 0025 = 2 R 0015.
Wie ersichtlich, muß die 0025 hier zweimal subtrahiert werden. Bei Dezimalzahlen muß 0-9mal, bei Binärzahlen nur 0-1mal subtrahiert werden. Auch wenn wir mehrfach subtrahieren müssen, ist das Ergebnis trotzdem stets einstellig. Deshalb schieben wir auch hier das Stellenergebnis 2 von rechts in Q hinein und übernehmen 0015 als R in die nächste Zeile.

Block 4: Auch hier ist die Prozedur wieder dieselbe, wobei allerdings nunmehr das Stellenergebnis 6 beträgt. R ist nachher 0000, und damit geht die Division insgesamt ohne Rest auf.
Selbst wenn wir, z.B. bei der Aufgabe
3151 : 0025 = 0126 R 0001,
einen Rest vorgefunden hätten, wäre die Division jetzt beendet, weil der Dividend bereits 4mal nach links geschoben worden ist.

Algorithmus in Kurzform
Schritt 1: Rest und Quotienten auf 0 setzen.
Schritt 2: Dividenden von rechts nach links um eine Stelle in den Rest hineinschieben.
Schritt 3: Subtraktion von Rest minus Divisor durchführen. Stellenergebnis der Subtraktion von rechts nach links in den Quotienten hineinschieben sowie neuen Rest eintragen.
Schritt 4: Prüfen, ob bereits sooft linksverschoben worden ist, wie der Dividend Stellen hat. Wenn nein, dann zurück zu Schritt 2. Wenn ja, dann Ende.

6.2. Binäre Division

Die binäre Division läßt sich haargenau wie die genormte dezimale Division durchführen. Es gibt nur eine kleine und zugleich erfreuliche Besonderheit: Das Stellenergebnis ist höchstens 1, und deshalb muß der Divisor vom jeweiligen Rest auch höchstens ein einziges Mal subtrahiert werden. Eine x-durch-x-Bit-Division besteht damit aus exakt

x Linksverschiebungen von DD und R
x Subtraktionen von R und DR
x Linksverschiebungen von Q,
wobei gleichzeitig entweder 0 oder 1 als Stellenergebnis von rechts in Q hineingeschoben wird.

Könnte das Stellenergebnis nicht doch einmal größer als 1 sein? Betrachten wir hierzu die nachstehende Division:

```
  11 :    7 =    1 R    4
1011 : 0111 = 0001 R 0100
-------------------------
  R-DR  DD    Q
  0000  1011  xxxx   0.
-------------------------
< 0001  011x        1.
- 0111        xxx0
= 0001
-------------------------
< 0010  11xx        2.
- 0111        xx00
= 0010
-------------------------
< 0101  1xxx        3.
- 0111
= 0101        x000
-------------------------
< 1011  xxxx        4.
- 0111        0001
= 0100  Rest
```

Wir haben also 4mal schieben müssen, bevor eine Subtraktion, nämlich die letzte, „aufging". Wenn Sie ähnliche Beispiele ausprobieren, werden Sie feststellen:
Die höchste Einserbit-Stelle des momentanen Rests kann die höchste Einserbit-Stelle des Divisors höchstens um eine einzige Stelle überragen, weil die Division durch 0 verboten ist und somit der Divisor stets größer als 1 ist. Wenn dann der Divisor vom momentanen Rest abgezogen werden kann, wird die besagte Einserbit-Stelle beim neuen Rest eliminiert. Wenn umgekehrt der Divisor nicht vom momentanen Rest abgezogen werden kann, dann muß es sich um den Rest nach der letzten Linksverschiebung handeln. Damit ist jedoch gleichzeitig die Division beendet.

Nachfolgend bringen wir ein komplizierteres Beispiel
dezimal 65535 : 17 = 3855 Rest 0
für eine Binärdivision, wobei wir die Zahlenkolonnen nach der 6502-Division in Abschnitt 6.3 anordnen. Es bedeuten
R = Rest
S = Shift-Dividend
D = Divisor
Q = Quotient
↑ = Es wurde 1mal subtrahiert.
Wenn der Rest kleiner als der Divisor ist, wird die (versuchte) Subtraktion nicht angezeigt und der Rest sofort weiterverschoben.

```
R 00000000 00000000 S 11111111 11111111
-----------------------------------------
R 00000000 00000001 S 11111111 11111110
D 00000000 00010001 Q xxxxxxxx xxxxxxx0
R 00000000 00000011 S 11111111 11111100
D 00000000 00010001 Q xxxxxxxx xxxxxx00
R 00000000 00000111 S 11111111 11111000
D 00000000 00010001 Q xxxxxxxx xxxxx000
R 00000000 00001111 S 11111111 11110000
D 00000000 00010001 Q xxxxxxxx xxxx0000
R 00000000 00011111 S 11111111 11100000
D 00000000 00010001 Q xxxxxxxx xxx00001
= 00000000 00001110                   ↑
R 00000000 00011101 S 11111111 11000000
D 00000000 00010001 Q xxxxxxxx xx000011
= 00000000 00001100                   ↑
R 00000000 00011001 S 11111111 10000000
D 00000000 00010001 Q xxxxxxxx x0000111
= 00000000 00001000                   ↑
R 00000000 00010001 S 11111111 00000000
D 00000000 00010001 Q xxxxxxxx 00001111
= 00000000 00000000                   ↑
R 00000000 00000001 S 11111110 00000000
D 00000000 00010001 Q xxxxxxx0 00011110
R 00000000 00000011 S 11111100 00000000
D 00000000 00010001 Q xxxxxx00 00111100
R 00000000 00000111 S 11111000 00000000
D 00000000 00010001 Q xxxxx000 01111000
R 00000000 00001111 S 11110000 00000000
D 00000000 00010001 Q xxxx0000 11110000
R 00000000 00011111 S 11100000 00000000
D 00000000 00010001 Q xxx00001 11100001
= 00000000 00001110                   ↑
R 00000000 00011101 S 11000000 00000000
D 00000000 00010001 Q xx000011 11000011
= 00000000 00001100                   ↑
R 00000000 00011001 S 10000000 00000000
D 00000000 00010001 Q x0000111 10000111
= 00000000 00001000                   ↑
R 00000000 00010001 S 00000000 00000000
D 00000000 00010001 Q 00001111 00001111
= 00000000 00000000                   ↑
```

6.3. 6502-Division

Bei der nachfolgenden Version 1 einer 16-durch-16-Bit-Division (D16V1) lassen sich folgende Schritte hervorheben:

Schritt 1: Quotient und Rest werden auf 0 gesetzt und der Dividend wird in einen Shift-Dividenden dupliziert, damit der ursprünglich Dividend erhalten bleibt.
Schritt 2: Vorab wird geprüft, ob der Divisor 0 ist. Wenn ja, wird die Routine sofort verlassen.
Schritt 3: Da der Dividend 16 Bits umfaßt, wird der Bit-Zähler auf 16 gesetzt. Der Bit-Zähler richtet sich nach dem Dividenden und wäre deshalb bei einer 16-durch-8-Bit-Division ebenfalls 16.
Schritt 4: Der Shift-Dividend wird von rechts nach links in den Rest hineingeschoben.
Schritt 5: Dann wird geprüft, ob die Subtraktion von Rest minus Divisor ohne Überlauf möglich ist. Im Falle eines Überlaufs ist C = 0, sonst C = 1.
Schritt 6: Wenn ja, wird die Differenz von Rest und Divisor als neuer Rest gespeichert.

Schritt 7: Das Ergebnis der erfolgreichen (C = 1) oder erfolglosen (C = 0) Subtraktion wird von rechts in den Quotienten hineingeschoben.
Schritt 8: Der Bit-Zähler wird dekrementiert und auf 0 überprüft. Wenn noch nicht alle 16 Bits abgearbeitet worden sind, kehrt die Programmschleife zu Schritt 4 zurück.

D16 V1

```
1             ORG  $0300
2     *
3     * D16 Version 1
4     * ===
5             JMP  S1
6     * Dividend
7     DDL     HEX  FF
8     DDH     HEX  FF
9     * Divisor
10    DRL     HEX  0F
11    DRH     HEX  00
12    * Quotient
13    QL      HEX  00
14    QH      HEX  00
15    * Rest
16    RL      HEX  00
17    RH      HEX  00
18    * Shift-Dividend
19    SDDL    HEX  00
20    SDDH    HEX  00
21    * Bit-Zähler
22    BITS    HEX  00
23    * 1. Q = 0, R = 0, SDD = DD
24    S1      LDX  #1
25    S1A     LDA  #0
26            STA  QL,X
27            STA  RL,X
28            LDA  DDL,X
29            STA  SDDL,X
30            DEX
31            BPL  S1A
32    * 2. DR = 0?
33    S2      LDA  DRL
34            ORA  DRH
35            BEQ  S9
36    * 3. Bit-Zähler auf 16
37    S3      LDA  #16
38            STA  BITS
39    * 4.  <-- SDDH-SDDL-0 links
40    S4      ASL  SDDL
41            ROL  SDDH
42    * 4A. <-- RH-RL-C links
43    S4A     ROL  RL
44            ROL  RH
45    * 5. R - DR >= 0?
46    S5      SEC
47            LDA  RL
48            SBC  DRL
49            TAX
50            LDA  RH
51            SBC  DRH
52            BCC  S7          ;C=0!
53    * 6. Wenn ja, R = R - DR
54    S6      STX  RL          ;C=1!
55            STA  RH
56    * 7. und <-- QH-QL-C links
57    S7      ROL  QL
58            ROL  QH
59    * 8. Bit-Zähler = 0?
60    S8      DEC  BITS
61            BNE  S4
62    * 9. Ende (auch bei DR=0)
63    S9      RTS
```

Die analoge Version 2 (D16V2) überträgt den Dividenden nicht in einen gesonderten Shift-Dividenden, sondern in den Quotienten. Dies ist möglich, weil mit jeder Linksverschiebung des „Quotienten" = Shift-Dividenden das rechte, nullte Bit frei wird und somit das Carry-Flag der erfolgreichen/erfolglosen Subtraktion aufnehmen kann. Wegen des INC-Befehls in Zeile 53 ist Version 2 geringfügig langsamer. Die Peeker-Sammeldisk enthält außerdem die „getunten" Versionen 3 (D16V3) und 4 (D16V4), von denen Version 4 ca. 25% schneller ist. Das zusätzliche Programm D24V2 ist eine 24-durch-24-Bit-Division nach dem Algorithmus von Version 2. Ferner sind auf die Sammeldisk die Demos DIV16.DEMO und DIV24.DEMO aufgenommen worden, die den internen Rechenprozeß Schritt für Schritt in Form von Binärzahlen anzeigen und sich deshalb besonders für die Schule eignen.

D16 V2

```
1             ORG  $0300
2     *
3     * D16 Version 2
4     * ===
5             JMP  S1
6     * Dividend
7     DDL     HEX  FF
8     DDH     HEX  FF
9     * Divisor
10    DRL     HEX  0F
11    DRH     HEX  00
12    * Quotient
13    QL      HEX  00
14    QH      HEX  00
15    * Rest
16    RL      HEX  00
17    RH      HEX  00
18    * Bit-Zähler
19    BITS    HEX  00
20    * 1. R = 0, Q = DD
21    S1      LDX  #1
22    S1A     LDA  #0
23            STA  RL,X
24            LDA  DDL,X
25            STA  QL,X
26            DEX
27            BPL  S1A
28    * 2. DR = 0?
29    S2      LDA  DRL
30            ORA  DRH
31            BEQ  S9
32    * 3. Bit-Zähler auf 16
33    S3      LDA  #16
34            STA  BITS
35    * 4.  <-- QH-QL-0 links
36    S4      ASL  QL
37            ROL  QH
38    * 4A. <-- RH-RL-C links
39    S4A     ROL  RL
40            ROL  RH
41    * 5. R - DR >= 0?
42    S5      SEC
43            LDA  RL
44            SBC  DRL
45            TAX
46            LDA  RH
47            SBC  DRH
48            BCC  S8
49    * 6. Wenn ja, R = R - DR
50    S6      STX  RL
51            STA  RH
52    * 7. und QL = QL + 1
53    S7      INC  QL
54    * 8. Bit-Zähler = 0?
55    S8      DEC  BITS
56            BNE  S4
57    * 9. Ende (auch bei DR=0)
58    S9      RTS
```

Peeker-Sammeldisk

Auf der Sammeldisk #14 befinden sich diverse Assemblerprogramme (mit Big-Mac-Quelltexten) und Demos zu diesem Beitrag, wobei die mit RUN gekennzeichneten Applesoft-Demos die jeweils nachfolgenden Maschinenprogramme starten:

Multiplikationsprogramme
M8TEST (RUN)
T.M8RLN und M8RLN
T.M8LRN und M8LRN
T.M8RLT und M8RLT
T.M8LRT und M8LRT
M16TEST (RUN)
T.M16RLN und M16RLN
T.M16LRN und M16LRN
T.M16RLT und M16RLT
T.M16LRT und M16LRT
MULT8RLN.DEMO (RUN)
T.MULT8RLN und MULT8RLN
MULT16RLN.DEMO (RUN)
T.MULT16RLN und MULT16RLN
Divisionsprogramme
D16TEST (RUN)
T.D16V1 und D16V1
T.D16V2 und D16V2
T.D16V3 und D16V3
T.D16V4 und D16V4
D24TEST (RUN)
T.D24V2 und D24V2
DIV16.DEMO (RUN)
T.DIV16 und DIV16
DIV24.DEMO (RUN)
T.DIV24 und DIV24

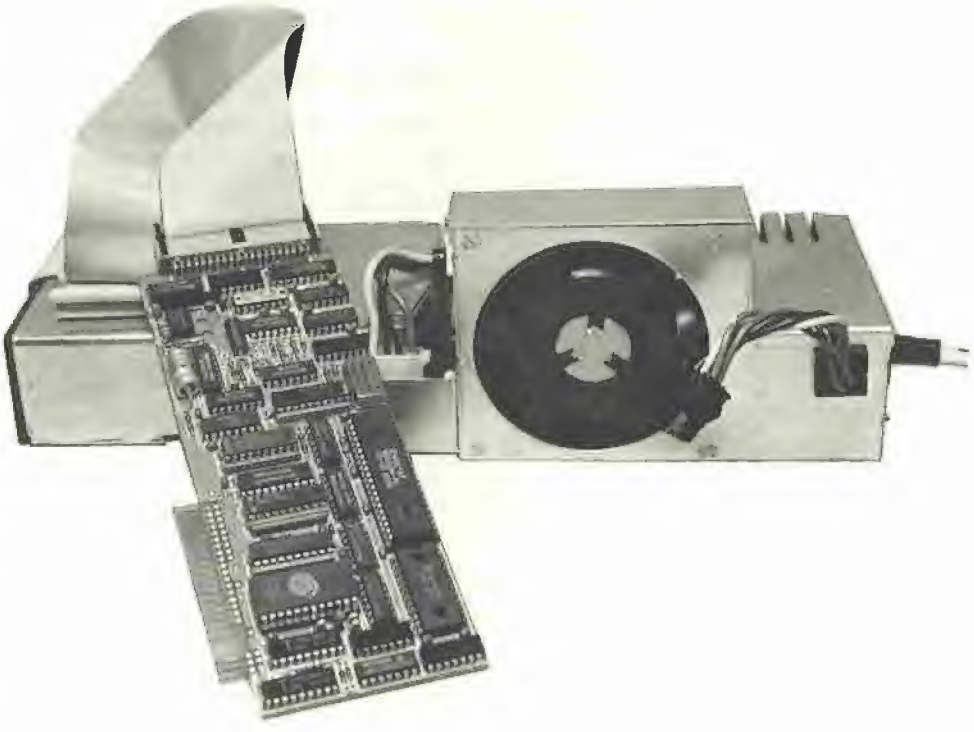
Ein intelligenter Harddisk-Controller

Aufbau und Arbeitsweise des Megaboards

von Dr.-Ing. T. Mulica

1. Einführung

Die optimale Ausnutzung der Möglichkeiten eines Mikrocomputers ist im wesentlichen von den angeschlossenen peripheren Geräten abhängig. Die externen Speichermedien des Rechners spielen dabei eine wichtige Rolle. Es ist unschwer einzusehen, daß die Verwendung eines Floppy-Disk-Laufwerkes anstelle eines Kassettenrecorders für die kleinen Rechner ganz neue Anwendungsgebiete eröffnet hat.
Mit dem Einzug der Mikrocomputer-Software in immer speicherintensivere Anwendungsgebiete, z.B. Textverarbeitung, Datenbanksysteme oder Grafik, sind die Anforderungen an die Speicherkapazität und die Zugriffsgeschwindigkeit der externen Speichermedien ständig gestiegen. Um diesen gewachsenen Bedürfnissen Rechnung zu tragen, wurde für die Mikrocomputer in den letzten Jahren eine neue Generation von Massenspeichern entwikkelt. Es handelt sich dabei um die Festplattenlaufwerke (Winchester-Laufwerke, Harddisk-Laufwerke), die dem Benutzer einen sicheren, komfortablen und vor allem schnellen Zugriff zu den abgespeicherten Datenmengen gewährleisten.



Bei vielen in den letzten Jahren entwickelten Mikrocomputern gehört die Festplatte bereits zur Standardausrüstung. Aber auch die „älteren" Rechner lassen sich mit der Festplatte nachrüsten. Diese Nachrüstung eröffnet dem Benutzer einen besseren und komfortablen Einsatz der am Markt eingeführten Standard-Software oder eigener Programme.
Besonders schwierig war die Lage der Apple-Besitzer mit den weitverbreiteten Geräten Apple II, II+, IIe und Kompatiblen. Der apple-eigene Standard für den Anschluß von Floppy-Disk-Laufwerken und das eigene Datenaufzeichnungsformat erschwerte selbst den einfachen Anschluß von Laufwerken mit einer größeren Speicherkapazität. Es gibt zwar zur Zeit die Möglichkeit, durch den Kauf eines speziellen Controllers (z.B. von den Firmen Erphi oder Ehring) Nicht-Apple-Standard-Laufwerke anzuschließen, jedoch bringt diese Lösung keine Verbesserung der Zugriffzeiten.
Um Apple-Benutzern auch den Anschluß eines Winchester-Laufwerkes an ihren Rechner zu ermöglichen, wurde auf dem

deutschen Markt von der Firma „Frank & Britting“ der intelligente Apple-Harddisk-Controller „Megaboard“ entwickelt. Dieser Controller kann direkt an den Apple-Bus angeschlossen werden und erlaubt dem Anwender, den Speicherplatz des an den Controller angeschlossenen Laufwerks zwischen den vier gängigsten Apple-Betriebssystemen DOS 3.3, CP/M 2.2, UCSD-Pascal 1.1 und ProDOS zu teilen. Die Firmware des Controllers enthält alle Treiberroutinen, die für die Anpassung der o.g. Betriebssysteme an das Harddisk-System nötig sind. Das „Einbinden“ dieser Routinen in das Originalbetriebssystem erfolgt automatisch während des Boot-Vorgangs. Dieser Boot-Vorgang von der Harddisk wird durch ein spezielles Firmware-Programm gesteuert, das auch gleichzeitig die notwendigen Modifikationen („Patches“) des betroffenen Betriebssystems durchführt. An das Megaboard können verschiedene Harddisk-Laufwerke angeschlossen werden, die über eine Standard-ST-506-Schnittstelle sowie bis maximal 8 Köpfe und bis maximal 1024 Zylinder verfügen.
Die interessantesten Aspekte, die bei der Hardware- und Software-Entwicklung des Apple-Harddisk-Controllers auftraten, sind das Hauptthema dieses Beitrags.

2. Wichtigste Anforderungen

An die Entwicklung des Harddisk-Controllers wurden folgende Anforderungen gestellt:

1. Der Benutzer kann nach eigenem Wunsch den zur Verfügung stehenden Speicherplatz der Harddisk auf die vier Betriebssysteme (DOS 3.3, CP/M 2.2, UCSD-Pascal 1.1, ProDOS) verteilen.

2. An den Controller können Laufwerke mit verschiedenen Speicherkapazitäten angeschlossen werden.

3. Die Umschaltung zwischen den verschiedenen Betriebssystemen erfolgt auf Software-Basis.

4. Die Anpassung des jeweiligen Betriebssystems an die neue Umgebung (Harddisk-System) erfolgt automatisch beim Boot-Vorgang, so daß das Boot-Volume immer eine Originalversion des Betriebssystems enthält.

5. Die Betriebssystem-Änderungen und -Erweiterungen sollen keinen Einfluß auf die Größe des Arbeitsspeichers des Rechners haben.

6. Bei jedem Betriebsssystem ist es möglich, einen Mischbetrieb Floppy-Disk/Harddisk durchzuführen.

Die Entwicklung eines solchen Controllers erfordert die Realisierung der dazu benötigten Funktionen durch Hard- und Software-Komponenten.

3. Hardware-Aufbau

Die komplette Hardware des Controllers wurde auf einer Platine untergebracht, die einen Apple-Slot belegt. Den prinzipiellen Aufbau des Controllers zeigt **Abb. 1**.

Der Anschluß an den Rechner erfolgt über einen 50poligen direkten Stecker (1). Der Controller belegt den für Slot 7 reservierten Adreßbereich $C0F0-$C0FF und $C700-$C7FF sowie zusätzlich den sog. „I/O-Expansion-Adreßbereich“ $C800-$CFFF. Die entsprechende Hardware-Schaltung zum Ein- und Ausblenden dieses eventuell auch von anderen Peripheriekarten verwendeten Zusatzbereiches ist vorhanden. Der Adreßbereich $C700-$CFFF wurde zwischen Firmware-EPROM (2) und Scratch-RAM (3) wie folgt aufgeteilt:
$C700-$C7FF – EPROM
$C800-$C9FF – RAM
$CA00-$CFFF – EPROM
Um den nötigen Speicherplatz für die Controller-Firmware zur Verfügung zu

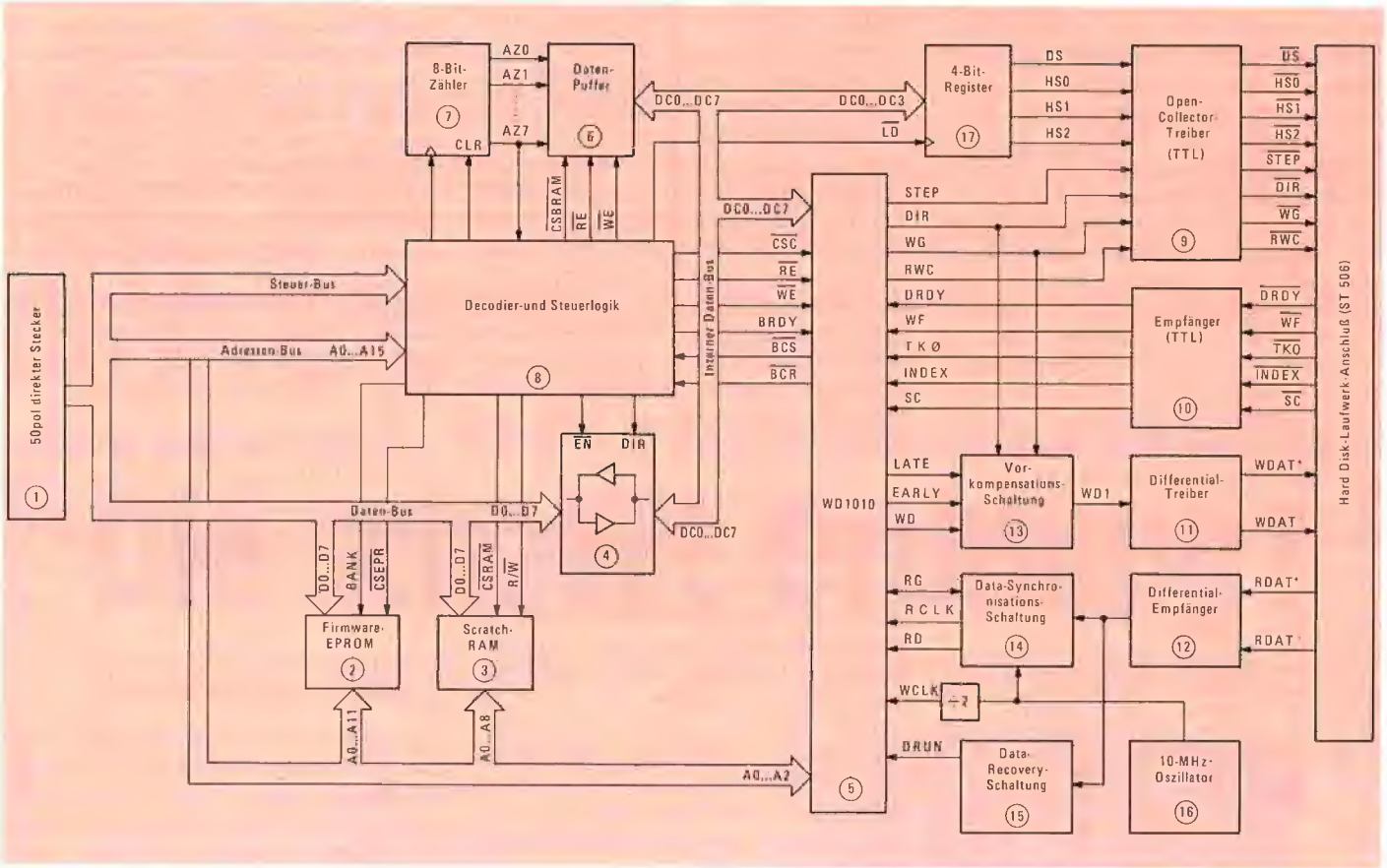


Abb. 1: Blockschaltbild des Apple-Harddisk-Controllers (die durch Kreise bezeichneten Ziffern sind im Text durch runde Klammern gekennzeichnet)

stellen, wird der Speicherbereich $CA00-$CFFF in zwei Bänke aufgeteilt. Die Auswahl der entsprechenden Bänke erfolgt durch das Lesen von zwei Adressen $C0F2 (Bank 0) und $C0F3 (Bank 1).
Die Hauptaufgabe der Controller-Firmware ist es, eine Software-Schnittstelle zu schaffen, die den Zugriff der Apple-Betriebssysteme auf der Harddisk ermöglicht. Die von der Firmware benötigten Daten werden im Controller-RAM (3) abgespeichert.
Wegen der hohen Datenübertragungsgeschwindigkeit zwischen der Harddisk und dem Controller (5 MBits/s) müssen fast alle Funktionen, die diesen Übertragungsprozeß steuern, durch die Hardware übernommen werden. Um trotzdem eine kompakte Lösung des Controllers zu erreichen, wurde der hochintegrierte Harddisk-Controller-Chip WD1010 (5) von Western Digital eingesetzt. Dieser Baustein übernimmt die meisten Funktionen, die für die Harddisk-Ansteuerung sowie die Datensicherung auf der Platte notwendig sind. Das Interface zwischen dem Host-Prozessor und dem WD1010 besteht aus dem 8 Bit breiten Datenbus, über den die Daten, Parameter und Status-Informationen übertragen werden. Die Kommunikation zwischen dem Rechner und dem Harddisk-Controller erfolgt über 8 Register, die 8 Adressen ($C0F8-$C0FF) im Speicherraum des Prozessors belegen. Diese Register sind wie folgt definiert:

	Lesen	Schreiben
$C0F8	Data Register	Data Register
$C0F9	Error Register	Write Precompensation
$C0FA	Sector Count	Sector Count
$C0FB	Sector Number	Sector Number
$C0FC	Cylinder Low	Cylinder Low
$C0FD	Cylinder High	Cylinder High
$C0FE	SDH Register	SDH Register
$C0FF	Status Register	Command Register

Die genaue Beschreibung dieser Register sowie die Programmierungserläuterungen findet der Leser in <1>.
Die Programmierung des Controllers wird durch die leistungsfähigen Befehle des WD1010 vereinfacht. Soll der Controller z.B. einen Datensektor (256 Bytes) auf die Platte schreiben, so muß man zuerst die entsprechenden Sektor-, Zylinder- und Kopfnummern spezifizieren und diese Informationen durch die Beschreibung des Sektor-, Zylinder- und SDH-Registers an den Controller übergeben. Das Command Register muß mit dem Schreibbefehl geladen werden (Code $30). Der Rechner kann jetzt durch das Lesen des Status-Registers feststellen (DRQ Bit im Status-Register gesetzt), daß der Controller auf die Datenübertragung zwischen dem Betriebssystem-Datenpuffer und dem internen Controller-Datenpuffer wartet. Der Controller-Datenpuffer besteht aus einem statischen RAM (6), dessen Adressen durch einen Zähler (7) erzeugt werden. Dadurch erscheint dieses RAM nach außen als FIFO-Speicher. Der Zugriff des Rechners auf den Controller-Datenpuffer erfolgt über das Daten-Register ($C0F8).

Hat der Rechner den Datenpuffer vollgeschrieben, so wird durch die Steuerungslogik (8) ein Signal (BRDY) erzeugt, das den WD1010 über dieses Ereignis informiert. Der WD1010 fängt automatisch an, die Daten aus dem Datenpuffer zu lesen und in entsprechend kodierter Form (MFM, s. Peeker, Heft 3/85, S. 10) auf die Harddisk zu übertragen. Die Umwandlung der im Datenpuffer abgespeicherten Daten in den MFM-Datenstrom sowie CRC- und Daten-Adreßmarkengenerierung findet in dem Chip statt. Während dieses Vorgangs muß der interne Datenbus des Controllers von dem Host-Datenbus isoliert werden. Dazu ist auf der Platine ein Daten-Treiber (4) vorgesehen, der sich in diesem Zeitbereich in einem hochohmigen Zustand befindet. Das durch den WD1010-Chip erzeugte Signal (BCS) informiert die Steuerungslogik darüber, wann der interne Datenbus des Controllers isoliert sein muß.

Solange der Controller mit der Ausführung des Befehls beschäftigt ist, wird das Busy Bit (Bit 7) im Status-Register gesetzt. Nach der Beendigung jedes Befehls wird dieses Bit zurückgesetzt. Die restlichen Bits des Status-Registers informieren den Rechner in diesem Fall über den Zustand des Laufwerks und über den Verlauf der Befehlsausführung. Sollte bei der Befehlsausführung ein Fehler aufgetreten sein, so wird das letzte Bit (Bit 0) des Status-Registers gesetzt. Die Art des Fehlers kann der Rechner durch das Lesen des Fehlerregisters erkennen.

Die von dem WD1010-Chip benötigte externe Logik auf der Laufwerksseite kann man in vier Teile gliedern:

– Treiber und Empfänger für Laufwerks-Steuer-, Status- und Daten-Signale (9, 10, 11, 12).
– Vorkompensationsschaltung (13)
– Daten-Synchronisationsschaltung (14)
– Data-Recovery-Schaltung (15)

– Die Aufgabe der Treiber und Empfänger ist es, die nötigen Signale zwischen dem Controller und dem Laufwerk aneinander anzupassen (z.B. muß das TTL-Datensignal WD1 durch einen Differential-Treiber (11) in ein Differential-Signal umgewandelt werden).

– Die Vorkompensationsschaltung wird von dem WD1010-Chip durch die vier Signale RWC, WG, EARLY und LATE gesteuert. Sie verschiebt den von dem WD1010 erzeugten Datenstrom gegenüber der Nominalposition, je nach dem Datenmuster und der Position des Schreib/Lesekopfes. Diese Verschiebung ist besonders auf den inneren Spuren des Laufwerks nötig. Der Benutzer kann jederzeit bei der Harddisk-Initialisierung eingeben, ab welchem Zylinder die Vorkompensation stattfinden soll.

– Die Aufgabe der Daten-Synchronisationsschaltung ist es, einen Lese-Takt (RCLK) für den WD1010 zu erzeugen, der mit den MFM-Rohdatenimpulsen, die bei einer Leseoperation von der Festplatte kommen, synchronisiert ist. Diese Schaltung wird durch ein RG-Signal von dem WD1010 gesteuert.
Die eigentliche Decodierung und Umsetzung der MFM-Daten in parallele Form sowie die Überprüfung der CRC-Bytes findet in dem WD1010 statt. Der Chip erzeugt auch die entsprechenden Signale, die den Datentransfer zwischen dem Datenpuffer und dem Chip ohne jeglichen Eingriff des Rechners ermöglichen.

– Die Data-Recovery-Schaltung informiert den WD1010-Chip über das Auftreten eines Nullmusters im MFM-Datenstrom. Entdeckt diese Schaltung ein Nullmuster, so aktiviert sie das WD1010-Eingangssignal (DRUN).

4. Software-Komponenten

Nachdem im vorherigen Abschnitt in erster Linie über den Hardware-Aufbau des Harddisk-Controllers gesprochen wurde, wollen wir nun die für den Einsatz des Controllers notwendige Software erläutern. Bevor man Daten mit dem Harddisk-System verarbeiten kann, muß das Harddisk-Laufwerk für die Datensicherung vorbereitet und die ursprünglichen Betriebssysteme modifiziert, d.h. um einige harddisk-spezifische Programmteile erweitert werden. Die dazu notwendige Software kann man grob gesehen in zwei Pakete teilen:
1. Das Software-Paket, das die Vorbereitung des Harddisk-Laufwerks für die Datensicherung unterstützt.
2. Das Software-Paket, das ständig im EPROM des Controllers vorhanden ist (Firmware).

4.1. Vorbereitung der Harddisk

Der Vorbereitungsprozeß sollte grundsätzlich nur bei der allerersten Inbetrieb-

nahme des Harddisk-Systems durchgeführt werden. Dieser Prozeß besteht aus drei Stufen:

a) Initialisierung der Harddisk;
b) Konfiguration der Harddisk;
c) Übertragung der Betriebssysteme auf die Harddisk.

Der Initialisierungsvorgang hat folgende Aufgaben:

– Die Parameter des Laufwerks (z.B. Anzahl der Köpfe, Anzahl der Zylinder usw.), die für die spätere Speicherverwaltung benötigt werden, sind zu erfassen. Diese Parameter müssen vom Benutzer eingegeben werden.
– Alle Spuren der Harddisk müssen formatiert werden. Die Spuren werden dabei in Sektoren aufgeteilt, wobei jeder Sektor aus einem Identifikationsfeld (ID-Feld) und einem 256 Bytes langen Datenfeld besteht.
– Dann wird überprüft, ob die während des Formatierungsvorgangs auf die Harddisk geschriebene Information lesbar ist.
– Für die Spuren, die aus irgendeinem Grund beschädigt und deshalb unlesbar sind, werden Ersatzspuren erstellt. Diese Ersatzspuren befinden sich natürlich an einer anderen (vom System reservierten) Position auf der Platte. Um diese Ersatzspuren bei Bedarf wiederzufinden, wird eine Tabelle (Ersatz-Track-Tabelle) erstellt und auf der Harddisk abgespeichert. Sollte die ganze Platte mehr als 32 defekte Stellen haben, läßt sich das Harddisk-Laufwerk nicht initialisieren.
– Einrichten eines DOS-Volumes auf der Harddisk. Auf diesem Volume werden die Hilfsprogramme abgespeichert, die für die weitere Arbeit mit der Harddisk notwendig sind.

Alle Programme, die den Initialisierungsvorgang unterstützen, sowie weitere Hilfsprogramme befinden sich auf einer DOS-Diskette, die zusammen mit dem Controller geliefert wird.

Nach Beendigung des Initialisierungsvorgangs kann man die Harddisk booten und ein Hilfsprogramm starten, das die Konfiguration der Harddisk ermöglicht. Während dieses Vorgangs kann der Benutzer den auf der Harddisk verfügbaren Speicherplatz den vier implementierten Betriebssystemen zuordnen. Außerdem kann festgelegt werden, welches Betriebssystem beim Einschalten des Rechners booten soll. Die vom Benutzer definierte Speicheraufteilung wird durch das Konfigurationsprogramm in einer Tabelle (Organisationstabelle) erfaßt und am Ende der Konfiguration auf der Harddisk abgelegt. Bei dem Aufteilungsprozeß müssen selbstverständlich die spezifischen Eigenschaften des jeweiligen Betriebssystems berücksichtigt werden (s.u.).

Die letzte Stufe des Vorbereitungsprozesses wird auch teilweise durch das Konfigurationsprogramm unterstützt. Hier werden die Betriebssystemdisketten auf die entsprechenden Harddisk-Bereiche kopiert. Die Struktur des Konfigurationsprogramms zeigt **Abb. 2**.

Eine denkbare Harddisk-Speicherorganisation nach der Konfiguration ist in **Abb. 3** dargestellt.

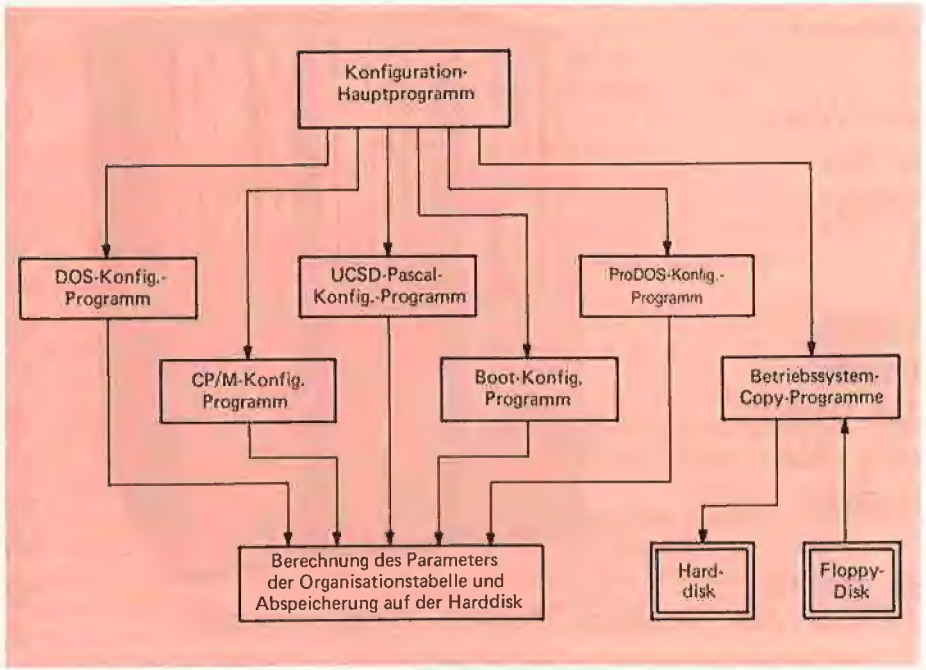


Abb. 2: Struktur des Konfigurationsprogramms

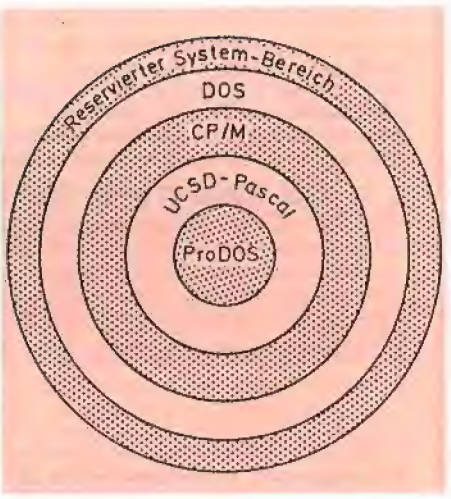


Abb. 3: Mögliche Speicherorganisation der Harddisk

Während des Vorbereitungsprozesses werden auf der Harddisk wichtige Informationen über die Charakteristik und Speicherorganisation der Platte abgespeichert. Für diese Informationen wurden einige Spuren reserviert, die in Zukunft für den Benutzer unzugänglich sind, jedoch später durch die Treiberroutinen des Controllers ständig gebraucht werden.
Nachdem am Ende des Konfigurationsvorgangs die im Grunde genommen „floppy-orientierten" Betriebssysteme auf die Harddisk übertragen worden sind, wollen wir jetzt die Software-Teile besprechen, die für die Anpassung der Betriebssysteme an die neue „Harddisk-Umgebung" verantwortlich sind. Diese Routinen sind im EPROM des Controllers abgespeichert und stehen dem Rechner jederzeit zur Verfügung.

4.2. Firmware-Routinen

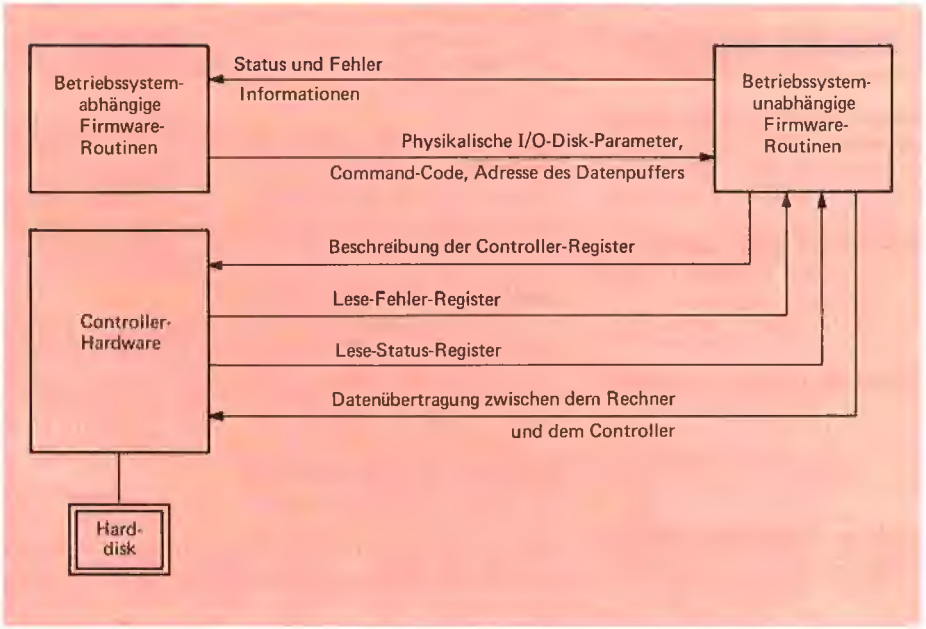
Die Betriebssysteme DOS, CP/M, UCSD-Pascal und ProDOS wurden im Apple unter ganz bestimmten Voraussetzungen bezüglich des Formats der externen Speichermedien implementiert. Aus diesem Grund erfordert jeder Anschluß von externen Datenträgern, die nicht dem Apple-Standard entsprechen, eine Reihe von Änderungen und Ergänzungen der Betriebssysteme. Ein solcher Fall liegt insbesondere beim Anschluß eines Harddisk-Laufwerks vor.
Um die Hard/Software-Schnittstelle des Harddisk-Controllers zu bedienen, müssen für jedes Betriebssystem neue Hardware-Treiberroutinen geschrieben werden. Wegen der größeren Speicherkapazität der Harddisk ist es auch erforderlich, die Dateiverwaltung der Betriebssysteme zu modifizieren, damit von dem erweiterten Speicherangebot Gebrauch gemacht werden kannn. Zusätzlich muß die Tatsache, daß der Speicherplatz auf dem Hard-

disk-Laufwerk auf vier verschiedene Betriebssysteme verteilt sein kann, beim Umrechnungsalgorithmus der logischen I/O-Disk-Parameter (z.B. Block-Nummer bei UCSD-Pascal) in die physikalischen Parameter (z.B. Sektor-, Zylinder- und Kopf-Nummer), die zur Ansteuerung des Controllers notwendig sind, berücksichtigt werden.

Bei all diesen Eingriffen in das jeweilige Betriebssystem ist unter anderem darauf zu achten, daß der für die Anwenderprogramme nutzbare RAM-Speicherbereich des Rechners durch die neu hinzugefügten Routinen nicht eingeengt wird. Andernfalls wäre die Verträglichkeit mit größeren Anwenderprogrammen gefährdet, die von der Floppy-Diskette auf die Harddisk kopiert wurden. Aus diesem Grund wurden fast alle diese Programmerweiterungen in einem Controller-EPROM abgespeichert. Auch das von diesen Programmen zusätzlich benötigte RAM befindet sich auf der Controller-Platine.

Ein weiteres Problem entsteht beim „Einbinden" der für die Harddisk benötigten Software-Erweiterungen in das Original-Betriebssystem. Beim allgemein üblichen Verfahren wird das Original-Betriebssystem im Speicher des Rechners geändert und in geänderter Form auf die Spuren des Boot-Volumes abgespeichert. Dieses Verfahren ist jedoch in unserem Fall aus mehreren Gründen ungeeignet: Erstens kann die Charakteristik des an den Controller angeschlossenen Harddisk-Laufwerks verschieden sein, und zweitens wird die Speicherkapazität für das jeweilige Betriebssystem erst durch den Benutzer festgelegt. Bei der Vielzahl der möglichen Systemkonfigurationen würde eine Unmenge „neuer" Betriebssystem-Varianten entstehen.

Es ist daher günstiger, ein anderes Verfahren zu verwenden, um das Original-Betriebssystem zu modifizieren (patchen). Bei diesem Verfahren enthalten die Harddisk-Boot-Volumes immer die Originalversion des Betriebssystems. Erst beim Booten (nur von der Harddisk) führt ein spezielles Hilfsprogramm (Autopatch-Boot-Programm) die nötigen Betriebssystem-Modifikationen durch. Dabei wird zuerst die Information über die Konfiguration und Charakteristik des Laufwerks von der Harddisk gelesen und dementsprechend automatisch die nötigen Software-Änderungen des Betriebssystems ausgeführt. Am Ende dieses Boot-Vorgangs befindet sich im Speicher des Rechners eine Betriebssystem-Version, die genau an die Parameter des vorhandenen Speichermediums angepaßt ist. Das Autopatch-Boot-Programm befindet sich auch im EPROM des Controllers.



▲
Abb. 4: Blockschaltbild der Hardware-Treiberroutinen

Zur Zeit werden die vier Apple-Betriebssysteme DOS 3.3, CP/M 2.2, UCSD-Pascal 1.1 und ProDOS beim Boot-Vorgang automatisch erkannt und modifiziert.
Die Firmware kann man in zwei Gruppen teilen:
1. Programme, die unabhängig vom jeweiligen Betriebssystem arbeiten.
2. Programme, die eine spezifische Form für jedes Betriebssystem haben.

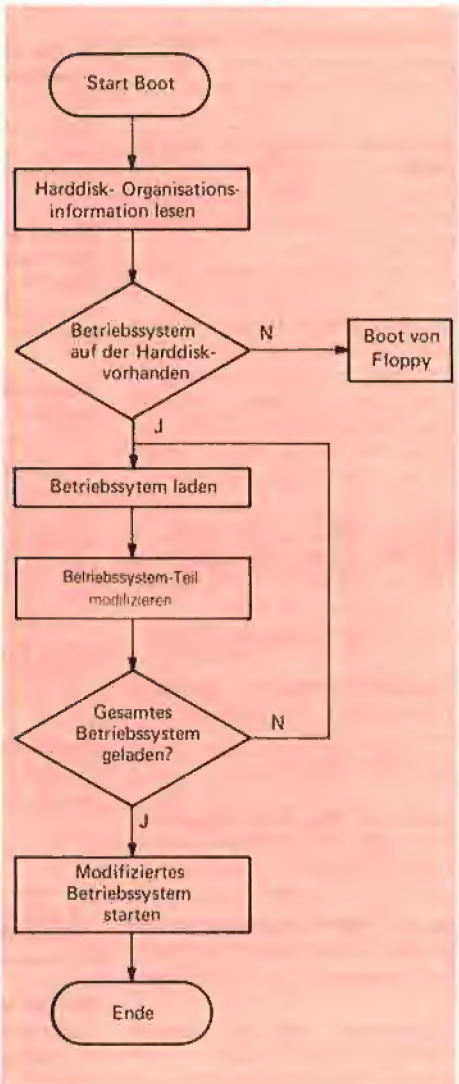
4.2.1. Betriebssystem-unabhängige Routinen

Jedes Betriebssystem muß letztendlich Daten auf der Platte abspeichern und von der Platte lesen können. Wie schon bei der Hardware-Beschreibung angedeutet wurde, braucht man dazu entsprechende Programme, die für die Ansteuerung der Controller-Hardware verantwortlich sind. Zu dieser Gruppe gehören die Programme der niedrigsten Ebene, z.B. das Programm, das einen Datensektor auf die Harddisk schreibt. Diese Routinen müssen vor allem die Hardware-Eigenschaften des Controllers berücksichtigen. Die nötigen Eingangsparameter werden durch die betriebssystem-abhängigen Firmware-Routinen geliefert.
Das Blockschaltbild dieser Routinen ist in **Abb. 4** dargestellt.

4.2.2. Betriebssystem-abhängige Routinen

Zu dieser Gruppe gehören die Autopatch-Boot-Programme sowie die Programme, die ein Software-Interface zwischen dem

Abb. 5: Allgemeine Struktur des Autopatch-Boot-Programmes
▼



PEEKER Börse

Verkauf Hardware

Wegen HD-Disk 2 Pr. FD-LW zu verk. FD55FV a) LW 300 + MIT4853 a) DM 250,–, Tel. 0 82 31 / 3 16 25 vorm. 10-12 Uhr

Apple IIc (12/85), 640K, Z80, Monitor, Maus, Joystick, Hotlin K u. a. Software, DM 4500,–, Tel. 0 71 56 / 2 74 51 nach 19 Uhr

Apple IIe, IIc Mac und Apple-Zubehör – Bei uns sind Sie an der richtigen Adresse.
Kostenlos Preisliste anfordern!
Welz Electronic, Tel. 041 92/46 28

Plotter, Flachbrett DIN A1 RS 232 V24/Centr. VHB DM 3500. Seelig, Fahltskamp 75d, 2080 Pinneberg

Apple II-Nachbau (COSMO) kompl. mit Laufwerk, Bildsch., Controller u. Handbuch zu verk. DM 1150,– (NP: ca. 1900,–), Tel. 0 64 46 / 16 57, Biebertal

Z80H (AP22) 8MHZ komplett neu zu verkaufen. Preis VHB Tel. 0 73 24 / 85 84 ab 18 Uhr

Verkaufe Ramworks und Z-RAM Speichererweiterung zu Apple IIe/c, Preise auf Anfrage. Musicomp, Basler Str. 52, CH-4102 Binnigen, Schweiz, Tel. 061 / 47 89 34 nachm.

****** **Apple IIe/+** ****** 512K RAM-DISK für DOS + CP/M + UCSD. !!!!! Rasend schnell !!!!! Lehrplatine + Software 99,– DM Bestückt (ohne Ram's) 199,– DM Norbert Menke, Tel. 05251/25887 Kilianstr. 32, 4790 Paderborn

IIe, 9 Mon. alt, 128K, 2 Laufw. J-Stick, viel Software + Literatur VB 3500,– mit Imagewriter, 3 Mon. alt DM 4800,–, Tel. 0 23 65 / 2 60 95 ab 17.00 Uhr

Tintenstrahldrucker Siemens PT88 und PREH-Tastatur preiswert zu verkaufen, Tel. 02 08 / 80 58 08

Apple IIc, Monitor, Joystick, div. Softw. u. viele Bücher VB DM 1800,–, Tel. 07 21 / 75 29 73

Epson Grafikinterfaceumbau 8132W für AppleWorks ab DM 20,–. Mahr, Waldacker 71, 7300 Esslingen

Macintosh 128 auf 512KB DM 399,–, Fa. Schlösser, Tel. ab 17.00 089 / 98 58 89

8-Zoll Laufwerke zu verkaufen, 2 Doppelkopf LW mit Controler und Software, L. Schulz, Tel. 0 72 45 / 47 86

68000 mit 512K (AP20) für CPIM68K, mit Pseudodisk, DM 1090,– CP/M 4, 5× schneller, Z80H (AP22), 64K, Software, DM 550! Tel. 05 51 / 7 64 26

Apple IIc (1 Jahr alt) + Maus + Joystick + 15 Disketten mit Org. Software (z. B. Summer Games + Time Zone + Mousepaint + FRZ. Lernprogramm . . .) für DM 2050,–, Tobias Ladewig, Tel. 07 91 / 46 29 50

Apple II+ (komp) IBM-Look Prof. Aufbau, 2 × 80 Spur (TEAC), Z80, 80 Zeichen, Druckerinterface, Boss-Tast. Monitor, Bücher, div. extra (FF-Cards, Prommer, etc.) Preis VHB, Tel. 0 73 24 / 85 84 ab 18 Uhr

Apple IIc Paket 2. LW, Maus, Imagewriter, bernst. 12. Monitor, 100 Disk., 5 dtsch. Handbücher. Preis DM 4400,–, Tel. 07 11 / 607 12 88.

Apple IIc, Monitor + Monitorstand, 2. Laufw., NZ80IIc-Karte, IIc-Lit. von M&T-Verlag, nur kompl., Verk. DM 3200,– evtl. + Imagewriter (DM 1000,–), Tel. 0 26 51 / 4 26 27

Apple Superseriel DM 250,– Neucom Parallel DM 100,–, Epson CX-21 DM 400,–, Fischer-Interface DM 180,–, und Computing DM 180,–, Tel. 0 71 71 / 8 98 98

Apple-IIc + Imagewriter + Maus + 2. Laufwerk + Software + Literatur + TAXAN Monitor VB DM 4200,–, Tel. 07 21 / 3 25 83, Karlsruhe

Apple II (comp.), 64 K, Monitor, FDC4, 80Z, Z80, Uhr, Sprachk., Joystick, FD-55F, PALK. UHF Mod., Eprommer, Druckerkarte, massig Software u. Literatur DM 2600,– nach 18 Uhr, Tel. 0 24 61 / 5 18 32

Verkaufe Apple IIe m. Drucker 2 LW Monitor, v. Software, Preis a. Anfrage, Tel. 089/325481

Ankauf Hardware

Suche MAC 512 + ext. LW + Software bis DM 5.500,–, Bilger, Altenbergstr. 62, 7000 Stuttgart 1, Tel. 607 12 88

Verkauf Software

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

Progr. zur Verwaltung und grafischen Darstellung (Kurven, Trendanalyse) von Börsenkursen. DM 50,–, Info unter Tel. 030 / 752 01 94

– – – STARTUP 1.0 – – – Universelles Begrüßungsprogramm unter ProDOS für Apple IIc bzw. IIe + 80 Zeichen. Assemblerprogr. mit vielen Features: Online, Prefix setzen, Subdirectory- und Programmaufruf bis zu 128 Z., usw. Bedienung des Leuchtzeiger-Menus mit Tastatur oder Joystick. Disk DM 39,90 incl. Source DM 69 zzgl. DM 4,50 Versandk. Info gegen Freiumschlag: M. Lindemann, Mallinckrodt 162, 4600 Dortmund 1

Verk. ORCA/M (ProDOS) DM 310,–, PRINT SHOP DM 75,–, ASMBL-Kurs DM 35,–, BEAGLE Graph DM 190,–, extra K DM 100,–, GPLE DM 140,– – Orig. – Tel. 0 45 31 / 8 50 75 ab 20 Uhr

Software Uhr für Apple II+, e, c, Zeitschaltmöglichkeit Diskette + Anleitung DM 25,– Oecking Tel: Do. 02 31 / 39 19 20

P-Code DISASSEMBLER DM 50,–, System Patch – Info: K. Seiler, Willy-A.-Allee 1, 7500 Karlsruhe

MerlinPro DOS/ProDOS-Ass., neu, Fullscr.-Ed. für 128K-Apple DM 360,– (NP 490,–), Tel. 00 41 61 / 63 35 85

Apple II: Verkaufe und erstelle Software. Info gegen 0,80 DM in Briefmarken bei W. Rittmeyer, Wehrbruchweg 30, 4060 Viersen 1

Einkommen-Lohnsteuer 1985 Apple-Disk. DM 45,–, Dipl.-Fin. Wirt. Uwe Olufs, 5216 Niederkassel 2, Bachstr. 70, Tel. 0 22 08 / 48 15 abends

Für Basis 108!! DOS Mover, FP Autoboot (kein Pre Boot mehr für BASIC) gg. DM 40,– Real, Dorstener 67, 4660 Gelsenkirchen-Buer

Super-Datenbank DATA-BOSS und andere Anwenderprogr., E. Heinz, Waldgürtel 7, 5060 B-Gld. 1

Pauker: Latein DM 40,– VOK. DM 25,– Heinz, Waldgürtel 7, 5060 B-Gld. 1

Ankauf Software

Suche Adventure für Apple IIe, Tel. 0 46 54 / 265

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

Für Apple II+: Als CP/M+ Card 699 Oki ML 80 499. Org. Apple 2 SW: ORCA/M 249. Merlin 149. MAB 6502 Debugger 99. Bag of Tricks 49. H. Jung, Masurenstr. 11, 7120 Bietigheim-Bissingen

Suche Kontakt zu IIe-User im GR-Hannover! Tel. 0 51 71 / 5 33 46, ab 19.00 Uhr

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

Anzeigenschluß für Peeker 5/86 ist am 1. 4. 1986

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**

jeweiligen Betriebssystem und den untersten Controller-Treiberroutinen schaffen. Diese Programme müssen die spezifischen Eigenschaften der Betriebssysteme berücksichtigen. Genaue Kenntnisse über die Arbeitsweise, Speicherorganisation und Struktur des Betriebssystems sind bei der Entwicklung dieser Programmen unerläßlich.
Wie schon angedeutet, übernimmt das Autopatch-Boot-Programm die Kontrolle über das Laden des Betriebssystems und führt dabei einige Änderungen aus. Am Ende des Ladevorganges befindet sich im Rechner eine Betriebssystem-Variante, die gleichzeitig mit der Floppy- und Harddisk arbeiten kann.
Grob gesehen setzt das Autopatch-Boot-Programm im Original-Betriebssystem an einigen Stellen Sprungbefehle bzw. Unterprogrammaufrufe ein, die dem Betriebssystem den Zugriff auf die entsprechenden Harddisk-Treiberroutinen ermöglichen.
Die allgemeine Struktur des Autopatch-Boot Programmes ist in **Abb. 5** dargestellt. Die nötigen Modifikationen sind vom jeweiligen Betriebssystem abhängig und müssen daher für jedes Betriebssystem getrennt betrachtet werden. Wir wollen uns dabei nur auf die grundsätzlichen Ideen der erforderlichen Betriebssystem-Änderungen und -Ergänzungen beschränken.

5. Die einzelnen Betriebssysteme

5.1. DOS 3.3

Die ursprüngliche Version des DOS 3.3 ist für die Verwaltung von Disketten mit einer Speicherkapazität von 143K vorgesehen. (Anm.: Wenn 1K = 1024 Bytes, dann exakt 140K; wenn 1K = 1000 Bytes, dann ca. 143K.) Durch die einfachen Modifikationen <2> des INIT-DOS-Befehles lassen sich die Verwaltungsmöglichkeiten des DOS-File-Managers bis auf 400K erweitern. Datenträger, die eine größere Speicherkapazität haben, können unter DOS 3.3 nicht mehr in Form einzelner geschlossener „Volumes" organisiert werden. Es taucht somit bei der Verwendung eines Festplattenlaufwerks die erste Frage auf: Ist es überhaupt möglich, eine viel größere Speicherkapazität der Festplatte unter DOS 3.3 vollständig auszunutzen? Diese Frage läßt sich mit „ja" beantworten. Man zerlegt den für DOS zugeteilten Speicherplatz auf der Harddisk in kleinere Einheiten. Diese Speichereinheiten können bis zu 400K groß sein und müssen so organisiert werden, daß sie vom DOS-File-Manager als getrennte Floppy-Disketten betrachtet werden können.

Sollten „Harddisk-Speichereinheiten", die größer als 143K sind, verwaltet werden, so sind einige Änderungen im DOS-File-Manager notwendig. Dies kann zu einigen Inkompatibilitäten der vorhandenen DOS-Utility-Programme führen. Darum ist es

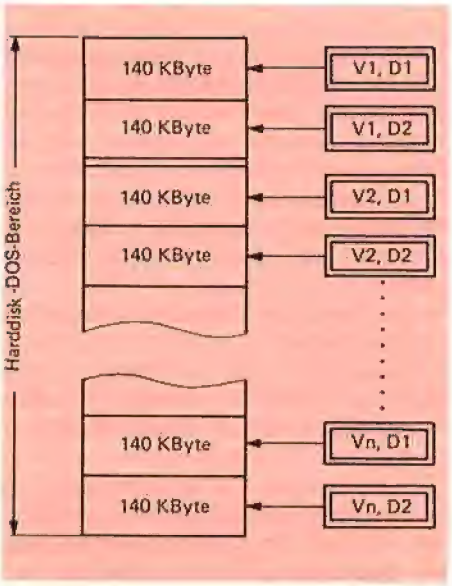


Abb. 6: Aufteilung des DOS-Bereichs auf der Harddisk

von DOS-Speichereinheiten selektieren zu können, müssen diese durch die Verwendung eines weiteren DOS-Befehlsparameters, nämlich der Volume-Nummer, kodiert werden. Diese Kodierung ist in **Abb. 6** dargestellt.

Die Kodierung sowie die Verteilung der Speichereinheiten auf der Harddisk übernimmt die entsprechende Software des Controllers.
Bei der Berücksichtigung der oben genannten DOS-Organisation lassen sich die für die Harddisk erforderlichen DOS-Patches auf die RWTS-Ebene beschränken. Das Betriebssystem muß nun vor allem um die an den Harddisk-Controller angepaßte RWTS erweitert werden, die sich im EPROM des Controllers befindet (d.h. Original-RWTS für Floppy-Zugriff und EPROM-RWTS für Harddisk-Zugriff).
Um diese Routinen im Original-Betriebssystem zu integrieren, muß das DOS-Autopatch-Boot-Programm einige Verzweigungen im RWTS-Bereich einsetzen, die bei jedem Zugriff auf die Harddisk die entsprechenden EPROM-RWTS-Routinen aufrufen. Beim Zugriff auf das Floppy-Laufwerk müssen die „alten" RWTS-Routinen verwendet werden, die sich im Apple-RAM befinden. **Abb. 7** zeigt die gepatchte DOS-Version.

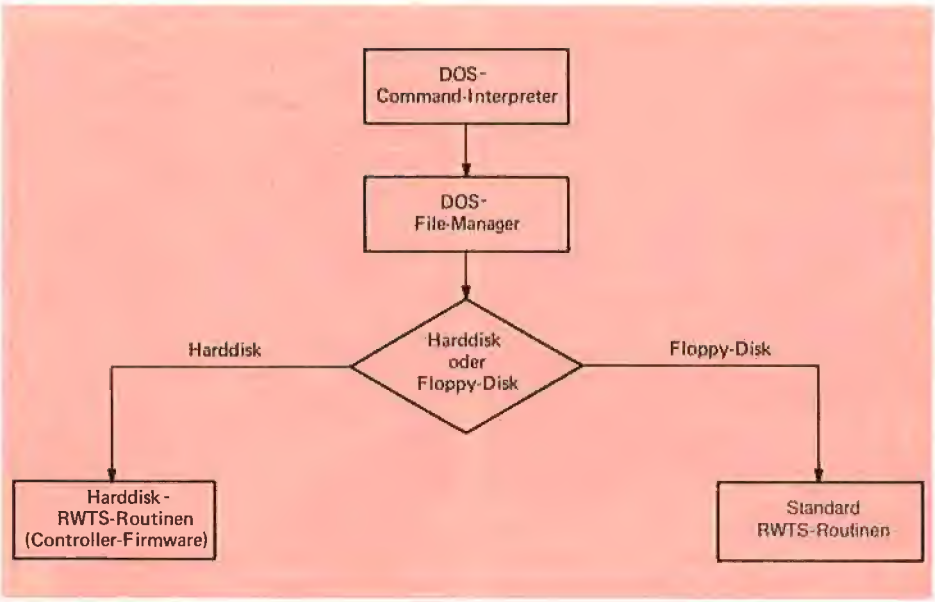


Abb. 7: Allgemeine Struktur der gepatchten DOS-Version

vorteilhafter, die Speichereinheitsgröße bei 143K zu belassen.
Hat man auf der Harddisk mehrere 143K-Einheiten eingerichtet, so muß dafür gesorgt werden, daß diese Einheiten vom Benutzer gezielt angesprochen werden können. Die DOS-Befehlstruktur erlaubt lediglich das Ansprechen von zwei Laufwerken. Um dennoch eine größere Anzahl

Die Struktur der Harddisk-RWTS-Routinen wird einerseits durch die Software-Schnittstelle des File-Managers und andererseits durch die Hard/Software-Schnittstelle des Controllers bestimmt. Die Schnittstelle zum File-Manager ist standardisiert und z.B. in <3, 4> ausführlich beschrieben. Die Parameterübergabe erfolgt dabei durch den 17 Bytes langen

Input/Output-Control-Block (IOB). Vor jedem Anruf der RWTS-Routinen setzt der DOS-File-Manager einige Bytes in den IOB ein (z.B. Drive-, Volume-, Sektor-Nummer, Adresse des internen Datenpuffers sowie Command Code). Diese Parameter stellen die Eingabeparameter für die RWTS-Routinen dar. Die RWTS führt den angegebenen Befehl aus und informiert den File-Manager über den Ausführungszustand. Diese Information wird durch die Beschreibung des Fehler-Bytes im IOB an den File-Manager übermittelt. Der grundsätzliche Unterschied zwischen der Harddisk- und Floppy-Disk-RWTS besteht in der Umrechnung der logischen (durch den File-Manager gelieferten) I/O-Disk-Parameter in die physikalischen Werte.
Bei der Berechnung einer physikalischen Harddisk-Track-Nummer müssen z.B. folgende Parameter berücksichtigt werden:
– logische Track-Nummer im IOB
– Volume-Nummer im IOB
– Anzahl der Köpfe des Laufwerks
– Anfang-Track für das angesprochene Volume
– Ersatz-Track-Tabelle
Die letzten drei Parameter wurden während des Einrichtungsvorgangs auf der Harddisk gespeichert und stehen dem Berechnungsprogramm jederzeit zur Verfügung.
Die allgemeine Struktur der Harddisk-RWTS-Routinen zeigt **Abb. 8**

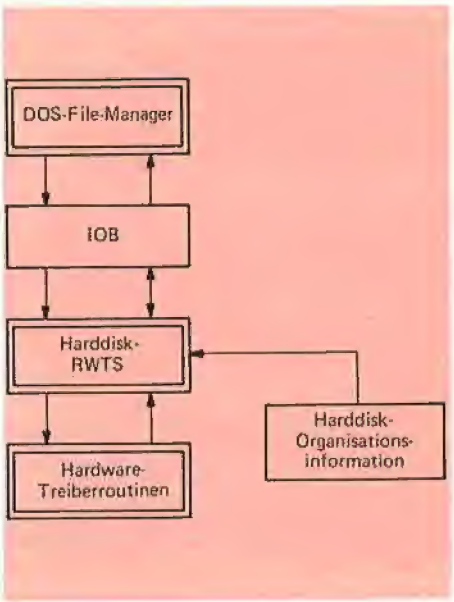


Abb. 8: Allgemeine Struktur der Harddisk-RWTS-Routinen

5.2. CP/M 2.2

Um den Betrieb mit der Harddisk zu ermöglichen, sind beim Betriebssystem CP/M 2.2 eine Reihe von Änderungen und Ergänzungen notwendig. Die modulare Struktur dieses Betriebssystems erlaubt jedoch, die notwendigen Eingriffe auf die BIOS-Ebene zu beschränken.
Die ursprüngliche Version des CP/M 2.2 ist imstande, bis zu sechs Floppy-Disk-Laufwerke zu verwalten, wobei jedes Laufwerk dieselbe Charakteristik besitzen muß. Daher wurde im BIOS nur ein einziger Disk Parameter Block (DPB) vorgesehen.
Um den für das CP/M-2.2-Betriebssystem auf der Harddisk verfügbaren Speicherplatz optimal (nach dem Wunsch des Benutzers) ausnutzen zu können, müssen folgende Voraussetzungen erfüllt werden:
1. Der Benutzer kann den für CP/M 2.2 vorgesehenen Speicherplatz auf der Harddisk auf bis zu sechs getrennte Volumes teilen.
2. Die Größe des Harddisk-Volumes soll variabel bleiben.
3. Die vom CP/M 2.2 verwalteten Volumes kann der Benutzer beliebig zwischen Harddisk-Volumes und Floppy-Disk-Volumes teilen.
Will der Benutzer z.B. in Zukunft nur mit zwei Floppy-Disk-Laufwerken arbeiten, dann können die restlichen vier Volumes der Harddisk zugeordnet werden.
Ein Beispiel der externen Speicherorganisation für CP/M 2.2 ist in **Abb. 9** dargestellt.
Die Verwaltung von Nicht-Standard-Volumes unter CP/M 2.2 ist nur dann möglich,

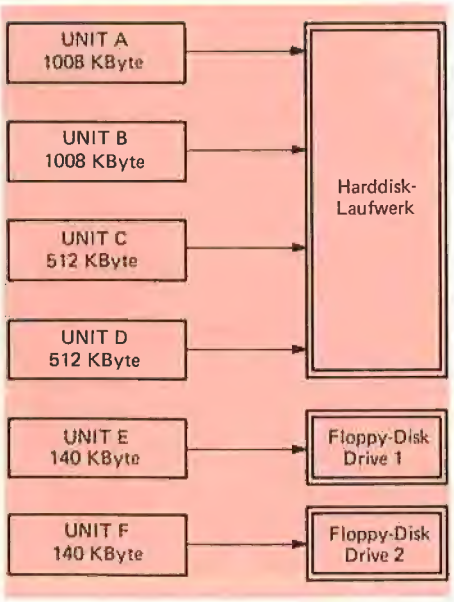


Abb. 9: Beispiel einer externen Speicherorganisation für das CP/M-2.2-Betriebssystem

wenn für jede vorgesehene Volume-Größe ein entsprechender DPB vorgesehen ist. Die Parameter des DPB charakterisieren das Speicher-Volume für das Betriebssystem und müssen an die Größe des Volumes angepaßt werden <5>. Die DPBs für die verschiedenen Volume-Größen sind im Harddisk-Controller-EPROM abgespeichert. Der Zugriff des Betriebssystems auf den DPB erfolgt über einen Pointer, der sich in dem Disk Parameter Header (DPH) des entsprechenden Volumes befindet. Es ist die Aufgabe des Autopatch-Boot-Programms, diese Pointer so zu setzen, daß sie auf die Adressen der entsprechenden DPBs im Controller-EPROM zeigen. Bei der Verwaltung von größeren Volumes müssen auch die Pointer in den DPHs, die auf die Allocation-Vektoren (ALV) zeigen, verändert werden. Das Autopatch-Boot-Programm muß auch die BIOS-Jump-Tabelle so „verbiegen", daß bei jedem Zugriff auf die Harddisk deren Treiberroutinen angesprungen werden. In allen diesen Routinen müssen bei der Berechnung der physikalischen Disk-Parameter nicht nur die BIOS-I/O-Disk-Parameter berücksichtigt werden, sondern auch die internen Harddisk-Informationen, die während des Vorbereitungsprozesses auf der Harddisk abgespeichert wurden.

5.3. Apple-UCSD-Pascal 1.1

Im Vergleich zu den DOS- und CP/M-Apple-Betriebssystemen ist das Apple-UCSD-Pascal-Betriebssystem viel flexibler, wenn es um die Verwaltung von externen Speichermedien im „Nicht-Apple-Standard" geht. Bei UCSD-Pascal sind die externen Speichermedien in Form von kontinuierlich durchnumerierten 512-Byte-Blöcken organisiert <6>.
Obwohl die gesamte Anzahl der Blöcke, die von UCSD-Pascal auf einem Volume verwaltet werden kann, sehr groß ist (65536), gibt es andere Beschränkungen, die eine optimale Ausnutzung von sehr großen Volumes unmöglich machen (so ist z.B. die maximale Anzahl von Directory-Einträgen für ein Volume auf 77 begrenzt). Darum erweist es sich in vielen Fällen vorteilhaft, den gesamten für UCSD-Pascal auf der Harddisk reservierten Speicherplatz auf mehrere Volumes zu verteilen. Dabei muß man berücksichtigen, daß die maximale Anzahl von Pascal-Volumes auf *sechs* begrenzt ist. Bei der Anpassung des UCSD-Pascal-Betriebssystems an das Harddisk-System müssen deswegen folgende Voraussetzungen erfüllt werden:
– Der Benutzer kann den für den UCSD-Pascal reservierten Speicherplatz auf bis zu sechs getrennte Volumes verteilen. Die Volumes können verschiedene Speicherkapazitäten haben.
– Die Volumes können beliebig zwischen Harddisk- und Floppy-Disk-Volumes geteilt werden.
Die Größe sowie die physikalische Plazierung der Volumes wird in der Harddisk-Organisationstabelle während des Konfigurationsvorgangs abgespeichert.

Die tatsächliche Anzahl der Blöcke, die auf einem bestimmten Volume Platz finden, wird beim Schreiben des Volume-Directory abgespeichert (dafür sorgt das Pascal-Konfigurationsprogramm) und kann daher bei Bedarf jederzeit vom Betriebssystem abgerufen werden.
Die nötigen UCSD-Pascal-Ergänzungen erstrecken sich im wesentlichen auf die an den Harddisk-Controller angepaßten Pascal-Disk-I/O-Routinen (BIOS). Diese Routinen sind im EPROM des Controllers abgespeichert.
Das Autopatch-Boot-Programm muß dafür sorgen, daß für jedes Harddisk-Volume die Adresse des Harddisk-Treibers in der Treiber-Adreßtabelle vorhanden ist. In dieser Tabelle $FEB6-$FEC1 sind für jedes Speicher-Volume zwei Bytes vorgesehen, die die Adresse der (für dieses Volume) zuständigen Treiber-Software enthalten. Vor dem Aufruf der Treiberroutinen werden von Pascal die entsprechenden I/O-Disk-Parameter gesetzt <7>. Die Parameterübergabe findet hier durch den Stack statt. Der Befehlscode selbst wird an die Treiberroutinen durch das X-Register übergeben. Nach dem Return enthält das X-Register den Fehlercode.

5.4. ProDOS

Das ProDOS-Betriebssystem, entwickelt von der Firma Apple Computer selbst, wurde bereits für eine Zusammenarbeit mit externen Speichermedien unterschiedlicher Kapazität konzipiert. Darum ist es bei ProDOS im Vergleich mit anderen Apple-Betriebssystemen relativ einfach, die Harddisk-Treiberroutinen in das Betriebssystem zu integrieren. Der Zugriff des ProDOS-File-Managers auf die externen Datenträger erfolgt auf Block-Ebene, wobei ein Block aus 512 Bytes besteht. Unter ProDOS kann ein einzelner Datenträger (Volume) insgesamt 32M und eine einzelne Datei insgesamt 16M umfassen. Berücksichtigt man dabei die hierarchische Struktur des ProDOS-Inhaltsverzeichnisses, die die Behandlung von mehreren Dateien auf einem Volume einfach und übersichtlich macht, kann man ohne Nachteile den ganzen vorgesehenen Platz für ProDOS auf der Harddisk als ein Volume anlegen.
Diese Betriebssystem-Konzeption vereinfacht die Anpassung an das Harddisk-System enorm. Um das ProDOS-Betriebssystem über das Vorhandensein der Harddisk-Treiberroutinen zu informieren, muß die Speicherstelle $C7FF (Controller-EPROM) einen Wert zwischen $01 und $FE enthalten. Die Charakteristik des externen Datenträgers und der Treiberroutinen ist durch das Status-Byte $C7FE beschrieben <8>. Sind mindestens die zwei letzten Bits in diesem Status-Byte gesetzt (das bedeutet, die Controller-Treiber-Software unterstützt den Daten-Schreib- und -Lesevorgang), so wird sich das ProDOS beim Booten automatisch die Adresse der Harddisk-Treiberroutinen in der „Global Page" merken. Das High-Byte dieser Adresse ist gleich $C7 und das Low-Byte wird durch den Inhalt der Speicherstelle $C7FF bestimmt.
Ist die gesamte Anzahl der Blöcke, die auf dem ProDOS-Teil der Harddisk abgespeichert werden können, von Anfang an bekannt, so kann man diesen Wert direkt in der Adresse $C7FC-$C7FD abspeichern. In unserem Fall ist jedoch dieses Verfahren nicht zu gebrauchen, weil die Größe des ProDOS-Volumes erst durch den Benutzer bestimmt wird. Das ProDOS bietet hier eine andere nützliche Möglichkeit, das Betriebssystem über die Größe des Volumes zu informieren. Dazu muß der Wert $0000 bei Adresse $C7FC-$C7FD abgespeichert werden. Dieser Wert teilt dem Betriebssystem mit, daß die Ermittlung der Blockanzahl durch die Ausführung eines Status-Request-Kommandos möglich ist. Das Status-Request-Kommando muß die Information der Organisationstabelle der Harddisk bei der Berechnung dieser Anzahl benutzen. Diese Anzahl von Blöcken wird vom ProDOS beim Beschreiben der Harddisk-„Volume-Bit-Map" sowie des Directory-Headers berücksichtigt, so daß in Zukunft erkannt wird, wie groß das Speichermedium ist.
Die Schnittstelle zwischen den Treiberroutinen und dem ProDOS-File-Manager ist definiert und z.B. in <8> beschrieben. Die Harddisk-Treiber-Software muß folgende Kommandos unterstützen:
– Status Request
– Read Request
– Write Request
Die Parameterübergabe für die Treiberroutinen erfolgt hier durch die Speicherstellen

$0042 bis $0047 folgendermaßen:
$0042: Comand Code
$0043: Unit-Nummer
$0044-$0045: Zeiger des Datenpuffers
$0046-$0047: Block-Nummer
Die Harddisk-Treiberroutinen sollen die physikalische Datenübertragung zwischen der Platte und dem ProDOS-Datenpuffer steuern und dem ProDOS-File-Manager die Fehlermeldungen liefern. Das Auftreten eines Fehlers wird durch Setzen des Carry-Flags signalisiert, und der Akkumulator enthält in diesem Fall den Fehlercode. Folgende Fehlercodes sollen implementiert werden:
$27: I/O-Fehler
$28: Harddisk-System nicht angeschlossen
$2B: Write Protected
Bei der Berechnung der physikalischen Harddisk-Parameter müssen die Treiberroutinen selbstverständlich die Harddisk-Organisations- und Ersatz-Track-Tabelle in Anspruch nehmen.

6. Schlußbemerkungen

Der Anschluß und die Integration eines Harddisk-Systems an den Apple-Rechner stellt ein komplexes Problem dar, das in einem Artikel nicht vollständig beschrieben werden kann. Wir mußten uns daher auf die allgemeine Beschreibung der Hard- und Software-Komponenten beschränken. Für diejenigen, die sich für die praktische Anwendung des Controllers interessieren, steht ein ausführliches Handbuch sowie eine Demo-Diskette zur Verfügung <9>.
Obwohl die dargestellten Überlegungen auf den Apple-Rechner bezogen waren, lassen sich die Grundideen auch auf andere Rechner übertragen. Als besonders praktisch und nützlich haben sich die Eigenschaften des Controllers erwiesen, die dem Benutzer eine gleichzeitige Ausnutzung des Harddisk-Speicherplatzes unter den verschiedenen Betriebssystemen erlauben. Die große Speicherkapazität der Harddisk kann dadurch flexibler und optimaler genutzt werden.
Zusammenfassend kann man sagen, daß der vorgestellte Harddisk-Controller dem Benutzer einen einfachen Anschluß eines großen und schnellen externen Datenträgers an seinen Rechner erlaubt.

Literatur

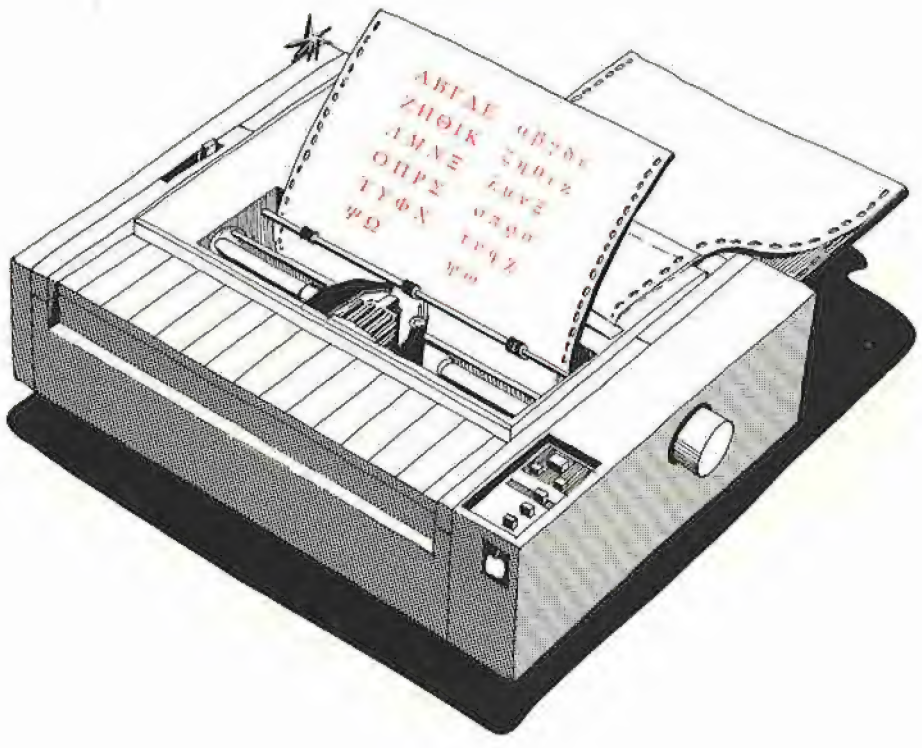
<1> Datenbläter der Firma Western Digital zu dem WD1010-Chip
<2> Schöpe, W.: Mehr Platz auf Apple-Disketten, mc 1984, Heft 89, S. 23
<3> Luebbert, W.F.: What's Where in the Apple, Micro Inc., 1982
<4> Stiehl, U.: Apple DOS 3.3, 3. Aufl. 1986
<5> CP/M 2.2. Alteration Guide, Digital Research, 1979
<6> Rosing, M., McLauren, K.: Pascal Internals, CALL-A.P.P.L.E In Depth, S. 79
<7> Schröter, M.: RAM-Disk-Driver für Pascal 1.1., Peeker 1985, Heft 6, S. 48
<8> Stiehl, U.: ProDOS für Aufsteiger, Bd. 1, speziell S. 30: Disk-Controller-Erkennungsbytes
<9> Megaboard Bedienungshandbuch, Frank & Britting GmbH
<10> Autopatch Floppy Disk Controller AFDC2 Benutzer-Handbuch, Erphi Electronic GmbH, 1985

Sonderzeichen auf dem Imagewriter

Mit BITEDITOR- und ANIMATRIX-Zeichensätzen

von Carl Frieder Mahr

Wollten Sie schon einmal einen Text mit wissenschaftlichen Sonderzeichen oder Grafiksymbolen auf dem Imagewriter ausdrucken? Dann haben Sie sicher erfahren, welche Mühe man mit den Byte-Berechnungen hat, bis das Zeichen endlich auf dem Papier erscheint. Das Programm **FONT.LOADER** zusammen mit BITEDITOR (von J. Klamt, Peeker 7/85, S. 29) oder „ANIMATRIX" (aus DOS-Toolkit-Diskette) hilft Ihnen bei dieser Arbeit, so daß Sie wieder Zeit haben, Dinge zu erledigen, die mehr Spaß machen, als genervt über endlosen Zahlenkolonnen zu sitzen.



1. Die Grundidee

Der Apple-Imagewriter kann in seinem Puffer bis zu 175 frei definierbare Zeichen speichern. Um diese Zeichen zu definieren, müssen bestimmte Steuerzeichen zum Drucker gesendet werden, gefolgt von den eigentlichen Daten-Bytes für das neue Zeichen.
Da das manuelle Berechnen der Daten-Bytes eines Zeichens sehr mühsam ist, sollte man diese Aufgabe dem Computer überlassen. Entsprechende Editierprogramme sind sowohl im Peeker wie auch auf der Apple-DOS-Toolkit-Diskette erschienen, die neben einem Assembler („EDASM") und einer Applesoft-Utitity („APA") diverse Zeichensätze mit entsprechenden Hilfsprogrammen („ANIMATRIX" u.a.) enthält, die in dem Buch „Bewegte Apple-Grafik" von Arne Schäpers disassembliert und kommentiert worden sind. Mit diesen Programmen lassen sich Zeichensätze einfach erstellen und verändern. Diese Zeichensätze sind wegen ihres Formates jedoch primär für die 80-Zeichenkarte-EPROMs oder die Hires-Grafik geeignet.
Mit FONT.LOADER lassen sich diese teilweise schon auf Diskette vorhandenen Zeichensätze in das Imagewriter-Format konvertieren und dann in den Drucker laden. Damit ist es nun möglich, Sonderzeichen oder bestimmte Schriften in Texten zu verwenden.

2. Das Programm

FONT.LOADER gliedert sich in drei Teile:

2.1. Das Steuerprogramm

Dieses Steuerprogramm erlaubt es Ihnen, die gewünschten Zeichensätze in den Drucker zu übertragen. Das Programm ist durch Menüführung sehr einfach zu bedienen und kann so auch von Anwendern

benutzt werden, die Texte mit dem Computer schreiben, sich aber mit dessen internem Ablauf nicht beschäftigen können oder wollen. FONT.LOADER stellt ihnen verschiedene Funktionen zur Verfügung:

2.1.1. Zeichensatzformat ändern
Mit dieser Funktion können Sie das Datenformat des Zeichensatzes einstellen, den Sie in den Drucker laden wollen. Keine Angst, Sie müssen sich nicht schon wieder mit den Zeichensatzinterna befassen! FONT.LOADER kann zwei gängige Formate verarbeiten: Zeichensätze des BITEDITORs und von ANIMATRIX aus dem DOS Toolkit. Wählen Sie diese Option, so wird abwechselnd eines der beiden Formate ausgewählt. Die Vorgabe beim Starten des Programms ist das DOS-Toolkit-Format.

2.1.2. Darstellungsart ändern
Sie können beim FONT.LOADER zwischen zwei Darstellungsarten auswählen: normal und invers. Diese entsprechen den beiden Applesoft-Modi „NORMAL" und „INVERSE". Um besonders wichtige Textstellen hervorzuheben, können Sie zum Beispiel inverse Schrift benutzen. Standard-Darstellungsformat beim Starten des Programms ist normale Schrift. Die Funktion „Darstellungsart ändern" wechselt zwischen den beiden Modi.

2.1.3. Zeichensatz von Diskette laden
Mit dieser Option können Sie fertige Zeichensätze von der Diskette laden. Beachten Sie bitte hierbei, daß das entsprechende Zeichensatzformat *vor* dem Einladen eines Zeichensatzes richtig gesetzt ist. Sollten Sie den Namen des Zeichensatzes nicht genau wissen, so können Sie sich durch Eingabe von Return den Diskettenінhalt anzeigen lassen. Sollten Sie aus Versehen diese Funktion gewählt haben, so kommen Sie mit Ctrl-E zum Hauptmenü zurück.

2.1.4. Zeichensatz zum Drucker übertragen
Durch Wählen dieser Funktion übertragen Sie den eingeladenen Zeichensatz aus dem Computer in den Drucker. Dieses Verfahren wird als „Downloading" bezeichnet und ist im Druckerhandbuch beschrieben. Sollten Sie den Drucker noch nicht eingeschaltet haben, so tun Sie dies nun und beantworten Sie die Frage „Imagewriter bereit?„ mit „J". Die in den Drukker geladenen Zeichensätze bleiben bestehen, bis Sie den Drucker abschalten. Es beeinflußt die Zeichensätze nicht, wenn Sie booten oder das Betriebssystem wechseln.

2.1.5. Probeausdruck
Diese Funktion erstellt Ihnen einen Probeausdruck des Zeichensatzes, der sich gerade im RAM des Druckers befindet. Es wird der normale Zeichensatz gedruckt und darunter die Zeichen des eingelesenen Zeichensatzes.

2.1.6. Zeichensatz abspeichern
Mit dieser Option können Sie den Zeichensatz, der sich gerade im Apple-RAM befindet, auf Diskette abspeichern. Sie haben hierbei zwei Möglichkeiten:
– **BRUN-fähige FONT.LOADER-Datei**: Wollen Sie eine BRUN-fähige Zeichensatzdatei erstellen, so wählen Sie diese Funktion. FONT.LOADER speichert hierbei den Zeichensatz und das Maschinenprogramm zusammen auf Diskette ab. Diese Datei kann dann mit BRUN gestartet werden und lädt den Zeichensatz automatisch in den Imagewriter. Durch Verwendung einer BRUN-fähigen Zeichensatzdatei sparen Sie sich zum einen alle Eingaben, die Sie normalerweise in FONT.LOADER machen müssen, zum anderen können Sie so eine Pre-Boot-Diskette für Ihr Anwendungsprogramm erstellen: Initialisieren Sie eine Diskette, speichern Sie darauf die BRUN-fähige Zeichensatzdatei und schreiben Sie ein kleines Hello-Programm, das diese Datei startet. Sie müssen dann nur Ihre Pre-Boot-Diskette einlegen und den Imagewriter sowie den Apple einschalten. Der Zeichensatz wird automatisch beim Booten dieser Diskette in den Drucker übertragen. Alle FONT.LOADER-Einstellungen wie Zeichensatzformat und Darstellungsart werden in die Zeichensatzdatei übernommen.
Beachten Sie bitte: Die Zeichensatzdatei überträgt zwar den Zeichensatz in den Drucker, aktiviert ihn jedoch nicht. Dies müssen Sie selbst in Ihrem Hello- bzw. Anwendungsprogramm tun.
– **Zeichensatz allein abspeichern**: Mit dieser Option kann der Zeichensatz auf Diskette abgespeichert werden. Im Unterschied zu oben wird jedoch das Maschinenprogramm nicht mit abgespeichert, sondern nur die Bit-Mustertabelle des Zeichensatzes. Sie können diese Funktion dazu benutzen, um Zeichensätze zwischen Disketten zu übertragen oder konvertierte Zeichensätze (siehe 2.1.7.) abzuspeichern und dann weiterzubearbeiten.

2.1.7. Zeichensatzkonvertierung
Diese Funktion konvertiert Zeichensätze zwischen den beiden von FONT.LOADER akzeptierten Formaten. FONT.LOADER erkennt automatisch das Format des aktuellen Zeichensatzes und konvertiert es in das jeweils andere. Sie können mit dieser Funktion also Zeichensätze zwischen BITEDITOR und DOS Toolkit austauschen. Sollten Sie DOS Toolkit nicht besitzen, so ist diese Funktion dennoch sehr wichtig für Sie, wenn Sie die auf der Sammeldiskette enthaltenen Zeichensätze im DOS-Toolkit-Format mit dem BITEDITOR weiterbearbeiten wollen. Konvertieren Sie hierzu den DOS-Toolkit-Zeichensatz in das BITEDITOR-Format und speichern Sie anschließend den Zeichensatz auf Diskette ab. Nach dem Verlassen von FONT.LOADER können Sie diese Zeichensatzdatei dann weiterbearbeiten.

2.1.8. Informationen
Mit dieser Option erhalten Sie Informationen über FONT.LOADER und darüber, wie Sie die in den Imagewriter geladenen Zeichensätze aktivieren.

2.1.9. Ende
Mit dieser Wahl kehren Sie zum Applesoft-BASIC zurück und können nun Ihr Anwendungsprogramm starten.

2.2. FONT.LOADER.OBJ

Dieses Programm erledigt die eigentliche Arbeit von FONT.LOADER. Es rechnet die verschiedenen Zeichensatzformate in das Format des Imagewriters um und lädt den Zeichensatz anschließend in den Drucker. Mit „Zeichensatz zum Drucker übertragen" wird dieses Maschinenprogramm von FONT.LOADER aus gestartet.

2.3. FONT.CONVERT

Dieses Programm dient der Konvertierung der Zeichensatzformate mit der Funktion „Zeichensatzkonvertierung".

2.4. Die Zeichensätze

FONT.LOADER arbeitet mit Zeichensätzen im DOS-Toolkit- oder BITEDITOR-Format. Die Zeichensatztabellen werden von der Diskette geladen, dann von FONT.LOADER in das Druckerformat konvertiert und in den Drucker geladen. Genaueres über die Zeichensätze finden Sie im Abschnitt 3.1. Auf der Sammeldiskette befinden sich drei Zeichensatzfiles: DEUTSCH.FONT, dieser entspricht dem deutschen Apple-Bildschirm-Zeichensatz und läßt sich mit „ANIMATRIX" modifizieren. Der Zeichensatz ASCII.FONT entspricht dem amerikanischen Bildschirm-Zeichensatz. SONDERZEICHEN.FONT stellt Ihnen einige Sonderzeichen und die Mousetext-Zeichen zur Verfügung. Welche Zeichen die einzelnen Sätze enthalten, sehen Sie in dem Probeausdruck in **Abb. 1**.

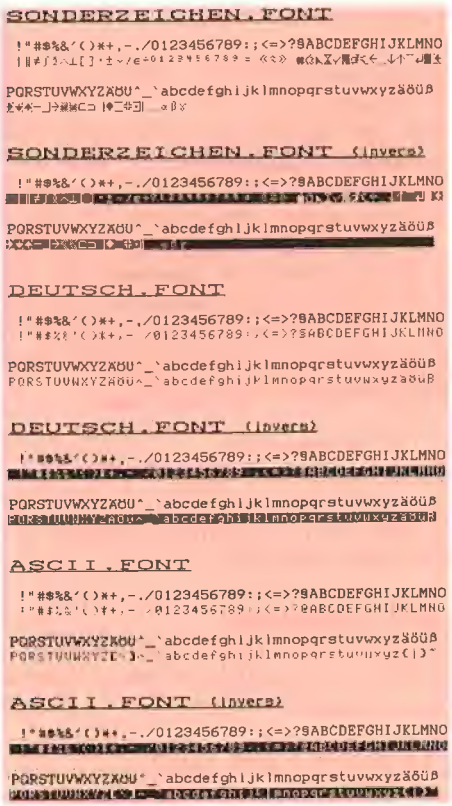


Abb. 1: Beispielausdruck

3. Die Ansteuerung der Sonderzeichen

3.1. Ansteuerung von Applesoft aus

Durch die Print-Anweisung
PRINT CHR$ (27); CHR$ (39)
wählen Sie den anwenderspezifischen Zeichensatz aus. Alle danach ausgegebenen Zeichen erscheinen im Muster dieses Zeichensatzes. Wollen Sie wieder normale Zeichen drucken, also den Standardzeichensatz verwenden, so benutzen Sie die folgende Anweisung:
PRINT CHR$ (27); CHR$ (36)
Hierdurch wird wieder auf den druckerinternen Zeichensatz zurückgeschaltet.

3.2. Ansteuerung von Pascal aus

Hier verwenden Sie die gleichen Steuercodes, die Sie durch entsprechende Write-Anweisungen an den Drucker übertragen.

3.3. Konfiguration von Appleworks

Um die Sonderzeichen in Appleworks drucken zu können, müssen Sie einen „Anderen Drucker" definieren. Gehen Sie hierzu in die Hauptauswahl. Tippen Sie nun 5 (Return), 7 (Return), 2 (Return). Sie müßten nun im Menü „Drucker hinzufügen" sein. Wählen Sie nun Option 12 „Anderer Drucker" und geben Sie einen Namen dafür ein, zum Beispiel „Imagewriter.SZ". Wählen Sie dann den Steckplatz aus, in dem die Druckerkarte steckt.
Nun beginnt die eigentliche Arbeit. Wählen Sie die Option „Steuerzeichen". Durch Eingabe von 3 (Return) können Sie die Steuerzeichen für Fettdruck, Index oder Exponent so definieren, daß anstatt Fettdruck, Index oder Exponent der umdefinierte Zeichensatz aktiviert wird. Wenn Sie im Menü „Fettdruck..." sind, können Sie die Steuerzeichen entweder auf Fettdruck, Index oder Exponent legen. Wählen Sie z.B. Fettdruck. Drücken Sie nun 1 (Return) für „Fettdruck ein" und tippen Sie die Steuerzeichen für Sonderzeichen ein, nämlich „ESC '" (ESC-Taste und dann „'"-Taste). Verlassen Sie „Fettdruck ein" mit Apfel-↑. Wählen Sie dann „Fettdruck aus" und geben Sie die Steuerzeichen „ESC $" ein. Mit Apfel-↑ verlassen Sie die Eingabe. Kehren Sie durch mehrmaliges Drücken von ESC zur Hauptauswahl zurück. Nun können Sie in der Textverarbeitung durch Eingabe von Ctrl-F die Sonderzeichen starten.
Sollten Sie genauere Informationen zum Definieren eines Druckers benötigen, so lesen Sie bitte im Appleworks-Handbuch nach.

4. Technische Informationen

Sie können FONT.LOADER ohne besondere Vorkenntnisse benutzen, doch viele Peeker-Leser wollen nicht nur vor einem fertigen Programm stehen, von dem sie zwar wissen, daß es, aber nicht wie es funktioniert. Für diese Leser möchte ich hier einige Grundlagen der zum Verständnis von FONT.LOADER notwendigen Dinge geben.

4.1. Zeichensätze

FONT.LOADER kann, wie oben schon erwähnt, zwei Zeichensatzformate verarbeiten:

4.1.1.DOS-Toolkit-Zeichensätze
Diese Zeichensätze werden mit dem DOS-Toolkit-Zeicheneditor „ANIMATRIX„ erstellt. Pro Zeichen belegt dieser Zeichensatz 8 Bytes. Diese 8-Byte-Gruppen sind so organisiert, daß sie direkt in den Hires-Speicher geladen werden können. Die oberste Zeile belegt also das relative Byte 0, die unterste das relative Byte 7. Da in der etwas kompliziert organisierten Apple-Hires-Grafik die Bytes spiegelverkehrt erscheinen, also Bit 0 links und Bit 6 rechts, sind auch alle Zeichen beim DOS Toolkit spiegelverkehrt abgelegt. Eine weitere Besonderheit: Das höchstwertige Bit eines jeden Bytes erscheint nicht auf dem Bildschirm. Es ist das Farb-Bit, das dazu dient, Zeilen eines Zeichens um einen halben Punkt nach rechts zu verschieben, um es in einer anderen Farbe erscheinen zu lassen. Dieses Bit darf also nicht an den Drucker übertragen werden. Der DOS-Toolkit-Zeichensatz beginnt mit der Bit-Tabelle für das Leerzeichen und endet mit der für das Delete-Zeichen. **Abb. 2** veranschaulicht den Aufbau der Zeichentabelle beim DOS-Toolkit am Beispiel eines „B". Man beachte, daß der Zeichensatz an die Adresse $8000 geladen wurde.

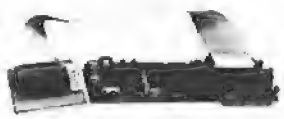
```
Adresse Inhalt Bitmuster

                BIT
                76543210
                ----------
$8110   $1E     !   **** ! 0 B
$8111   $21     !  *    * ! 1 Y
$8112   $21     !  *    * ! 2 T
$8113   $1E     !   **** ! 3 E
$8114   $21     !  *    * ! 4
$8115   $21     !  *    * ! 5
$8116   $1E     !   **** ! 6
$8117   $00     !        ! 7
                ----------
```

Abb. 2: „B" bei DOS Toolkit

4.1.2. BITEDITOR-Zeichensätze
Diese mit dem BITEDITOR erstellten Zeichensätze dienen primär dem Brennen von verbesserten 80-Zeichen-EPROMs. Sie sind so organisiert, daß die Video-Elektronik direkt darauf zugreifen kann. Die Zeichenmatrix beträgt 8 * 9 Punkte, wobei normale Zeichen die unteren 8 Punkte benutzen. Da der Imagewriter nur eine Zeichenmatrix von 8 * 8 Punkten hat, muß man auf die oberste Zeile verzichten. Die oberste Zeile für den Drucker beginnt also beim relative Byte 1. Der Speicherplatzverbrauch ist bei diesem Zeichensatzformat doppelt so groß, obwohl die Zeichenmatrix nur eine Zeile größer ist. Die Zeichenlänge beträgt 16 Bytes, da pro Zeichen jeweils 7 Bytes ungenutzt bleiben. Beim BITEDITOR-Format beginnt der Zeichensatz auch nicht mit dem Leerzeichen, sondern mit dem NUL-Zeichen (Ctrl-§). Um den Zeiger auf das Leerzeichen zu setzen, habe ich mich eines kleinen Tricks bedient: Ich erhöhe das höherwertige Byte der Zeichensatzadresse um 2, was gerade 32 * 16 = 512 Bytes entspricht. Ein weiterer kleiner Patch ist notwendig, um immer die zweitoberste Zeile eines Zeichens mit 0 bezeichnen zu können: Wenn man den niederwertigen Teil der Zeichenadresse um 1 erhöht, so bezeichnet der Byte-Index 0 immer das zweite Byte des Zeichens. Die Zeichen stehen beim BITEDITOR nicht spiegelverkehrt im RAM, so daß man die LSR-Anweisung in der Convert-Routine durch eine ASL-Anweisung ersetzen muß, um eine richtige Zuordnung der Zeichen-Bits zu

CSW UPC II
(eine universelle Programmierkarte)

für alle APPLE-II- und kompatiblen Rechner



- jetzt auch mit 12,5 V Brennspannung
- Komfortable Bedienungssoftware mit Menüsteuerung DOS 3.3 – keine Umschaltung über DIP-Switches auf der Karte!
- Anzeige und Sockel auf externer Platine – 28poliger Nullkraftsockel – 500-mm-Flachbandkabel
- EPROM-Typen 2516, 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A

DM 580.–

CSW 72 I/O-Port-Card

- 9 programmierbare 8-Bit-I/O-Ports
- lauffähig unter allen Betriebssystemen

DM 350.–

WEISS COMPUTER Dipl.-Psych. Karl-Heinz Weiß
Am Wiesenhof 17, 2940 Wilhelmshaven, Tel. 0 44 21/8 31 79

APPLE & CP/M-80 & MS-DOS SOFTWARE & HARDWARE

z. B. für APPLE II und Kompatible

Wir liefern die RAM-Karte (AE) für den Apple IIe mit max. 3 MB (Appleworks mit mehr als 2 MB)! 64-K-Ausf.	DM 650,–
SpeeDemon 3.56 MHz Coproz. für II+/e (McT)	DM 980,–
UPC-Programmer-Card 2716-128 komfortabel	DM 580,–
72 I/O Port Card programmierbar DOS+CP/M	DM 350,–
AD 16 Ch. 12 Bit, schnell! (Applied Eng.)	DM 1150,–
PKASO/U-Printer-Karte (IS)	DM 550,–
CP/M-Plus-Card, 6 MHz, 64 K, CP/M 3.0 (ALS)	DM 1250,–
NicePrint-Printer-Karte (Spies Lobarat.)	DM 490,–
Timemaster II H. O., die Uhrenkarte! (AE)	DM 540,–
Softerm 2 II+/e/c (Softronics)	DM 800,–
ELF kompl. Statistik-Software (TWG)	DM 800,–
Prime-Plotter-Grafik-Software (Primesoft)	DM 960,–
TTL-IC-Tester inkl. Software	DM 320,–
Memory-Tester & Eprom-Writer/Tester	DM 450,–
Z-RAM 512 K für APPLE IIc (AE)	DM 1350,–

z. B. für IBM und Kompatible

APPLE Turnover (Vertex) Lesen/Schreiben von Apple Disks im IBM PC & Komp.	DM 1200,–
XENO-COPY plus (Vertex) Lesen/Schreiben div. CP/M & MS-DOS Formate im IBM	DM 600,–
ELF PC kompl. Statistik Software (TWG)	DM 800,–
PROM Blaster 28-Pin (Apparat Inc.)	DM 620,–

z. B. für alle Systeme

Printerchanger 3 parall. Drucker auf 1 Micro inkl. Kabel/Netzteil (Keyzone)	DM 570,–
Printersharer 3 Micros auf 1 parall. Drucker inkl. Kabel/Netzteil (Keyzone)	DM 460,–
Shufflebuffer 64 K (IS)	DM 1350,–

Wir sind Import-Spezialisten und bieten Ihnen eine große Auswahl an Software und Hardware bedeutender Hersteller aus den USA und England. Informationen gegen DM 3.– in Briefmarken.

WEISS COMPUTER Dipl.-Psych. Karl-Heinz Weiß
Am Wiesenhof 17, 2940 Wilhelmshaven, Tel. 0 44 21/8 31 79

Verpassen Sie Ihrem Apple den richtigen Biß mit der C86 Steckkarte

Die C86-Karte ist ein kompletter 16-Bit Mikrocomputer mit eigenem Speicherbereich (512KB) und Schnittstelle zum Applebus.

Die Erweiterung bietet Ihnen:

- einen leistungsfähigen 16-Bit Mikrocomputer mit Intel 8086CPU, es sind fast alle betriebssystemgestützten IBM PC Programme auf der C86-Karte lauffähig
- serienmäßig 128KB oder 512KB Speicherkapazität
- ein spezielles Applebusinterface für den sicheren Betrieb
- alle notwendigen Teile auf einer Apple-Slotkarte

Mit dieser Erweiterung entspricht Ihr Apple dem neuesten Stand der Technik.

C86 mit 128K **nur 598,– DM**
C86 mit 512K **nur 998,– DM**

D86, Diskettencontroller zur C86-Karte

- ermöglicht das Einlesen aller handelsüblichen Diskettenformate (MS-DOS, CPM, Apple-DOS) in den Apple od. compatible Computer.

D86 298,– DM

IBM AT COMPATIBLER MIKRO-COMPUTER **ab 7500,–DM**
IBM XT COMPATIBLER MIKRO-COMPUTER **ab 1600,–DM**
HANDBUCH FÜR IBM COMPATIBLE COMPUTER 68,–DM

Händleranfragen erwünscht: Anton Peter & Partner
Brüsseler Straße 16
1000 Berlin 65

alle Preise incl. 14% MWST, zuzgl. Verpackung u. Porto
Apple und IBM sind eingetragene Markenzeichen

Microcomputer-Reparaturen
(Preise inkl. Bauteile)

Apple II

Motherboard . . . . . DM 150,–
Interfacekarten . . . . . DM 50,–
Diskettenlaufwerke . . . DM 120,–

Druckerreparaturen (z. B. Epson, Star) sowie Reparaturen anderer Geräte auf Anfrage

P.S.: ROLL-MAUS für Apple II . DM 195,–

Dipl.-Math. W. Dederichs

Büro für Software- und Hardware-Entwicklung
Hackstückstr. 11, 4320 Hattingen,
Telefon 0 23 24/5 22 40

Z80+ Card

Die leistungsfähige und vielseitige Z80 Karte für Ihren Apple II

Hardware:
+ ALS CP/M-Card- und (!) SoftCard-kompatibel
+ 8MHz Taktfrequenz (ohne Waitzyklen) – dadurch werden Programme bis zu 4x schneller als bisher (2MHz Taktfrequenz im SoftCard-Modus).
+ Interrupts sind für die beiden Betriebsmodi der Z80+ Card getrennt zu- und abschaltbar

Software:
+ CP/M 2.20/2.23 Pseudodisktreiber, der die eigenen 64K RAM der Z80+ Card im SoftCard-Modus nutzt
+ CP/M-Patch generiert aus Ihrem SoftCard CP/M 2.20 ein CP/M 2.2 für den schnellen Taktmodus der Z80+ Card mit überragenden Eigenschaften:
Bis zu 4x schnellerer Disk-Zugriff (Read), integrierte Pseudodisk mit 32/48/110K bei einem Apple mit 48/64/128K RAM, Unterstützung von Disk-Laufwerken bis 640K an den verschiedensten Controllern, Unterstützung der gängigsten RAM-Karten als Pseudodisk, schnelle Bildschirmausgabe, bedienerfreundlich durch neue und erweiterte Befehle in der Kommandoebene, 58K TPA
+ Der Preis: 696.-- DM (= 610,53 DM ohne Mwst)

Uwe Zimmermann – Microcomputerentwicklungen
Schwalbenstraße 30 – 6090 Rüsselsheim
Telefon: 06142 / 56 34 56

Berlin im Apple II+ 384 x 288 Bildpunkte.

VIDEO-1000

DIGITIZER ZUM APPLE IIE

INTERFACE+ SOFTWARE **295,-DM**

* Auflösung 384x288 Bildpunkte
* für TV, Recorder und Kamera
* Aufnahmezeit 20 msec
* Umrechnung auf HIRES-Seite

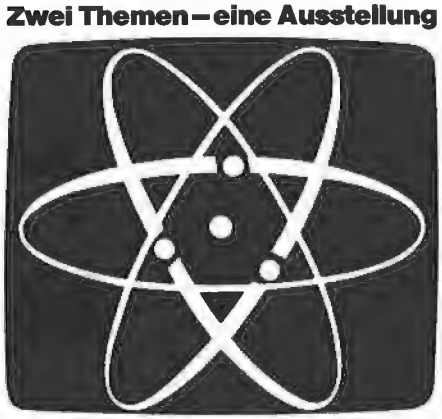
Erweiterte Software z.B. Bilder mit bis zu 500.000 Bildpunkten, 7 Graustufen, Double Hires, Kurvenerstellung etc auf Anfrage.

256 K RAM-Karte mit Software ..... 495,- DM
1024K RAM-Karte mit Software ..... 1295,- DM
16 K AKKU-RAM-Karte mit Software ..... 195,- DM
128K AKKU-RAM-Karte mit Software ..... 595,- DM

Demodisk gegen Einsendung von 10 DM oder V-Scheck. Info gratis. Versand p.NN. oder beim Fachhändler.

Ing.Büro M.Fricke, Neue Str.13, 1000 Berlin 37
Telefon: 030 / 801 50 52

Zwei Themen – eine Ausstellung



Hobby-tronic

9. Ausstellung für Funk- und Hobby-Elektronik

COMPUTER-SCHAU

2. Ausstellung für Computer, Software und Zubehör

Dortmund
23.–27. April 1986

Die umfassende Marktübersicht für Hobby-Elektroniker und Computer-Anwender, klar gegliedert:

In Halle 5 das Angebot für CB- und Amateurfunker, Videospieler, DX-er, Radio-, Tonband-, Video- und TV-Amateure, für Elektro-Akustik-Bastler und Elektroniker. Mit dem Actions-Center und Laborversuchen, Experimenten, Demonstrationen und vielen Tips.

In Halle 4 das Super-Angebot für Computer-Anwender in Hobby, Beruf und Ausbildung. Dazu die „Computer-Straße", als Aktionsbereich, der Wettbewerb „Jugend programmiert" und der Stand des WDR-Computer-Clubs.

Ausstellungsgelände Westfalenhallen Dortmund täglich 9.00-18.00 Uhr

erhalten. Anders als beim DOS Toolkit darf man hier nicht das höchstwertige Bit ausmaskieren, da es hier einen Zeichenpunkt enthalten kann. Auch hier sei der Aufbau eines „B" in **Abb. 3** verdeutlicht. Der Zeichensatz ist wieder ab $8000 in das Apple-RAM geladen worden.

```
Adresse Inhalt Bitmuster

                 BIT
                 76543210
                ----------
$8420    $00    !        !
$8421    $7C    ! *****  ! 0 B
$8421    $42    ! *    * ! 1 Y
$8421    $42    ! *    * ! 2 T
$8421    $7C    ! *****  ! 3 E
$8421    $42    ! *    * ! 4
$8421    $42    ! *    * ! 5
$8421    $7C    ! *****  ! 6
$8421    $00    !        ! 7
                ----------
```

Abb. 3: „B" bei BITEDITOR

4.1.3. Imagewriter-Zeichenformat
Beim Downloading in den Imagewriter werden die Zeichen nicht zeilenweise übertragen, sondern das Bit-Muster wird spaltenweise als ASCII-Zeichen codiert an den Drucker gesendet. Das höchstwertige Bit bezeichnet hier die unterste Drucknadel. Das erste ASCII-Zeichen entspricht der linken Zeichenspalte, das achte der rechten. Das „B" wird also nach dem Muster in **Abb. 4** übertragen.

```
BYTE
01234567
----------
! *****  ! 0 B
! *    * ! 1 I
! *    * ! 2 T
! *****  ! 3
! *    * ! 4
! *    * ! 5
! *****  ! 6
!        ! 7
----------

Bytefolge:

$00 $7F $49 $49 $49 $49 $36 $00
```

Abb. 4: „B" bei Downloading

4.2. Beschreibung der Programme

4.2.1. Steuerprogramm FONT.LOADER
Das Applesoft-Programm dient dazu, dem Benutzer die Bedienung von FONT.LOADER durch Menüsteuerung zu erleichtern. Es startet die verschiedenen Maschinenprogramme mit korrekt gesetzten Parametern und stellt darüber hinaus noch weitere Funktionen wie das Abspeichern von Zeichensätzen zu Verfügung. Das Applesoft-Steuerprogramm belegt den normalen Speicherbereich von $803 bis LOMEM.

4.2.2. Maschinenprogramm FONT. LOADER.OBJ
Dieses Programm dient der Übertragung der Zeichensatzdatei an den Imagewriter. Es initialisiert das Downloading und überträgt dann mit einer speziellen Ausgaberoutine, die im Gegensatz zur BASIC-Ausgaberoutine 8-Bit-Zeichen verarbeiten kann, die Zeichendaten an den Drucker. Vor dem Aufruf von FONT.LOADER.OBJ müssen die Flags INVERS ($00FA = 250) und TOOLKIT ($00FB = 251) ensprechend den Kommentaren im Assemblerlisting korrekt gesetzt werden. Die von FONT.LOADER.OBJ belegten Speicherbereiche sind aus dem kommentierten Assemblerlisting ersichtlich. Durch Neuassemblierung kann FONT.LOADER.OBJ auch an andere Speicherbereiche gelegt werden, jedoch müssen dann alle CALLs im Steuerprogramm und die Startadresse des Zeichensatzes ebenfalls geändert werden.

4.2.3. Maschinenprogramm FONT. CONVERT
Dieses Programm konvertiert Zeichensätze zwischen den beiden von FONT.LOADER akzeptierten Zeichensatzformaten. Der Aufruf erfolgt über die Funktion „Zeichensatzkonvertierung" des Applesoft-Steuerprogramms FONT.LOADER. Bei der Konvertierung wird automatisch das Zeichensatzformat-Flag TOOLKIT angepaßt. Die Speicherbelegung kann man aus dem Assemblerlisting entnehmen.

4.2.4. Zeichensätze
Die Zeichensätze werden standardmäßig vom Steuerprogramm nach $8000 geladen, können aber nach einem kleinen Patch im Maschinenprogramm in der Initialisierungsroutine auch in andere Speicherbereiche geladen werden. DOS-Toolkit-Zeichensätze belegen $300 Bytes, BITEDITOR-Zeichensätze wegen ihrer anderen Organisation $800 Bytes ohne bzw. $1000 Bytes mit inversem Zeichensatz.
Um FONT.LOADER auszuprobieren, können Sie den SONDERZEICHEN.FONT eingeben. Mit „BSAVE SONDERZEICHEN.FONT, A$8000, L$0300" speichern Sie ihn auf Diskette ab.

4.3. Downloading beim Imagewriter

Von Applesoft aus lassen sich nicht ohne weiteres anwenderspezifische Zeichensätze in den Imagewriter laden, da die BASIC-Ausgaberoutine bei jedem ausgegebenen Zeichen das höchstwertige Bit setzt. Alle ausgegebenen Zeichen haben also ASCII-Codes im Bereich von 128 bis 255. Für das Programmieren von Zeichensätzen hat dies folgende Konsequenzen: Man definiert nicht wie gewünscht Low-ASCII-Zeichen im Bereich von 32 bis 126, sondern High-ASCII-Zeichen mit Codes im Bereich 160 bis 254. Dies wäre nicht weiter tragisch, da man diese Zeichen mit „ESC *" anwählen kann. Im High-ASCII-Bereich lassen sich jedoch nicht die vollen 96 ASCII-Zeichen speichern. Man müßte also auf einige Buchstaben verzichten. Weiterhin werden von der BASIC-Ausgaberoutine auch die höchstwertigen Bits der Daten-Bytes auf 1 gesetzt, was zur Folge hat, daß jedes selbst programmierte Zeichen auf dem Drucker unterstrichen erscheint. Beide Probleme – das des Speicherbereichs und das der Unterstreichung – lassen sich mit einem Maschinenprogramm umgehen, das Low-ASCII sendet und die Daten-Bytes wirklich so ausgibt, wie sie im Apple-Speicher stehen. Mit der Ausgabe-Routine von Dieter Geiß (s. Superdump, Peeker 6/85, S. 22), die ich in FONT.LOADER verwendet habe, läßt sich dies erreichen.

5. Schlußbemerkung

Ich hoffe, daß Sie durch FONT.LOADER von den mühsamen Zeichensatzberechnungen entlastet werden. Wenn Sie nicht wissen, was Sie mit den vielen gesparten Stunden anfangen sollen, können Sie ja Ihre freigewordene Zeit dazu nutzen, Macintosh-Zeichensätze auf dem Imagewriter zu implementieren. Vielleicht entwickeln Sie auch ein Programm zur direkten Übertragung von Mac-Zeichen über die seriellen Ports? Viel Spaß auf jeden Fall beim Ausprobieren der neuen Zeichen!

Kurzhinweise

1. Zweck:
Hilfsprogramm zum Übertragen von DOS-Toolkit- und BITEDITOR-Zeichensätzen an den Imagewriter.
2. Konfiguration:
Apple II, II+, IIe mit Super Serial Card oder IIc
Imagewriter I oder II
DOS 3.3 oder ProDOS
3. Test
RUN FONT.LOADER, dann:
mit „4" Zeichensatz an Drucker übertragen,
mit „5" Probeausdruck erstellen
4. Sammeldisk
FONT.LOADER
(Applesoft-Steuerprogramm)
T.FONT.LOADER.OBJ
T.FONT.CONVERT
(Big-Mac-Quelltexte)
FONT.LOADER.OBJ
FONT.CONVERT
(Maschinenprogramme)
SONDERZEICHEN.FONT
DEUTSCH.FONT
ASCII.FONT
(Zeichensätze im DOS-Toolkit-Format)

FONT.LOADER

```
1000 REM FONT.LOADER
1010 REM Carl Frieder Mahr
1020 REM 1986
1030 :
1040 REM Initialisierung
1050 :
1060 D$ = CHR$ (4): POKE 768,0
1070 A2C = PEEK ( - 1101) = 6 AND PEEK ( - 1088) = 0
1080 ZS$ = "SONDERZEICHEN.FONT"
1090 ZF$(0) = "BITEDITOR":ZF$(1) = "DOS Toolkit"
1100 PRINT D$"BRUNFONT.LOADER.OBJ"
1110 PRINT D$"BLOAD"ZS$",A$8000"
1120 :
1130 REM Menü und Statusanzeigen
1140 :
1150 PRINT CHR$ (21): PRINT D$"PR#0": TEXT : HOME
1160 PRINT "FONT.LOADER": PRINT "============"
1170 PRINT : PRINT "Zeichensatz-Utility für den Imagewriter"
1180 PRINT : PRINT "Zeichensatz      : ";ZS$
1190 PRINT : PRINT "Zeichensatzformat: ";ZF$( PEEK (251))
1200 PRINT "Darstellungsart  : ";
1210 IF PEEK (250) = 0 THEN PRINT "normal"
1220 IF PEEK (250) = 1 THEN PRINT "invers"
1230 PRINT : PRINT "1. Zeichensatzformat ändern"
1240 PRINT "2. Darstellungsart ändern"
1250 PRINT "3. Zeichensatz von Diskette laden"
1260 PRINT "4. Zeichensatz zum Drucker übertragen"
1270 PRINT "5. Probeausdruck"
1280 PRINT "6. Zeichensatzdatei abspeichern"
1290 PRINT "7. Zeichensatzkonvertierung"
1300 PRINT "8. Informationen": PRINT : PRINT "9. Ende"
1310 0 = 9: GOSUB 2260
1320 IF W = 9 THEN HOME : END
1330 ON W GOSUB 1380,1430,1470,1580,1690,1870,2010,2090
1340 GOTO 1150
1350 :
1360 REM Zeichensatzformat ändern
1370 :
1380 POKE 251, NOT ( PEEK (251))
1390 ZS$ = "(kein Zeichensatz)": RETURN
1400 :
1410 REM Darstellungsart ändern
1420 :
1430 POKE 250, NOT ( PEEK (250)): RETURN
1440 :
1450 REM Zeichensatz-File laden
1460 :
1470 HOME : PRINT "Zeichensatz von Diskette laden": PRINT
1480 PRINT "Catalog durch Eingabe von <Return>"
1490 PRINT "Zurück zum Menü mit <Ctrl-E>"
1500 PRINT : INPUT "Zeichensatzname: ";ZZ$
1510 IF ZZ$ = CHR$ (5) THEN RETURN
1520 IF ZZ$ = "" THEN HOME : PRINT D$"CATALOG": GOTO 1500
1530 PRINT D$"BLOAD"ZZ$",A$8000"
1540 ZS$ = ZZ$: RETURN
1550 :
1560 REM Zeichensatz in Druckerspeicher laden
1570 :
1580 HOME : PRINT "Zeichensatz zum Drucker übertragen"
1590 IF ZS$ < > "(kein Zeichensatz)" THEN 1620
1600 PRINT : PRINT "Kein Zeichensatz vorhanden!": PRINT
1610 PRINT "Bitte eine Taste drücken ";: GET A$: RETURN
1620 PRINT : PRINT "Imagewriter bereit? (j/n) ";: GET W$
1630 IF W$ < > "j" AND W$ < > "J" THEN RETURN
1640 PRINT W$: PRINT : PRINT "Zeichensatz wird übertragen..."
1650 CALL 32538: RETURN
1660 :
1670 REM Probeausdruck
1680 :
1690 PRINT : PRINT D$"PR#1"
1700 PRINT SPC( 16) CHR$ (27)"X" CHR$ (14)ZS$ CHR$ (15);
1710 IF PEEK (250) THEN PRINT "  (invers)";
1720 PRINT CHR$ (27)"Y": PRINT
1730 PRINT SPC( 16): GOSUB 1800: PRINT : PRINT SPC( 16)
1740 PRINT CHR$ (27)"'";: GOSUB 1800: PRINT CHR$ (27)"$"
1750 PRINT
1760 PRINT SPC( 16): GOSUB 1820: PRINT : PRINT SPC( 16)
1770 PRINT CHR$ (27)"'";: GOSUB 1820: PRINT CHR$ (27)"$"
1780 PRINT : PRINT : PRINT D$"PR#0": RETURN
1790 :
1800 FOR I = 32 TO 79: PRINT CHR$ (I);
1810 NEXT : RETURN : REM " " bis "O"
1820 FOR I = 80 TO 126: PRINT CHR$ (I);
1830 NEXT : RETURN : REM "P" bis "~"
1840 :
1850 REM Font-Datei erstellen
1860 :
1870 HOME : PRINT "Zeichensatzdatei abspeichern"
1880 PRINT : PRINT "1. BRUN-fähige Font.Loader-Datei"
1890 PRINT "2. nur Zeichensatz abspeichern"
1900 0 = 2: GOSUB 2260
1910 PRINT : PRINT "Ende mit <Return>": PRINT
1920 INPUT "Dateiname: ";ZZ$: IF ZZ$ = "" THEN RETURN
1930 A = 32512 + 256 * (W - 1): REM für BSAVE
1940 L = 2304 - 256 * (W - 1) - 1280 * PEEK (251)
1950 POKE 32513, PEEK (250): POKE 32517, PEEK (251)
1960 POKE 32537,234: PRINT D$"BSAVE";ZZ$;",A";A;",L";L
1970 RETURN
1980 :
1990 REM Zeichensatzkonvertierung
2000 :
2010 HOME : PRINT "Zeichensatzkonvertierung": PRINT
2020 PRINT "Konvertierung des Zeichensatzes:": PRINT
2030 PRINT ZF$( PEEK (251));" -> ";ZF$( NOT ( PEEK (251)))
2040 IF PEEK (768) = 165 THEN CALL 768: RETURN
2050 PRINT D$;"BRUN FONT.CONVERT": RETURN
2060 :
2070 REM Informationen und Hilfe
2080 :
2090 HOME : PRINT "Informationen": PRINT
2100 PRINT "FONT.LOADER überträgt Zeichensätze aus"
2110 PRINT "DOS Toolkit oder dem BITEDITOR von"
2120 PRINT "Joachim Klamt aus Peeker 7/85."
2130 PRINT : PRINT "Die Zeichensätze bleiben im Image-"
2140 PRINT "writer, bis Sie ihn ausschalten."
2150 PRINT : PRINT "Mit 'PRINT CHR$(27);CHR$(39)' können Sie"
2160 PRINT "diese Zeichensätze anwählen."
2170 PRINT : PRINT "Mit Appleworks oder anderen Programmen"
2180 PRINT "ist es so möglich, Sonderzeichen zu"
2190 PRINT "drucken."
2200 PRINT : PRINT "Abgeschaltet werden die Zeichensätze"
2210 PRINT "durch 'PRINT CHR$(27);CHR$(36)'.": PRINT
2220 PRINT "Weiter mit beliebiger Taste ";: GET A$: RETURN
2230 :
2240 REM Tastatureingabe
2250 :
2260 PRINT : PRINT "Bitte wählen Sie ";
2270 GET W$:W = VAL (W$): IF W < 1 OR W > 0 THEN 2270
2280 PRINT W: RETURN
```

SONDERZEICHEN.FONT

BSAVE SONDERZEICHEN.FONT, A$8000, L$0300

```
$8000: 00 00 00 00 00 00 00 00
$8008: 08 08 08 08 08 08 08 08
$8010: 14 14 14 14 14 14 14 14
$8018: 10 10 3E 08 3E 04 04 00
$8020: 30 08 08 08 08 08 08 06
$8028: 14 08 14 12 14 08 14 00
$8030: 00 00 08 14 22 41 00 00
$8038: 00 08 08 08 08 08 7F 00
$8040: 1C 04 04 04 04 04 04 1C
$8048: 1C 10 10 10 10 10 10 1C
$8050: 00 00 18 18 00 00 00 00
$8058: 08 08 3E 08 08 00 3E 00
$8060: 00 00 00 4C 32 00 00 00
$8068: 00 20 20 10 10 08 4A 00
$8070: 00 00 3C 02 3E 02 3C 00
$8078: 00 08 00 3E 00 08 00 00
$8080: 06 09 09 09 06 00 00 00
$8088: 02 03 02 02 07 00 00 00
$8090: 07 08 04 02 0F 00 00 00
$8098: 07 08 07 08 07 00 00 00
$80A0: 05 05 0F 04 04 00 00 00
$80A8: 0F 01 07 08 07 00 00 00
$80B0: 06 01 07 09 06 00 00 00
$80B8: 0F 08 04 02 02 00 00 00
$80C0: 06 09 06 09 06 00 00 00
$80C8: 06 09 0E 08 06 00 00 00
$80D0: 00 00 7A 00 7A 00 00 00
$80D8: 00 00 00 00 00 00 00 00
$80E0: 08 24 12 09 12 24 08 00
$80E8: 00 4C 32 00 4C 32 00 00
$80F0: 08 12 24 48 24 12 08 00
$80F8: 00 00 00 00 00 00 00 00
$8100: 10 08 36 7F 3F 3F 7E 36
$8108: 10 08 36 41 21 21 4A 36
$8110: 00 00 02 06 0E 1E 36 42
$8118: 7F 22 14 08 08 14 2A 7F
$8120: 00 00 40 20 11 0A 04 04
$8128: 7F 7F 3F 5F 6E 75 7B 7B
$8130: 70 60 7E 31 79 30 3F 02
$8138: 00 18 07 00 07 0C 08 70
$8140: 08 04 02 7F 02 04 08 00
$8148: 00 00 00 00 00 00 00 2A
$8150: 08 08 08 08 49 2A 1C 08
$8158: 08 1C 2A 49 08 08 08 08
$8160: 7F 00 00 00 00 00 00 00
```

```
$8168: 40 40 40 44 46 7F 06 04
$8170: 3F 3F 3F 3F 3F 3F 3F 3F
$8178: 13 18 1C 7E 1C 18 10 6F
$8180: 64 0C 1C 3F 1C 0C 04 7B
$8188: 40 48 08 7F 3E 1C 48 40
$8190: 40 48 1C 3E 7F 08 48 40
$8198: 00 00 00 7F 00 00 00 00
$81A0: 40 40 40 40 40 40 40 7F
$81A8: 08 10 20 7F 20 10 08 00
$81B0: 2A 55 2A 55 2A 55 2A 55
$81B8: 55 2A 55 2A 55 2A 55 2A
$81C0: 00 3E 41 01 01 01 7F 00
$81C8: 00 00 3F 40 40 40 7F 00
$81D0: 40 40 40 40 40 40 40 40
$81D8: 08 1C 3E 7F 3E 1C 08 00
$81E0: 7F 00 00 00 00 00 00 7F
$81E8: 14 14 77 00 77 14 14 00
$81F0: 7F 40 40 4C 4C 40 40 7F
$81F8: 01 01 01 01 01 01 01 01
$8200: 00 00 00 00 00 00 55 00
$8208: 00 00 00 2C 12 12 2C 00
$8210: 18 24 24 14 24 24 14 04
$8218: 00 00 62 14 08 14 14 08
$8220: 00 00 00 00 00 00 00 00
$8228: 00 00 00 00 00 00 00 00
$8230: 00 00 00 00 00 00 00 00
$8238: 00 00 00 00 00 00 00 00
$8240: 00 00 00 00 00 00 00 00
$8248: 00 00 00 00 00 00 00 00
$8250: 00 00 00 00 00 00 00 00
$8258: 00 00 00 00 00 00 00 00
$8260: 00 00 00 00 00 00 00 00
$8268: 00 00 00 00 00 00 00 00
$8270: 00 00 00 00 00 00 00 00
$8278: 00 00 00 00 00 00 00 00
$8280: 00 00 00 00 00 00 00 00
$8288: 00 00 00 00 00 00 00 00
$8290: 00 00 00 00 00 00 00 00
$8298: 00 00 00 00 00 00 00 00
$82A0: 00 00 00 00 00 00 00 00
$82A8: 00 00 00 00 00 00 00 00
$82B0: 00 00 00 00 00 00 00 00
$82B8: 00 00 00 00 00 00 00 00
$82C0: 00 00 00 00 00 00 00 00
$82C8: 00 00 00 00 00 00 00 00
$82D0: 00 00 00 00 00 00 00 00
$82D8: 00 00 00 00 00 00 00 00
$82E0: 00 00 00 00 00 00 00 00
$82E8: 00 00 00 00 00 00 00 00
$82F0: 00 00 00 00 00 00 00 00
$82F8: 00 00 00 00 00 00 00 00
```

FONT.LOADER

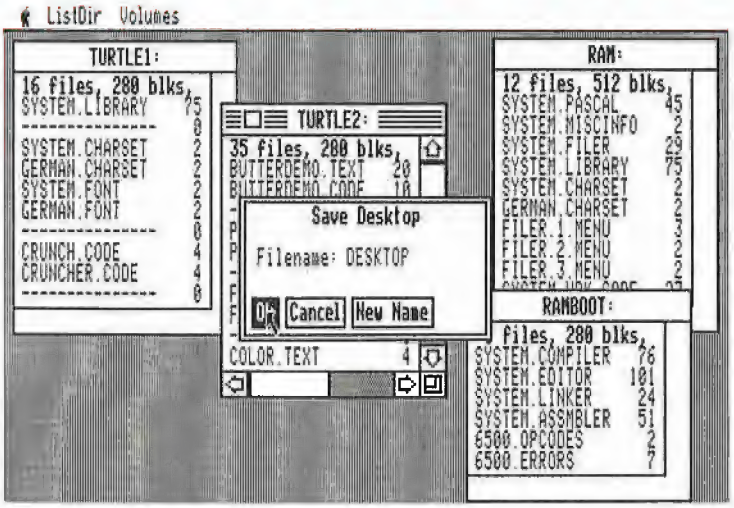
BSAVE FONT.LOADER, A$7F00, L$00FC

```
1   ********************************
2   *                              *
3   *         FONT.LOADER          *
4   *                              *
5   *   Carl Frieder Mahr, 1986    *
6   *                              *
7   ********************************
8
9   * Gerätekonfiguration:
10  *
11  * Apple II, II+, IIe mit SSC in Slot 1
12  * Apple IIc mit Drucker an Port 1
13  * Apple-Imagewriter I oder II
14
15          ORG  $7F00
16
17  * Label-Definitionen
18
19  SPACE   EQU  ' '
20  DEL     EQU  $7F
21  CTRLD   EQU  $84
22  ESC     EQU  $9B
23  OPASL   EQU  $0A        ;Opcode ASL
24  OPLSR   EQU  $4A        ;Opcode LSR
25
26  * Invers: 0: Zeichensatz normal laden
27  *         1: Zeichensatz invers laden
28
29  INVERS  EQU  $FA        ;Flag
30
31  * Toolkit: 0: BITEDITOR-Zeichensatz
32  *          1: DOS-Toolkit-Zeichensatz
33
34  TOOLKIT EQU  $FB        ;Flag
35
36  * Ready: IIc Port: $20, SSC: $40
37
38  READY   EQU  $FC        ;Ready-Bit
39
40  * Offset: DOS-Toolkit-Zeichensatz: 8
41  *         Biteditor-Zeichensatz  : 16
42
43  OFFSET  EQU  $FD
44
45  CHAR    EQU  $CE        ;bearb. Zeichen
46  WIDTH   EQU  $08        ;Zeichenbreite
47  IND     EQU  $06        ;Basis
48  NO      EQU  $06F8      ;$10 für Slot 1
49  MISCFLG EQU  $07F9      ;SSC Modebits
50  FONT    EQU  $8000      ;Tabellenbeginn
51  ID1     EQU  $FBB3      ;ID-Byte #1
52  ID2     EQU  $FBC0      ;ID-Byte #2
53
54  ********************************
55  *     Initialisierungsteil     *
56  ********************************
57
58  * Flags auf DOS-TOOLKIT-Zeichensätze
59  * und normale Darstellung setzen
60
7F00: A2 00     61  INIT    LDX  #$00
7F02: 86 FA     62          STX  INVERS
7F04: A2 01     63          LDX  #$01
7F06: 86 FB     64          STX  TOOLKIT
65
66  * Patch für Status-Register-Unterschied
67  * bei SSC und IIc Ports
68
7F08: A9 40     69          LDA  #$40
7F0A: AE B3 FB  70          LDX  ID1
7F0D: E0 06     71          CPX  #$06       ;ID-Bytes
7F0F: D0 06     72          BNE  NOT2C      ;für IIc
7F11: AE C0 FB  73          LDX  ID2        ;überprüfen
7F14: D0 01     74          BNE  NOT2C
7F16: 4A        75          LSR             ;Ready-Bit IIc
7F17: 85 FC     76  NOT2C   STA  READY
7F19: 60        77          RTS
78
79  ********************************
80  *       Ausführungsteil        *
81  ********************************
82
83  * Super Serial Card initialisieren
84
7F1A: 20 CC 7F  85  PGMSTART JSR  INITSSC
86
87  * Zeichensatzbeginn initialisieren
88
7F1D: A9 00     89          LDA  #<FONT
7F1F: 85 06     90          STA  IND
7F21: A9 80     91          LDA  #>FONT
7F23: 85 07     92          STA  IND+1
93
94  * Zeichenoffset und Opcode für Umformung
95  * des Zeichensatz-Bytes in Imagewriter-
96  * Bytes initialisieren. DOS Toolkit: LSR,
97  * BITEDITOR: ASL
98
7F25: A9 08     99          LDA  #$08
7F27: 85 FD     100         STA  OFFSET
7F29: A9 4A     101         LDA  #OPLSR
7F2B: 8D 5D 7F  102         STA  CONVERT2
7F2E: A5 FB     103         LDA  TOOLKIT
7F30: D0 0D     104         BNE  INIDWNLD
105
106 * Oberste Linie bei BITEDITOR-Zeichen-
107 * sätzen fällt weg, da nur 8 Drucknadeln
108 * zur Verfügung stehen. Der Zeichenoffset
109 * ist 16 Bytes. Die ersten $200 Bytes ent-
110 * halten nichtdruckbare Zeichen.
111
7F32: 06 FD     112         ASL  OFFSET     ;Offset = 16
7F34: E6 06     113         INC  IND        ;Oberstes Byte
7F36: E6 07     114         INC  IND+1      ;und Control's
7F38: E6 07     115         INC  IND+1      ;überspringen
7F3A: A9 0A     116         LDA  #OPASL
7F3C: 8D 5D 7F  117         STA  CONVERT2
118
119 * Downloading initialisieren
120
7F3F: A9 9B     121 INIDWNLD LDA  #ESC
7F41: 20 E0 7F  122         JSR  PRINT
7F44: A9 AD     123         LDA  #"-"
7F46: 20 E0 7F  124         JSR  PRINT
```

TurtleGraphics-Library-Paket

von Dieter Geiß

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



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

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

Im einzelnen bietet das Paket folgende Möglichkeiten:

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

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg

```
7F49: A9 9B      125           LDA  #ESC
7F4B: 20 E0 7F   126           JSR  PRINT
7F4E: A9 C9      127           LDA  #"I"
7F50: 20 E0 7F   128           JSR  PRINT
                 129
                 130  * Zeichenschleife für Downloading
                 131  * initialisieren
                 132
7F53: A9 20      133           LDA  #SPACE
7F55: 85 CE      134           STA  CHAR
                 135
                 136  * Zeichen-Bit-Muster vom DOS Toolkit bzw.
                 137  * BITEDITOR in das Imagewriter-Format um-
                 138  * formen
                 139
7F57: A0 00      140  CONVERT  LDY  #$00
7F59: B1 06      141  CONVERT1 LDA  (IND),Y    ;Byte laden
7F5B: A2 00      142           LDX  #$00
7F5D: 4A         143  CONVERT2 LSR             ;gepokt
7F5E: 7E F4 7F   144           ROR  DATA,X     ;Zeichen-Bits
7F61: E8         145           INX             ;in Datenpuf-
7F62: E0 08      146           CPX  #08        ;fer schieben
7F64: 90 F7      147           BCC  CONVERT2
7F66: C8         148           INY
7F67: C0 08      149           CPY  #$08
7F69: 90 EE      150           BCC  CONVERT1
                 151
7F6B: A5 FB      152  TOOLKIT? LDA  TOOLKIT
7F6D: F0 07      153           BEQ  INVERT?
                 154
                 155  * Farb-Bit (= 8. Druckspalte) beim DOS-
                 156  * Toolkit-Zeichensatz ausmaskieren, da
                 157  * auf dem Imagewriter die Zeichenzeilen
                 158  * nicht wie in der HGR-Grafik um einen
                 159  * halben Punkt verschoben werden können.
                 160
7F6F: A9 00      161           LDA  #$00
7F71: A2 07      162           LDX  #$07
7F73: 9D F4 7F   163           STA  DATA,X
                 164
7F76: A5 FA      165  INVERT?  LDA  INVERS
7F78: F0 0D      166           BEQ  DOWNLOAD
                 167
                 168  * Zeichen invertieren
                 169
7F7A: A2 07      170           LDX  #$07
7F7C: BD F4 7F   171  INVLOOP  LDA  DATA,X     ;Alle Daten-Bytes
7F7F: 49 FF      172           EOR  #%11111111 ;invertieren und
7F81: 9D F4 7F   173           STA  DATA,X     ;wieder speichern
7F84: CA         174           DEX             ;fertig?
7F85: 10 F5      175           BPL  INVLOOP
                 176
                 177  * Bit-Muster an Imagewriter übertragen
                 178
                 179  * Zuerst die Einleitung mit Zeichen- und
                 180  * Zeichenbreitencode...
                 181
7F87: A5 CE      182  DOWNLOAD LDA  CHAR        ;Zeichen holen
7F89: 20 E0 7F   183           JSR  PRINT
7F8C: A9 48      184           LDA  #WIDTH+$40 ;Zeichenbreite
7F8E: 20 E0 7F   185           JSR  PRINT
                 186
                 187  * ...dann die eigentlichen Daten-Bytes zum
                 188  * Drucker übertragen
                 189
7F91: A2 00      190           LDX  #$00
7F93: BD F4 7F   191  DOWNLD0  LDA  DATA,X     ;Bytes aus vor-
7F96: 20 E0 7F   192           JSR  PRINT      ;bereiteter Ta-
7F99: E8         193           INX             ;belle holen
7F9A: E0 08      194           CPX  #$08       ;schon fertig?
7F9C: 90 F5      195           BCC  DOWNLD0    ;weiter
                 196
                 197  * Zeiger auf nächstes Zeichen setzen
                 198
7F9E: A5 06      199           LDA  IND        ;Zeichenbasis
7FA0: 18         200           CLC             ;laden und
7FA1: 65 FD      201           ADC  OFFSET     ;Offset addieren,
7FA3: 85 06      202           STA  IND        ;um nächstes
7FA5: 90 02      203           BCC  NEXTCHAR   ;Zeichen zu
7FA7: E6 07      204           INC  IND+1      ;bekommen
                 205
                 206  * nächstes Zeichen
                 207
7FA9: E6 CE      208  NEXTCHAR INC  CHAR        ;Buchstaben
7FAB: A5 CE      209           LDA  CHAR       ;erhöhen
7FAD: C9 7F      210           CMP  #DEL       ;schon fertig?
7FAF: 90 A6      211           BCC  CONVERT    ;nein
                 212
                 213  * Mit Ctrl-D das Downloading beenden
                 214
7FB1: A9 84      215           LDA  #CTRLD
```

```
7FB3: 20 E0 7F   216           JSR  PRINT
                 217
                 218  * Läuft das Programm auf einem Apple
                 219  * mit SSC oder auf einem IIc?
                 220
7FB6: AD B3 FB   221           LDA  ID1        ;ID-Byte #1
7FB9: C9 06      222           CMP  #$06
7FBB: D0 06      223           BNE  LFPATCH    ;-> kein IIc
7FBD: AD C0 FB   224           LDA  ID2        ;ID-Byte #2
7FC0: D0 01      225           BNE  LFPATCH    ;-> kein IIc
7FC2: 60         226           RTS             ;Alles fertig!
                 227
                 228  * Bei der SSC wird aus mir unbekannten
                 229  * Gründen das "LF nach CR"-Flag auf Null
                 230  * gesetzt. Die Folge davon ist, daß alle
                 231  * Zeichen, die auf dem Drucker ausgegeben
                 232  * werden, in eine Zeile gedruckt werden.
                 233  * LFPATCH setzt dieses Bit wieder auf 1.
                 234
7FC3: AD F9 07   235  LFPATCH  LDA  MISCFLG    ;LF nach CR
7FC6: 09 01      236           ORA  #$01       ;auf Eins
7FC8: 8D F9 07   237           STA  MISCFLG    ;setzen
7FCB: 60         238           RTS             ;nun alles OK!
                 239
                 240  * Anm: Die folgenden Ausgaberoutinen sind
                 241  *      geringfügig modifiziert aus dem Pro-
                 242  *      gramm "Superdump" von Dieter Geiß
                 243  *      entnommen. Das Programm erschien im
                 244  *      Peeker 6/85, Seite 22ff.
                 245
                 246  *********************************
                 247  *      SSC initialisieren       *
                 248  *********************************
                 249
7FCC: AD FF CF   250  INITSSC  LDA  $CFFF      ;ROMs abschalten
7FCF: AD 0D C1   251           LDA  $C10D      ;Offset-Byte
7FD2: A2 C1      252           LDX  #$C1       ;Register X und Y
7FD4: A0 10      253           LDY  #$10       ;für Slot 1 setzen
7FD6: 85 06      254           STA  IND        ;Sprung zur Karte
7FD8: 86 07      255           STX  IND+1      ;vorbereiten
7FDA: 8C F8 06   256           STY  N0         ;N0 setzen
7FDD: 6C 06 00   257           JMP  (IND)      ;ins ROM
                 258
                 259  *********************************
                 260  *      Ausgabe auf SSC          *
                 261  *********************************
                 262
7FE0: 48         263  PRINT    PHA             ;Zeichen retten
7FE1: AD 99 C0   264  WAIT     LDA  $C099      ;Status-Reg. Slot 1
7FE4: 29 10      265           AND  #$10       ;fertig?
7FE6: F0 F9      266           BEQ  WAIT       ;noch nicht
7FE8: AD 99 C0   267           LDA  $C099      ;Status-Register
7FEB: 25 FC      268           AND  READY      ;fertig?
7FED: D0 F2      269           BNE  WAIT       ;wieder nicht
7FEF: 68         270           PLA             ;Zeichen holen
7FF0: 8D 98 C0   271           STA  $C098      ;und ausgeben
7FF3: 60         272           RTS
                 273
                 274  *********************************
                 275  *     Interner Datenpuffer      *
                 276  *********************************
                 277
7FF4: 00 00 00   278  DATA     HEX  00000000   ;Zwischenspeicher
7FF7: 00
7FF8: 00 00 00   279           HEX  00000000   ;für Zeichen-Bytes
7FFB: 00

252 Bytes
```

FONT.CONVERT

BSAVE FONT.CONVERT, A$0300, L$00B8

```
                 1   *********************************
                 2   *                               *
                 3   *         FONT.CONVERT          *
                 4   *                               *
                 5   *   Carl Frieder Mahr, 1986     *
                 6   *                               *
                 7   *********************************
                 8
                 9            ORG  $0300
                10
                11   * konvertiert DOS-Toolkit-Zeichensätze in
                12   * das BITEDITOR-Format und umgekehrt.
                13   *
                14   * Aufruf aus FONT.LOADER mit CALL 768
                15
                16   * Label-Definitionen
                17
```

```
                        18  IND       EQU  $06
                        19  IND2      EQU  $08
                        20  TEMP      EQU  $CE
                        21  CHAR      EQU  $CF
                        22  SPACE     EQU  ' '
                        23  DEL       EQU  $7F
                        24
                        25  * Toolkit: 0: BITEDITOR-Zeichensatz
                        26  *          1: DOS-Toolkit-Zeichensatz
                        27
                        28  TOOLKIT   EQU  $FB        ;Flag
                        29
                        30  * Zeichensatzbeginn und -längen
                        31
                        32  FONT      EQU  $8000
                        33  DTLEN     EQU  $0300
                        34  BELEN     EQU  $0800
                        35
                        36  * Zeichsatzformat feststellen
                        37
0300: A5 FB             38  CONVERT   LDA  TOOLKIT
0302: F0 66             39            BEQ  BE:TO:DT
                        40
                        41  ****************************
                        42  * DOS Toolkit -> BITEDITOR *
                        43  ****************************
                        44
                        45  * BITEDITOR-Flag setzen
                        46
0304: C6 FB             47  DT:TO:BE DEC  TOOLKIT
                        48
                        49  * Zeichenbasis initialisieren
                        50
0306: A9 F7             51            LDA  #<FONT+DTLEN-9
0308: 85 06             52            STA  IND
030A: A9 82             53            LDA  #>FONT+DTLEN-1
030C: 85 07             54            STA  IND+1
030E: A9 F0             55            LDA  #<FONT+BELEN-16
0310: 85 08             56            STA  IND2
0312: A9 87             57            LDA  #>FONT+BELEN-1
0314: 85 09             58            STA  IND2+1
                        59
                        60  * Zeichenschleife initialisieren
                        61
0316: A9 7F             62            LDA  #DEL
0318: 85 CF             63            STA  CHAR
                        64
                        65  * nicht benutzte Daten-Bytes auf 0 setzen
                        66
031A: A9 00             67  CONVDT    LDA  #$00
031C: A0 00             68            LDY  #$00
031E: 91 08             69            STA  (IND2),Y
0320: A0 0F             70            LDY  #$0F
0322: 91 08             71  CONVDT0   STA  (IND2),Y
0324: 88                72            DEY
0325: C0 09             73            CPY  #$09
0327: B0 F9             74            BCS  CONVDT0
                        75
                        76  * zeichenweise Konvertierung
                        77
0329: A0 08             78            LDY  #$08
032B: B1 06             79  CONVDT1   LDA  (IND),Y
032D: 20 AD 03          80            JSR  REVERT
0330: 91 08             81            STA  (IND2),Y
0332: 88                82            DEY
0333: D0 F6             83            BNE  CONVDT1
                        84
                        85  * Zeiger auf nächstes Zeichen
                        86
0335: A5 06             87            LDA  IND
0337: 38                88            SEC
0338: E9 08             89            SBC  #$08
033A: 85 06             90            STA  IND
033C: B0 02             91            BCS  CONVDT2
033E: C6 07             92            DEC  IND+1
0340: A5 08             93  CONVDT2   LDA  IND2
0342: 38                94            SEC
0343: E9 10             95            SBC  #$10
0345: 85 08             96            STA  IND2
0347: B0 02             97            BCS  CONVDT3
0349: C6 09             98            DEC  IND2+1
034B: C6 CF             99  CONVDT3   DEC  CHAR
034D: A5 CF            100            LDA  CHAR
034F: C9 20            101            CMP  #SPACE     ;fertig?
0351: B0 C7            102            BCS  CONVDT     ;weiter
                       103
                       104  * Bereich der Ctrl-Zeichen auf
                       105  * Null setzen
                       106
0353: A9 00            107            LDA  #<FONT
0355: 85 06            108            STA  IND
```

```
0357: A9 80            109            LDA  #>FONT
0359: 85 07            110            STA  IND+1
035B: 20 60 03         111            JSR  ERAPAGE    ;Trick
035E: E6 07            112            INC  IND+1
                       113
                       114  * Speicher-Page löschen
                       115
0360: A0 00            116  ERAPAGE   LDY  #$00
0362: A9 00            117            LDA  #$00
0364: 91 06            118  ERAPAGE0  STA  (IND),Y
0366: C8               119            INY
0367: D0 FB            120            BNE  ERAPAGE0
0369: 60               121            RTS
                       122
                       123  ****************************
                       124  * BITEDITOR -> DOS Toolkit *
                       125  ****************************
                       126
                       127  * DOS-Toolkit-Flag setzen
                       128
036A: E6 FB            129  BE:TO:DT INC  TOOLKIT
                       130
                       131  * Zeichenbasis initialisieren
                       132
036C: A9 01            133            LDA  #<FONT+1
036E: 85 06            134            STA  IND
0370: A9 82            135            LDA  #>FONT+$0200
0372: 85 07            136            STA  IND+1
0374: A9 00            137            LDA  #<FONT
0376: 85 08            138            STA  IND2
0378: A9 80            139            LDA  #>FONT
037A: 85 09            140            STA  IND2+1
                       141
                       142  * Zeichenschleife initialisieren
                       143
037C: A9 20            144            LDA  #SPACE
037E: 85 CF            145            STA  CHAR
                       146
                       147  * zeichenweise Konvertierung
                       148
0380: A0 07            149  CONVBE    LDY  #$07
0382: B1 06            150  CONVBE0   LDA  (IND),Y
0384: 20 AD 03         151            JSR  REVERT
                       152
                       153  * Farb-Bit für DOS-Toolkit-Zeichensatz
                       154  * auf 0 setzen
                       155
0387: 29 7F            156            AND  #$7F
0389: 91 08            157            STA  (IND2),Y
038B: 88               158            DEY
038C: 10 F4            159            BPL  CONVBE0
                       160
                       161  * Zeiger auf nächstes Zeichen
                       162
038E: A5 06            163            LDA  IND
0390: 18               164            CLC
0391: 69 10            165            ADC  #$10
0393: 85 06            166            STA  IND
0395: 90 02            167            BCC  CONVBE1
0397: E6 07            168            INC  IND+1
0399: A5 08            169  CONVBE1   LDA  IND2
039B: 18               170            CLC
039C: 69 08            171            ADC  #$08
039E: 85 08            172            STA  IND2
03A0: 90 02            173            BCC  CONVBE2
03A2: E6 09            174            INC  IND2+1
03A4: E6 CF            175  CONVBE2   INC  CHAR
03A6: A5 CF            176            LDA  CHAR
03A8: C9 80            177            CMP  #DEL+1     ;fertig?
03AA: 90 D4            178            BCC  CONVBE     ;weiter
03AC: 60               179            RTS
                       180
                       181  * Zeichen-Byte umdrehen
                       182
03AD: A2 08            183  REVERT    LDX  #$08       ;8 Bits
03AF: 0A               184  REVERT0   ASL             ;links raus
03B0: 66 CE            185            ROR  TEMP       ;rechts rein
03B2: CA               186            DEX
03B3: D0 FA            187            BNE  REVERT0
03B5: A5 CE            188            LDA  TEMP
03B7: 60               189            RTS

184 Bytes
```

Haushaltsverwaltung

auf dem Apple IIc/IIe

von Thilo Schön

Mein Ziel war es, ein einfaches und übersichtliches Applesoft-Programm zu schreiben, das mit möglichst großem Komfort und vielen Möglichkeiten helfen sollte, die Verwaltung des Geldes im Haushalt zu übernehmen. Das im folgenden beschriebene Programm erfüllt diese Anforderungen und hat sich in der Praxis bewährt.

1. Die verwalteten Daten

Das wichtigste Problem war zunächst zu entscheiden, welche Daten erfaßt und wie diese verwaltet werden sollten. Ich entschloß mich, die Einnahmen und Ausgaben in verschiedene Sparten aufzuteilen, die einen besseren Überblick und eine leichtere Verwaltung ermöglichen. Die Sparten können vom Anwender frei gewählt werden. Mögliche Namen sind zum Beispiel AUTO, HOBBY, KLEIDUNG, LEBENSM., SPORT oder EINNAHMEN. Ein vollständiger Datensatz enthält das aktuelle Datum, die Sparte, eine Bemerkung und den genauen Geldbetrag in DM.

2. Die Ausgabe

Ein weiterer wichtiger Punkt ist die Ausgabe der gespeicherten Daten. Hier gibt es drei Möglichkeiten.
– Die erste gibt eine Übersicht aller Einträge. Man kann dabei auch einzelne Sparten auswählen, so daß zum Beispiel nur die Einträge der Sparte AUTO ausgegeben werden.
– Als zweites können die Einträge der einzelnen Sparten summiert werden, wobei auch eine Gesamtbilanz erstellt wird. Auch hier ist es möglich, nur einzelne Sparten oder nur bestimmte Zeiträume zu berechnen.
– Diese Berechnungen können schließlich als Balkendiagramm ausgegeben werden (s. Grafik). Dabei veranschaulicht ein grauer Teilstrich 25 Prozent der Gesamtausgaben.
Die Übersicht und die Berechnungen können auch auf einen Drucker ausgegeben werden. Das Programm wurde mit einem Imagewriter getestet. Da jedoch keinerlei Steuerzeichen verwendet werden, läuft es auch mit beliebigen anderen Druckern.

3. Die Grafik

Für die grafische Darstellung wird die doppelt niedrigauflösende Grafik verwendet, da die hochauflösende im Speicher mit dem Programm kollidieren würde. Beim IIe kann die Double Lores nicht in ähnlich einfacher Weise realisiert werden wie beim IIc. Daher müssen hier für die einfache Grafik die unter dem Listing angegebenen Zeilen ersetzt werden.

4. Die Programmgliederung

In dem Programm gibt es vier Hauptmenüs, die durch Großschreibung gekennzeichnet sind. Zwischen diesen Hauptmenüs (DISKETTE, NEUEINGABE, ÄNDERUNG und AUSGABE) können Sie beliebig hin- und herspringen. Die eigentlichen Programmfunktionen sind durch Kleinschreibung gekennzeichnet. In allen Programmteilen gibt es die Möglichkeit, bei Rückfragen des Programms durch die Eingabe von „<“ (anstatt eines Linkspfeils) zum vorhergehenden Hauptmenü zurückzuspringen.
Abschließend ist zu sagen, daß sich das Programm in den drei Monaten, in denen es getestet werden konnte, hervorragend bewährt hat. Wenn man alle Kassenbelege aufhebt und jeden Abend alle Ausgaben – und auch die Einnahmen – gewissenhaft einträgt, dann hat man jederzeit einen genauen Überblick über seine Finanzen und kann am Ende des Monats nachprüfen, wieviel Geld man für was ausgegeben hat. Es hat sich übrigens als günstig herausgestellt, für jeden Monat eine Datei zu führen. Dann hat man mit dem geringsten Zeitaufwand den besten Überblick über seine Ein- und Ausgaben.

Kurzhinweise

1. Zweck:
Programm zur Haushaltsverwaltung
2. Konfiguration:
IIc oder IIe;
DOS 3.3 oder ProDOS
3. Test:
RUN HAUSHALT
4. Sammeldisk:
HAUSHALT
HAUSHALT.40
DATEI
(Applesoft-Programm)

HAUSHALT

```
1000 REM $$$$$$$$$$$$$$$$$$
1010 REM $$   Haushalt    $$
1020 REM $ von Thilo Schön $
1030 REM $$ Dezember 1985 $$
1040 REM $$$$$$$$$$$$$$$$$$
1050 D$ = CHR$ (4): PRINT D$"PR#3"
1060 DI = 1000
1070 DIM DT$(DI),DD$(DI),DG$(DI),DS(DI),DS$(15),BS(15),BE(15)
1080 REM _____Diskette_____
1090 TEXT : NORMAL : HOME : PRINT
1100 MT$(1) = "Einladen einer Datei":MT$(2) = "Speichern einer
     Datei":MT$(3) = "Neueröffnung einer Datei":MT$(4) =
     "NEUEINGABE":MT$(5) = "ÄNDERUNG":MT$(6) =
     "AUSGABE":MT$(7) = "ENDE"
1110 MA = 7:MU$ = "DISKETTE": GOSUB 2990: IF A < 4 THEN 1130
1120 ON A - 3 GOTO 1500,1780,2250,2960
1130 ON A GOSUB 1140,1250,1340: GOTO 1080
1140 REM -----Laden---------
1150 HOME : PRINT : INVERSE : PRINT "Einladen einer Datei":
     PRINT
1160 NORMAL : PRINT "Geben Sie den Namen der Datei oder
     <RETURN> für den Catalog ein : ": INPUT N$
1170 IF N$ = "" THEN PRINT D$"CATALOG": GOTO 1160
1180 IF N$ = "<" THEN RETURN
1190 PRINT D$"OPEN ";N$: PRINT D$"READ ";N$
1200 INPUT DD$,DA,DS: FOR LI = 1 TO DS: INPUT DS$(LI): NEXT LI
1210 PRINT : PRINT "Die Datei enthält ";DA;" Einträge und
     wurde am ";DD$;" eröffnet."
1220 IF DA = Ø THEN 1240
1230 FOR LI = 1 TO DA: INPUT DD$(LI),DT$(LI),DG$(LI),DS(LI):
     NEXT LI
1240 PRINT : PRINT D$"CLOSE ";N$: PRINT "EINLADEN BEENDET -
     Bitte drücken Sie eine Taste. ";: GET A$: RETURN
1250 REM -----Speichern-----
1260 HOME : PRINT : INVERSE : PRINT "Abspeichern einer Datei":
     NORMAL
1270 PRINT : PRINT "Geben Sie den Namen der Datei ein (
     <RETURN> = ";N$;" ) :": INPUT NN$: IF NN$ = "<" THEN
     RETURN
1280 IF NN$ < > "" THEN N$ = NN$
1290 PRINT D$"OPEN ";N$: PRINT D$"WRITE ";N$
1300 PRINT DD$: PRINT DA: PRINT DS: FOR SI = 1 TO DS: PRINT
     DS$(SI): NEXT SI
1310 IF DA = 0 THEN 1330
1320 FOR SI = 1 TO DA: PRINT DD$(SI): PRINT DT$(SI): PRINT
     DG$(SI): PRINT DS(SI): NEXT SI
1330 PRINT D$"CLOSE ";N$: RETURN
1340 REM -----Öffnen--------
1350 HOME : PRINT : INVERSE : PRINT "Neueröffnen einer Datei":
     NORMAL : POKE 34,2
1360 PRINT : PRINT "Geben Sie den Namen der Datei ein : ":
     INPUT N$
1370 HOME : IF N$ = "<" THEN RETURN
1380 PRINT : INPUT "Wollen Sie die Spartennamen von einer
     alten Datei übernehmen (J/N)? ";A$: IF A$ = "N" OR A$ =
     "n" THEN HOME : GOTO 1420
1390 INPUT "Geben Sie den Namen der alten Datei ein : ";NA$
1400 PRINT D$"OPEN "NA$: PRINT D$"READ "NA$
1410 INPUT XX$,XX,XX: FOR OI = 1 TO XX: INPUT DS$(OI): VTAB OI
     + 5: PRINT OI;: HTAB 5: PRINT "--> ";DS$(OI): NEXT OI:
     PRINT D$"CLOSE "NA$: GOTO 1470
1420 PRINT "Sie können nun die Namen von bis zu 15 Sparten
     (maximal 10 Buchstaben) eingeben.Danach geben Sie
     <RETURN> ein."
1430 PRINT :OI = 1
1440 VTAB OI + 5: PRINT OI;: HTAB 5: INPUT "--> ";DS$(OI)
1450 IF LEN (DS$(OI)) > 10 THEN 1440
1460 IF OI < 15 AND DS$(OI) < > "" THEN OI = OI + 1: GOTO 1440
1470 PRINT : INPUT "War alles richtig (J/N)? ";A$: IF A$ = "N"
     OR A$ = "n" THEN 1430
1480 PRINT : INPUT "Geben Sie bitte das heutige Datum ein :
     ";DD$:DA = 0:DS = OI - 1
1490 POKE 34,0: GOSUB 1290: RETURN
1500 REM _____Neueingabe_____
1510 TEXT : HOME :MT$(1) = "Neueingabe von Datensätzen":MT$(2)
     = "DISKETTE":MT$(3) = "ÄNDERUNG":MT$(4) =
     "AUSGABE":MT$(5) = "ENDE"
1520 MU$ = "NEUEINGABE":MA = 5: GOSUB 2990
1530 IF A = 1 THEN GOSUB 1550: GOTO 1500
1540 ON A - 1 GOTO 1080,1780,2250,2960
1550 REM -----Neueingabe----
1560 DA = DA + 1
1570 HOME : PRINT : INVERSE : VTAB 1: PRINT "Neueingabe von
     Datensätzen": NORMAL : PRINT : POKE 34,2
1580 PRINT "DATUM (";DD$(DA - 1);"):": PRINT SPC(
     17);"'''''''''": PRINT : PRINT "SPARTE(1-";DS;"):": PRINT
     SPC( 17);"''''''''''": PRINT : PRINT "BEMERKUNG:"
1590 PRINT SPC( 17);"''''''''''''''''''''''''''''''''''''''''":
     PRINT : PRINT "GELDBETRAG(+/-):": PRINT SPC(
     17);"''''''''''"
1600 VTAB 21: FOR EI = 1 TO DS: PRINT EI;"=";DS$(EI),: NEXT EI
1610 VTAB 19: PRINT : FOR EI = 1 TO 79: PRINT "_";: NEXT EI
1620 VTAB 3: HTAB 18: INPUT "";DD$(DA): IF LEN (DD$(DA)) > 8
     THEN 1620
1630 IF DD$(DA) = "<" THEN DA = DA - 1: GOTO 1770
1640 IF DD$(DA) = "" THEN DD$(DA) = DD$(DA - 1): VTAB 3: HTAB
     18: PRINT DD$(DA - 1)
1650 VTAB 6: HTAB 18: INPUT "";DS(DA): IF DS(DA) > DS OR
     DS(DA) < 1 THEN 1650
1660 VTAB 6: HTAB 18: PRINT DS$(DS(DA))
1670 VTAB 9: HTAB 18: INPUT "";DT$(DA): IF LEN (DT$(DA)) > 38
     THEN 1670
1680 VTAB 12: HTAB 18: INPUT "";DG: VTAB 17: PRINT SPC( 70)
1690 DG$(DA) = STR$ ( ABS (DG)):PU = 0:ZL = LEN (DG$(DA))
1700 FOR I = 1 TO 10: IF MID$ (DG$(DA),I,1) = "." THEN PU =
     I:I = 10
1710 NEXT I: IF PU = 0 THEN DG$(DA) = DG$(DA) + ".":PU = LEN
     (DG$(DA))
1720 DG$(DA) = DG$(DA) + MID$ ("00",1,2 - (ZL - PU)): IF DG <
     0 THEN DG$(DA) = "-" + DG$(DA)
1730 IF DG > = 0 THEN DG$(DA) = "+" + DG$(DA)
1740 PRINT : VTAB 16: INPUT "War die Eingabe korrekt (J/N)?
     ";A$: IF A$ = "N" OR A$ = "n" THEN 1600
1750 IF FL = 1 THEN RETURN
1760 INPUT "Wollen Sie noch einen Datensatz eingeben (J/N)?
     ";A$: IF A$ = "J" OR A$ = "j" THEN 1550
1770 POKE 34,0: RETURN
1780 REM _____Änderung_____
1790 TEXT : HOME :MT$(1) = "Ändern eines Datensatzes":MT$(2) =
     "Löschen eines Datensatzes":MT$(3) = "DISKETTE":MT$(4) =
     "NEUEINGABE":MT$(5) = "AUSGABE":MT$(6) = "ENDE"
1800 MU$ = "ÄNDERUNG":MA = 6: GOSUB 2990
1810 IF A < 3 THEN ON A GOSUB 1900,1830: GOTO 1780
1820 ON A - 2 GOTO 1080,1500,2250,2960
1830 REM -----Löschen-------
1840 HOME : PRINT : INVERSE : VTAB 1: PRINT "Löschen eines
     Datensatzes": NORMAL : PRINT : POKE 34,2
1850 GOSUB 1960: REM Suchen
1860 VTAB 17: PRINT "Löschen des Datensatzes mit <SPACE> ";:
     GET A$: PRINT : IF A$ = "<" THEN 1850
1870 IF A$ < > " " THEN 1860
1880 FOR LI = SF + 1 TO DA:DD$(LI - 1) = DD$(LI):DT$(LI - 1) =
     DT$(LI):DG$(LI - 1) = DG$(LI):DS(LI - 1) = DS(LI): NEXT
     LI:DA = DA - 1
1890 POKE 34,0: RETURN
1900 REM -----Ändern--------
1910 HOME : PRINT : INVERSE : VTAB 1: PRINT "Ändern eines
     Datensatzes": NORMAL : PRINT : POKE 34,2
1920 GOSUB 1960: REM Suchen
1930 VTAB 17: PRINT "Ändern des Datensatzes mit <SPACE> ";:
     GET A$: PRINT : IF A$ = "<" THEN 1920
1940 IF A$ < > " " THEN 1930
1950 ADA = DA:DA = SF:FL = 1: GOSUB 1600:FL = 0:DA = ADA:
     RETURN
1960 REM =====Suchen=======
1970 HOME : VTAB 3: PRINT "Nach welchem Kriterium soll der
     Datensatz gesucht werden ?": PRINT : PRINT "1 - DATUM":
     PRINT "2 - SPARTE": PRINT "3 - BEMERKUNG": PRINT "4 -
     GELDBETRAG"
1980 VTAB 10: INPUT "Geben Sie das gewünschte Kriterium (1-4)
     ein : ";A$: PRINT : IF A$ = "<" THEN POP : GOTO 1780
1990 SK = VAL (A$): IF SK < 1 OR SK > 4 THEN 1980
2000 HOME :SA = 1: ON SK GOTO 2010,2050,2110,2160
2010 REM +++++Datum+++++++++
2020 INPUT "Geben Sie das Datum ein : ";AA$: IF AA$ = "<" THEN
     1970
2030 SF = 0: FOR SI = SA TO DA: IF DD$(SI) = AA$ THEN SF =
     SI:SI = DA
2040 NEXT SI: GOTO 2200
2050 REM +++++Sparte++++++++
2060 FOR SI = 1 TO DS: PRINT SI;" - ";DS$(SI): NEXT SI: PRINT
2070 VTAB 23: INPUT "Geben Sie die Nummer der Sparte ein :
     ";AA$: IF AA$ = "<" THEN 1970
2080 AA = VAL (AA$): IF AA < 1 OR AA > (DS) THEN 2070
2090 SF = 0: FOR SI = SA TO DA: IF DS(SI) = (AA) THEN SF =
     SI:SI = DA
2100 NEXT SI: GOTO 2200
2110 REM +++++Bemerkung+++++
2120 VTAB 5: PRINT "Geben Sie die Bemerkung ein :": INPUT
     "";AA$: IF AA$ = "<" THEN 1970
2130 AA = LEN (AA$): IF AA > 42 THEN 2120
2140 SF = 0: FOR SI = SA TO DA: IF LEFT$ (DT$(SI),AA) = LEFT$
     (AA$,AA) THEN SF = SI:SI = DA
2150 NEXT SI: GOTO 2200
2160 REM +++++Geldbetrag++++
2170 INPUT "Geben Sie den Geldbetrag ein : ";AA$: IF AA$ = "<"
     THEN 1970
```

```
2180 AA = VAL (AA$):SF = 0: FOR SI = SA TO DA: IF VAL
     (DG$(SI)) = (AA) THEN SF = SI:SI = DA
2190 NEXT SI: GOTO 2200
2200 IF SF = 0 THEN PRINT "N i c h t s  g e f u n d e n !":
     FOR W = 1 TO 1000: NEXT W: GOTO 1970
2210 HOME : VTAB 3: PRINT "DATUM:" TAB( 18)DD$(SF): VTAB 6:
     PRINT "SPARTE:" TAB( 18)DS$(DS(SF)): VTAB 9: PRINT
     "BEMERKUNG:" TAB( 18)DT$(SF): VTAB 12: PRINT "GELDBETRAG:
     " TAB( 18) VAL (DG$(SF))
2220 VTAB 16: INPUT "Wollen Sie weitersuchen (J/N)? ";A$: IF
     A$ < > "N" AND A$ < > "n" AND A$ < > "j" AND A$ < > "J"
     THEN 2220
2230 IF A$ = "J" OR A$ = "j" THEN SA = SF + 1: ON SK GOTO
     2030,2090,2140,2180
2240 RETURN
2250 REM _____Ausgabe_______
2260 TEXT : HOME :MT$(1) = "Übersicht ausgeben":MT$(2) =
     "Summen berechnen":MT$(3) = "DISKETTE":MT$(4) =
     "NEUEINGABE":MT$(5) = "ÄNDERUNG":MT$(6) = "ENDE"
2270 MA = 6:MU$ = "AUSGABE + BERECHNUNG": GOSUB 2990
2280 IF A < 3 THEN ON A GOSUB 2300,2510: GOTO 2250
2290 ON A - 2 GOTO 1080,1500,1780,2960
2300 REM -----Übersicht-----
2310 HOME : PRINT : INVERSE : VTAB 1: PRINT "Übersicht
     ausgeben": NORMAL : PRINT
2320 INPUT "Wieviele Sparten wollen Sie ausgeben (A=Alle)?
     ";A$: IF A$ = "<" THEN RETURN
2330 IF A$ = "A" OR A$ = "a" THEN BA = DS: FOR I = 1 TO
     DS:BS(I) = I: NEXT : GOTO 2390
2340 BA = VAL (A$): IF BA < 1 OR BA > DS THEN 2310
2350 VTAB 21: FOR EI = 1 TO DS: PRINT EI;"=";DS$(EI),: NEXT EI
2360 VTAB 19: PRINT : FOR EI = 1 TO 79: PRINT "_";: NEXT EI
2370 VTAB 4: PRINT : PRINT "Geben Sie nun die auszugebenden
     Sparten ein :"
2380 FOR I = 1 TO BA: PRINT I;". Sparte: ";: INPUT "";BS(I):
     NEXT I
2390 INPUT "Soll die Übersicht auf den Drucker ausgegeben
     werden (J/N)? ";A$:DR = 0: IF A$ = "<" THEN RETURN
2400 IF A$ = "J" OR A$ = "j" THEN DR = 1: PRINT D$"PR#1":
     PRINT " ": GOSUB 2500
2410 FOR I = 1 TO DA: IF (I - 1) / 20 = INT ((I - 1) / 20) AND
     DR = 0 THEN GOSUB 2490
2420 IF BA = DS THEN 2450
2430 FL = 0: FOR II = 1 TO BA: IF BS(II) = DS(I) THEN FL = 1
2440 NEXT II: IF FL = 0 THEN 2470
2450 PRINT DD$(I); SPC( 9 - LEN (DD$(I)));DS$(DS(I)); SPC( 11
     - LEN (DS$(DS(I))));DT$(I); SPC( 38 - LEN (DT$(I)));: IF
     VAL (DG$(I)) < = 0 THEN PRINT SPC( 11);
2460 PRINT RIGHT$ ("          " + RIGHT$ (DG$(I), LEN (DG$(I))
     - 1),10)
2470 NEXT I
2480 IF DR = 1 THEN PRINT D$"PR#3": RETURN
2490 PRINT : PRINT "Bitte drücken Sie eine Taste ";: GET A$:
     HOME
2500 PRINT "DATUM     SPARTE      BEMERKUNG
     GUTSCHRIFT
     LASTSCHRIFT═══════╪═══════╪═══════════════════
     ═══════╪═══════╪═══════";: RETURN
2510 REM -----Berechnung----
2520 FOR I = 1 TO 15:BE(I) = 0:BS(I) = 0: NEXT I
2530 TEXT : HOME : PRINT : INVERSE : VTAB 1: PRINT "Berechnung
     einzelner Summen": NORMAL : PRINT : POKE 34,2
2540 FOR BI = 1 TO DS: PRINT " ";BI;" ";DS$(BI): NEXT BI:
     PRINT : POKE 34,DS + 3
2550 VTAB DS + 4: INPUT "Mit wievielen Sparten wollen Sie
     rechnen (A=Alle)? ";A$: IF A$ = "<" THEN RETURN
2560 IF A$ = "A" OR A$ = "a" THEN BA = DS: FOR I = 1 TO
     DS:BS(I) = I: NEXT : GOTO 2600
2570 BA = VAL (A$): IF BA < 1 OR BA > DS THEN 2550
2580 HOME : PRINT "Geben Sie nun die zu berechnenden Sparten
     ein:"
2590 FOR I = 1 TO BA: PRINT I;". Sparte: ";: INPUT "";BS(I):
     NEXT I
2600 HOME : INPUT "Wollen Sie alle Daten berechnen (J/N)?
     ";A$: IF A$ = "<" THEN 2550
2610 IF A$ = "j" OR A$ = "J" THEN BO$ = DD$(1):B1$ =
     DD$(DA):BO = 1:B1 = DA: GOTO 2670
2620 PRINT : INPUT "Geben Sie das 1.Datum der Berechnung ein:
     ";BO$: INPUT "Geben Sie das letzte Datum der Berechnung
     ein: ";B1$
2630 BO = 0: FOR I = 1 TO DA: IF DD$(I) = BO$ THEN BO = I:I =
     DA
2640 NEXT I:B1 = 0: FOR I = BO TO DA: IF DD$(I) = B1$ THEN B1
     = I
2650 NEXT I
2660 IF BO = 0 OR B1 = 0 THEN PRINT " D a t e n  n i c h t
     e f u n d e n !": FOR W = 1 TO 1000: NEXT W: GOTO 2620
2670 HOME : PRINT "Die Berechnung erfolgt ";
2680 FOR S = 1 TO BA: PRINT ".";: FOR I = BO TO B1
2690 IF BS(S) = DS(I) THEN BE(S) = BE(S) + VAL (DG$(I))
2700 NEXT I,S: PRINT CHR$ (7): PRINT : INPUT "Soll die
     Berechnung auf den Drucker ausgegeben werden (J/N)? ";A$:
     IF A$ = "J" OR A$ = "j" THEN PRINT D$"PR#1"
2710 TEXT : HOME : PRINT : INVERSE : PRINT "Berechnung der
     Summen vom ";BO$;" bis zum ";B1$
2720 NORMAL : PRINT : FOR I = 1 TO BA: PRINT "Sparte "; LEFT$
     (DS$(BS(I)) + " ..........",14);" ";: IF BE(I) > 0 THEN
     PRINT "Einnahmen : ";: GOTO 2740
2730 PRINT "Ausgaben  : ";
2740 BE$ = STR$ ( ABS ( INT (BE(I) * 100 + .5))):BE$ = LEFT$
     (BE$, ABS ( LEN (BE$) - 2)) + "." + RIGHT$ (BE$,2): PRINT
     RIGHT$ ("          " + BE$,10);" DM": NEXT I: PRINT
2750 BE = 0:BP = 0: FOR I = 1 TO BA: IF BE(I) < 0 THEN BE = BE
     + BE(I)
2760 NEXT I: FOR I = 1 TO BA: IF BE(I) > 0 THEN BP = BP +
     BE(I)
2770 NEXT I: PRINT "Ausgaben  : "; ABS (BE);" DM"
2780 PRINT "Einnahmen : "; ABS (BP);" DM"
2790 PRINT "Bilanz    : "; INT ((BP + BE) * 100 + .5) / 100;"
     DM"
2800 IF A$ = "j" OR A$ = "J" THEN PRINT D$"PR#3"
2810 IF BA < > DS THEN PRINT : INPUT "Drücken Sie <RETURN>
     ";A$: GOTO 2950
2820 PRINT : INPUT "Wollen Sie die Grafik sehen (J/N)? ";A$:
     IF A$ = "N" OR A$ = "n" THEN 2950
2830 REM ═══Grafik═══
2840 QQ = PEEK ( - 16290): GR : HOME :SB = INT (80 / BA):VT =
     21:TL = 0
2850 COLOR= 10: FOR I = 0 TO 3: HLIN 0,79 AT I * 10: NEXT I
2860 COLOR= 15: FOR I = 0 TO 8 STEP 2: VLIN 0,4 AT I: NEXT I:
     FOR I = 2 TO 6 STEP 4: FOR J = 0 TO 4 STEP 4: HLIN I,I +
     2 AT J: NEXT J,I
2870 FOR I = 0 TO 4 STEP 4: FOR J = 20 TO 24 STEP 4: HLIN I,I
     + 2 AT J: NEXT J,I: HLIN 0,2 AT 22: VLIN 20,22 AT 0: VLIN
     22,24 AT 2: VLIN 20,24 AT 4: VLIN 20,24 AT 6
2880 FOR I = 1 TO BA: IF BE(I) < 0 THEN TL = TL + ABS (BE(I))
2890 NEXT I
2900 FOR I = 1 TO BA:SH = 0: IF BE(I) < > 0 THEN SH = INT (40
     / (TL / ABS (BE(I))) + .5): IF SH > 39 THEN SH = 39
2910 COLOR= VT - 9: IF BE(I) > 0 THEN COLOR= 15
2920 FOR J = (I - 1) * SB TO (I - 1) * SB + SB - 1: IF SH < >
     0 THEN VLIN 39 - SH,39 AT J: NEXT J
2930 HTAB (I - 1) * SB + 1: VTAB VT: PRINT LEFT$ (DS$(I),SB *
     2 - 1):VT = 43 - VT: NEXT I
2940 VTAB 24: INPUT "Drücken Sie <RETURN> ";A$
2950 RETURN
2960 REM _____Ende_____
2970 PRINT : INVERSE : PRINT "Wollen Sie das Programm wirklich
     beenden ?": PRINT "Haben Sie alles abgespeichert ?";:
     NORMAL : INPUT "";A$: IF A$ < > "J" AND A$ < > "j" THEN
     HOME : GOTO 1080
2980 TEXT : NORMAL : HOME : END
2990 REM ═══Menü═══
3000 PRINT : INVERSE :ML = (80 - LEN (MU$)) / 2: VTAB 1: PRINT
     SPC( ML);MU$; SPC( ML)
3010 FOR MI = 1 TO MA: INVERSE : VTAB MI * 2 + 2: PRINT MI;:
     NORMAL : PRINT " "MT$(MI): NEXT MI
3020 VTAB 20: PRINT "Geben Sie 1 bis ";MA;" ein : ";: INPUT
     "";A$
3030 A = VAL (A$): IF A < 1 OR A > (MA) THEN 3020
3040 RETURN
```

Der Applesoft-Interpreter des IIe (nicht Enhanced IIe) unterstützt die doppelt niedrigauflösende Grafik nicht. Daher müssen für die einfache Grafik folgende Zeilen in dem Listing geändert werden.

```
2840 GR : HOME : SB = INT (40 / BA):VT = 21:TL = 0
2850 COLOR= 10: FOR I = 0 TO 3: HLIN 0,39 AT I * 10:NEXT I
2930 POKE 1403,(I - 1) * SB * 2 + 1: VTAB VT: PRINT LEFT$
     (DS$(I),SB * 4 - 1):VT = 43 - VT: NEXT I
```

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 Anfrage.

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

PEEKER

Peeker-Sammeldisk #12 und #13

(Einzelpreis DM 28,–; Fortsetzungspreis DM 20,–)

Disk #12

(DOS 3.3; Heft 12/85)

COSMO CRUMBLE
T.CC2 – T.CC6
T.CC8
CC2 – CC8
CC.LEVELS
(1) Reaktionsspiel für Tastatur oder Joystick; (2) Heft 12/85, S. 6; (3) II+, IIe oder IIc; wahlweise Joystick; (4) DOS 3.3; (5) RUN COSMO CRUMBLE

IW.DEMO
T.IW.IN
IW.IN
T.IW.OUT
IW.OUT
ECHO
(1) Initialisierung des Imagewriters durch mit IW.DEMO generierbare, BRUN-fähige Steuer-Sequenz-Programme; (2) Heft 12/85, S. 12; (3) IIe oder IIc (II+ mit Einschränkungen, wegen der Tastatur); Imagewriter (beim IIe mit Super-Serial-Card); (4) DOS 3.3 oder ProDOS; (5) RUN IW.DEMO

ASCII.EDITOR
COPY.TEXT.DEMO
T.COPY.TEXT
COPY.TEXT
ASCII.CODES
(1) Erstellung eines eigenen Zeichensatzes; Kopieren des 40-Z/Z-Text-Bildschirms in den HGR-Bereich; (2) Heft 12/85, S. 16; (3) speziell II+, aber auch IIe oder IIc mit Einschränkungen; (4) DOS 3.3; (5) RUN ASCII.EDITOR; RUN COPY.TEXT-.DEMO; (6) Nach FLASH werden Großbuchstaben auf dem HGR-Bildschirm als Kleinbuchstaben ausgegeben

GETTEXT.DEMO
T.GETTEXT
GETTEXT
GETTEXT.PRODOS
(1) Hilfsprogramm zur Erweiterung des Tastatur-INPUT-Befehls durch Editierfunktionen; (2) Heft 12/85, S. 20; (3) II+, IIe oder IIc (40 und 80 Z/Z, auch Videx); (4) DOS 3.3 (GETTEXT) oder ProDOS mit BASIC.SYSTEM 1.0 (GETTEXT.PRODOS); (5) RUN GETTEXT.DEMO; (6) für Videx und ProDOS leichte Anpassung von GETTEXT.DEMO erforderlich, s. Heft

RAMDISK.PAS
(1) Installierung einer CP/M-56K-RAM-Disk unter Turbo-Pascal; (2) Heft 12/85, S. 62; (3) IIe mit 64K-Karte; (4) CP/M 2.20 56K; Turbo-Pascal 2.0 oder 3.0; (5) RAMDISK (unter CP/M, nach Konvertierung mit APDOS und Compilieren auf Diskette); (6) entspricht der RAM-Disk aus Heft 6/1985, S. 55

JAHRESINHALT
WORT.SUCHER
WORT.SUCHER.OBJ
(1) Jahresinhaltsverzeichnis als binäre Textdatei mit Programm zur Suche von Stichwörtern; (2) Heft 12/85, S. 72; (3) II+, IIe oder IIc; 40 und 80 Z/Z (auch Videx); (4) DOS 3.3 und ProDOS; (5) RUN STICHWORT.SUCHER

Disk #13

(DOS 3.3; Heft 1/86)

QUICK.RANDOM
QUICK.SPEZIAL
QUICK.TASC
QUICK.DISK
(1) Demonstration von Quicksort; (2) Heft 1/86, S. 6; (3) II+, IIe oder IIc; 80-Zeichenkarte die INVERS/NORMAL erkennt; (4) DOS 3.3 oder ProDOS; (5) RUN QUICK.RANDOM, RUN QUICK.SPEZIAL (Demo für aufsteigend vorsortiert, absteigend vorsortiert, gleiche Elemente), RUN QUICK.DISK (für Diskettensortieren); (6) QUICK.TASC eignet sich als Quelltext für den Tasc-Compiler

QUICKSORT.DEMO
T.QUICKSORT
QUICKSORT
(1) Ampersand-Utility zur Sortierung eines eindimensionalen Arrays nach dem Quicksort-Algorithmus; (2) Heft 1/86, S. 16; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN QUICKSORT.-DEMO

VOK.TRAINER
VOK.COPY
VOK.PACK
VOK.BCOPY
GWS.INFO
GWS.VOK
(1) Vokabellernprogramm mit Editor für Vokabeln; (2) Heft 1/86, S. 20; (3) II+, IIe oder IIc; Laufwerk mit ggf. 40 Spuren; (4) DOS 3.3; (5) RUN VOK.-TRAINER; (6) GWS.INFO und GWS.VOK beinhalten den englischen Grundwortschatz

AGE.DEMO
T.AGE
AGE
(1) Ampersand-Erweiterung der Grafik-Befehle in Applesoft; (2) Heft 1/86, S. 30; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN AGE.DEMO

GRAFIK.DEMOS.2
(1) Demonstration effektvoller Grafiken; (2) Heft 1/86, S. 61; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN GRAFIK.DEMOS.2

Peeker-Sammeldisk #14 und #15

(Einzelpreis DM 28,–; Fortsetzungspreis DM 20,–)

Disk #14

(DOS 3.3; Heft 1 und 2/86)

T.PROTODOS
PROTODOS
(1) Programm zur Konvertierung von ProDOS- in DOS-3.3-Dateien; (2) Heft 1/86, S. 36; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) BRUN PROTODOS

DESIGNER.TEXT
(1) Entwurf und Änderung von Zeichensätzen für die SYSTEM.CHARSET-Datei; (2) Heft 1/86, S. 43; (3) II+, IIe oder IIc; (4) UCSD-Pascal 1.1 oder 1.2; (5) E(xecute) DESIGNER.TEXT nach dem Assemblieren (6) Datei mit GETDOS nach UCSD-Pascal übertragen

READPAS.PAS
(1) Programm zur Konvertierung von UCSD- in Turbo-Pascal-Textfiles; (2) Heft 1/86, S. 48; (3) II+ oder IIe; (4) CP/M mit Turbo-Pascal 2.0 oder 3.0; (5) nach dem Compilieren mit Option C von CP/M aus als COM-Datei starten; (6) Datei mit APDOS nach CP/M übertragen

KOMPLEMENT.DEMO
T.KOMPLEMENT
KOMPLEMENT
(1) Demo-Programm zur Komplement-Addition; (2) Heft 2/86 S. 6; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN KOMPLEMENT.DEMO

DOSMOVER.START
T.DOSMOVER
DOSMOVER
(1) Verschieben von DOS 3.3 in die LC zur Erweiterung des verfügbaren Speicherplatzes; (2) Heft 2/86, S. 17; (3) II+ mit LC, IIe oder IIc; (4) DOS 3.3; (5) RUN DOSMOVER.START

MATRIX
VEKTOR
AMPERSOFT1 –
AMPERSOFT8
LRS1
LRS2
(1) Demo-Programme zur Matrizenrechnung; (2) Heft 2/86, S. 29; (3) II+, IIe oder IIc; (4) DOS 3.3 mit Ampersoft und LRS-System; (5) s. Heft; (6) die Daten-Files sind wegen ihres Umfangs nicht enthalten und müssen selbst erstellt werden

INIT.SERIELL
(1) Programm zur Initialisierung einer der seriellen Schnittstellen beim IIc; (2) Heft 2/86, S. 34; (3) IIc; (4) DOS 3.3 oder ProDOS; (5) RUN INIT.SERIELL; (6) PIN-Werte ggf. im Programm anpassen

T.CHRGET.SPRUNG
CHRGET.SPRUNG
(1) Hilfsprogramm zur Applesoft-Befehlserweiterung; (2) Heft 2/86, S. 38; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) BRUN CHRGET.SPRUNG, dann §BELL zum Test

SC.MOVER.DEMO
T.SCREEN.MOVER
SCREEN.MOVER
(1) Programm zur Pufferung von bis zu 8 Bildschirmseiten in der LC; (2) Heft 2/86, S. 41; (3) II+ mit LC, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN SC.MOVER.DEMO; (6) unter ProDOS die Pufferzahl durch POKE 782,2 begrenzen

DARSTELLUNG.3D
(1) Darstellung dreidimensionaler Funktionen; (2) Heft 2/86, S.43; (3) II+ mit G/K, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN DARSTELLUNG.3D

TIPS5.TEXT
(1) Demo-Programme zur Assembler-Ein/Ausgabe in UCSD-Pascal; (2) Heft 2/86, S. 47; (3) II+, IIe oder IIc; (4) UCSD-Pascal 1.1 oder 1.2; (5) s. Heft

Disk #15

(DOS 3.3; Heft 3/86)
Achtung: Enthält sowohl DOS-3.3- als auch UCSD-Pascal-Directory!

REELL.1
REELL.1A
REELL.2 – REELL.10
(1) Demos zur internen Darstellung von reellen Zahlen; (2) Heft 3/86, S. 12; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN REELL.1 usw.

T.MAKESUB
MAKESUB
(1) Erstellung zusammenhängender Subdirectories unter ProDOS; (2) Heft 3/86, S. 20; (4) II+ (mit LC), IIe oder IIc; (4) ProDOS;
MAKESUB muß zuerst mit DOSTOPRO auf Ihre ProDOS-Diskette kopiert werden; (5) BRUN MAKESUB.

DATENDEMOS
SUBDEMOS
STRINGDEMOS
FILEDEMOS
(1) Demos zum Kyan-Aufbaukurs sowie String-Include-Datei, enthalten in STRINGDEMOS; (2) Heft 3/86, S. 32; (3) II+ (mit LC) IIe oder IIc; (4) ProDOS; Dateien müssen zunächst auf ProDOS-Diskette kopiert werden (z.B. mit DOSTOPRO); (5) Unter Kyan-Pascal zu SYS-Files compilieren.

FILETEST.TEXT
FIBTEST.TEXT
(1) FILETEST erzeugt Demo-Testfiles, die mit FIBTEST analysiert werden; (2) 3/86, S. 48; (3) II+, IIe oder IIc; (4) UCSD-Pascal 1.1 und 1.2; die Dateien müssen zunächst mit GETDOS (s.u.) auf Pascal-Diskette konvertiert werden; (5) FILETEST und FIBTEST compilieren und dann FILETEST und FIBTEST starten (in dieser Reihenfolge)

MESSWERT.START
MESSWERT
MESSWERT.TITEL
MW.ZEICHENSATZ
(1) Programm zur Auswertung von Meßwertkolonnen; (2) Heft 3/86, S. 58; (3) II+, IIe oder IIc; ggf. Epson-Drucker FX-80 mit Grafik-Interface; (4) DOS 3.3, ggf. in LC geschoben; (5) RUN MESSWERT.START; (6) das DOS wird gepatcht

GETDOS.TEXT
GETDOS.CODE
AND_7F.TEXT
AND_7F.CODE
(1) Verbesserte Version von GETDOS aus Heft 1/85, S. 70; Konvertierung von DOS-3.3-Textfiles in USCD-Pascal-TEXT-Files; speziell für Peeker-DOS-Sammeldisketten konzipiert; (3) II+, IIe oder IIc; 2 Drives; (4) USCD-Pascal 1.1 oder 1.2; (5) Ihre UCSD-Arbeitsdiskette booten, Sammeldisk #15 einlegen, E(xecute GETDOS, wieder Arbeitsdiskette in Drive 1 und eine der Peeker-Sammeldisketten mit UCSD-Textfiles in Drive 2 einlegen, Return drücken für DOS-Catalog, Nummer des Textfiles eingeben, Rest automatisch

Kyan-Pascal und Assembler

von Ulrich Stiehl

In Pascal können mit Hilfe von Prozeduren und Funktionen mit Wert- und Variablenparametern verhältnismäßig leicht Befehlserweiterungen implementiert werden. Doch sind solche simulierten Befehle zumeist alles andere als schnell und kompakt. Deshalb wird man aus Gründen der Geschwindigkeit und effizienten Speicherausnutzung in vielen Fällen Assemblerroutinen vorziehen müssen.

1 UCSD-Pascal (Apple-Pascal 1.1/1.2) bietet hierzu ein vollständiges Instrumentarium an, doch ist das Editieren, Assemblieren, Linken und Austesten von Assemblerroutinen kompliziert, schwerfällig und damit zeitraubend (vgl. z.B. die Maus-Routinen aus Peeker 4/85, S. 48ff.).

2 In Turbo-Pascal (Apple-CP/M-Version 3.0) kann man keine Assembler-Quelltexte eingeben, und der Inline-Code in Form von hexadezimalen Zahlen ist nur ein schwacher Ersatz (vgl. z.B. den CP/M-RAM-Disk-Driver aus Peeker 12/85, S. 62f.).

3 Der wesentliche Vorteil von Kyan-Pascal gegenüber Turbo- und UCSD-Pascal besteht darin, daß man problemlos Assemblerroutinen in Kyan-Quelltexte einbauen kann, denn der Kyan-Compiler ist zugleich auch ein 6502-Assembler. Hinzu kommt, daß man unter Kyan-Pascal von ROM-Routinen Gebrauch machen kann, was besonders denjenigen zustatten kommt, die sich bereits mit dem Monitor befaßt haben.

Beispiel

Nehmen wir einmal an, ein Pascal-Befehl solle die Ausgabe hexadezimaler Zahlen im Bereich von 0-255 simulieren (= Byte-Ausgabe). Die Kyan-Assemblerroutine würde folgendermaßen aussehen:

```
PROCEDURE HEXOUT (H: INTEGER);
BEGIN
#A
 STX T
 LDY #3
 LDA (SP),Y
 JSR $FDDA    ;PRBYTE
 LDX T
#
END;
```

Wenn man nunmehr die gleiche Routine in UCSD- und Turbo-Pascal zu schreiben versucht, so wird die Assembler-Überlegenheit von Kyan-Pascal evident.

Zum besseren Verständnis dieses Beitrages sollten Sie über Pascal-Kenntnisse (s. „Kyan-Pascal Grund- und Aufbaukurs" aus Peeker 2/86 und 3/86) sowie Assembler-Kenntnisse (s. 16seitiger Sonderteil „6502 leicht gemacht" aus Peeker 7/85) verfügen. Es sei jedoch darauf hingewiesen, daß Kyan-Programmierer nicht unbedingt Assembler lernen müssen, um die in den folgenden Peeker-Heften veröffentlichten Assemblerroutinen benutzen zu können, denn für deren Anwendung ist es lediglich erforderlich, daß man in das eigene Programm eine Include-Zeile in der Form
#I ROUTINEN
einfügt und sich darüber hinaus mit der Syntax der definierten Befehle vertraut macht.

1. #A...#-Befehl

```
PROGRAM MONITOR;
BEGIN
WRITELN ('Monitor-Exit');
#A
MON EQU $FF69
    JMP MON
#
END.
```

Eine Assemblerroutine wird mit dem Befehl
#A
eingeleitet, der *in einer eigenen Zeile* stehen muß, d.h. Return + „#A" + Return. Statt „#A" kann man auch „#a" schreiben. Mögliche Fehler:
□#A
Hier steht vor dem „#A" noch eine Leertaste (□), was nicht zulässig ist.
WRITELN; #A
Hier geht dem „#A" ein Pascal-Befehl in derselben Zeile voran.
#A LDA $C000
Hier folgt dem „#A" in derselben Zeile bereits der erste Befehl der Assemblerroutine. Dies führt zwar zu keiner Fehlermeldung, doch würde „LDA $C000" ignoriert werden.

Die mit „#A“ eingeleitete Folge von Assemblerinstruktionen wird mit einem nackten
#
abgeschlossen, das ebenfalls auf einer eigenen Zeile stehen muß, da es sonst mit einem „#“ von „LDA #$A0“ usw. verwechselt werden würde.
Die Befehle zwischen „#A“ und „#“ müssen der klassischen MOS-Syntax entsprechen, die gegenüber dem im Peeker benutzten Big Mac bzw. Merlin folgende Besonderheiten aufweist:

1. High/Low: Das High-Byte (HH) einer Adresse wird mit „#<“ und das Low-Byte (LL) mit „#>“ gekennzeichnet, z.B.

```
      LDA #<$1000 ;= 10
      LDY #>$1000 ;= 00
;oder
LABEL EQU $1000
      LDA #<LABEL ;= 10
      LDY #>LABEL ;= 00
```

2. Akku-Befehle: Die Akkumulator-Befehle ASL, LSR, ROL und ROR müssen durch „A“ ergänzt werden, z.B.

```
ASL A
LSR A
ROL A
ROR A
```

3. Labels: Es gibt gewisse vordefinierte Labels, insbesondere „T“ und „SP“ (s.u.), die man zwar benutzen, aber nicht selbst (um)definieren darf. Darüber hinaus sind alle Labelnamen tabu, die mit „L“ anfangen, weil der Kyan-Compiler mit „L“ durchnumerierte Labelnamen erzeugt (L1, L2, L3 usw.).

4. Pseudo-Opcodes: Die alte Kyan-Version 1.2 kennt nur die Pseudo-Opcodes ORG (= Origin), EQU (= Equation), DB (= Define Byte) und DW (= Define Word). Dieser Beitrag wurde geschrieben, als die Version 2.0 noch nicht vorlag. Über die Erweiterungen des 2.0-Assemblers werden wir zu einem späteren Zeitpunkt berichten.

2. Inline und Include

```
PROGRAM ASMMAIN;
 VAR C: CHAR;
BEGIN
 READLN (C);
#A
 LDA $C056  ;Mixed-Lores-Grafik
 LDA $C054
 LDA $C050
 LDA $C053
#
 READLN (C);
#A
 LDA $C051  ;Textdarstellung
 LDA $C052
#
END.
{-------------------------------}
PROGRAM ASMPROC;
VAR C: CHAR;
PROCEDURE SETLORES; {Include-Anfang}
BEGIN
#A
 LDA $C056
 LDA $C054
 LDA $C050
 LDA $C053
#
END;
PROCEDURE SETTEXT;
BEGIN
#A
 LDA $C051
 LDA $C052
#
END;   {Include-Ende}
BEGIN
 READLN (C);
 SETLORES;
 READLN (C);
 SETTEXT
END.
{-------------------------------}
PROGRAM ASMINCLUDE;
VAR C: CHAR;
#I ASMFILE
BEGIN
 READLN (C);
 SETLORES;
 READLN (C);
 SETTEXT
END.
```

Wie man aus den drei Demos ersehen kann, gibt es prinzipiell drei Möglichkeiten, um Assembler-Quelltexte in einen Pascal-Quelltext einzubauen:

1. Eingebettete Routinen: Zwar lassen sich Assemblerroutinen an beliebigen Stellen des Hauptprogramms einfügen (s. Listing ASMMAIN), doch sollte man dies nur in Spezialfällen sowie zum Austesten neuer Befehle tun, denn die Mischung von Assembler- und Pascal-Quelltext führt zu einem unübersichtlichen Programmierstil.

2. Gesonderte Routinen: Statt dessen wird empfohlen, alle Assemblerroutinen als Prozeduren oder Funktionen zu definieren, wie dies im Listing ASMPROC demonstriert wird. Wie ersichtlich, wird eine Assemblerprozedur mit BEGIN eingeleitet und mit END abgeschlossen, wobei das END gleichzeitig das RTS ersetzt. *Am Ende einer Assemblerprozedur darf also niemals ein RTS stehen,* weil sonst der END-Befehl nicht mehr ausgeführt werden würde und damit bei verschachtelten Prozeduren die internen Zeiger völlig durcheinander gerieten.
Assemblerprozeduren und -funktionen werden dort eingefügt, wo sich auch die „normalen“ Prozeduren und Funktionen befinden, also im Anschluß an den CONST-TYPE-VAR-Teil des Hauptprogramms.

3. Include-Routinen: Immer wieder benötigte Assemblerroutinen kann man zu gesonderten Include-Dateien zusammenfassen, die dann beim Compilieren in beliebige andere Quelltexte eingespielt werden können (s. ASMINCLUDE). Eine Include-Datei sollte sich in demselben Volume- oder Subdirectory des ProDOS-Datenträgers befinden, in dem der Quelltext des Hauptprogramms gespeichert ist. Auf „#I“ bzw. „#i“ folgt der Include-Dateiname in der üblichen ProDOS-Syntax, z.B.
#I /VOLUME/SUB/ASMFILE
Der „#I“-Befehl wird im Gegensatz zum „#A“-Befehl nicht durch „#“ abgeschlossen.

3. Speicherorganisation

Assemblerroutinen müssen auf die Speicherorganisation unter Kyan-Pascal Rücksicht nehmen (vgl. auch **Abb. 1: Speicherorganisation**):
Nullseite ($0000-$00FF): Die Speicherstellen **T** bis T+14 (unter Version 1.2 absolut $0010-$001E) werden vom compilierten Kyan-Programm als „Scratch-Pad“ für temporäre Werte benutzt, doch darf eine Assemblerroutine ebenfalls diese Speicherstellen verwenden. Dies sind immerhin 15 Bytes, was in den meisten Fällen ausreichen dürfte. Das Label „T“ (für „temporär“) ist vordefiniert. Der übrige Teil der Nullseite sollte nicht verändert werden. Eine Ausnahme stellen diejenigen Zero-Page-Speicherstellen dar, die mit der Standard-Ein/Ausgabe zusam-

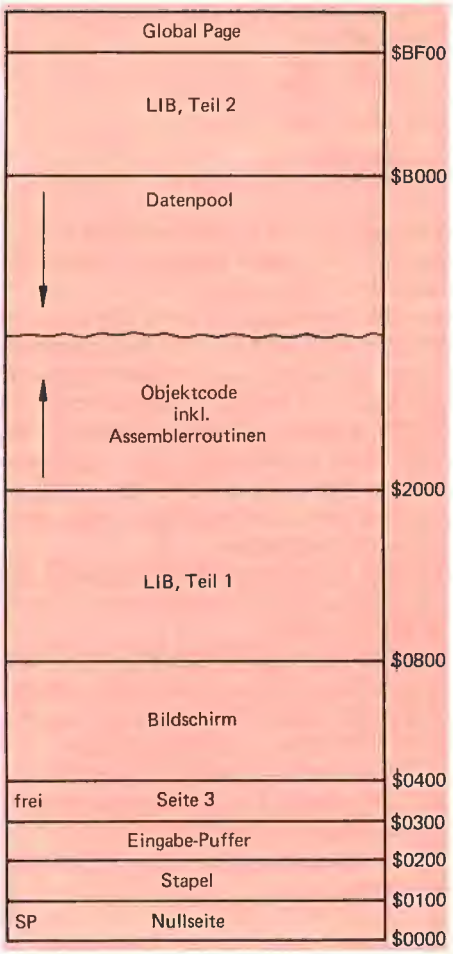


Abb. 1: Speicherorganisation

menhängen, z.B. CSWL, KSWL, BASL usw., da auch das compilierte Kyan-Programm von COUT und RDKEY Gebrauch macht.
Stapel ($0100-$01FF): Der Prozessorstapel wird unter Kyan-Pascal wie üblich benutzt und ist für eigene Assemblerroutinen tabu. Notfalls kann man den Bereich $0100-$0110 als Scratch-Pad verwenden.
Eingabe-Puffer ($0200-$02FF): Dieser Bereich wird erstens als Tastatur-Eingabe-Puffer sowie zweitens (ab $0280) als Zwischenspeicher für ProDOS-Dateinamen benutzt. Eine Assemblerroutine darf deshalb den Eingabe-Puffer in gleicher Weise verwenden.
Page 3 ($0300-$03CE): Mit Ausnahme der Vektoren für Reset usw. am Ende der Page 3 ist dieser Bereich frei und kann deshalb z.B. für Drucker-Treiber verwendet werden.
Bildschirm ($0400-$07FF): Dies ist der übliche Textbildschirm-Speicher.
LIB Teil 1 ($0800-$1FFF): Der erste Teil der Runtime-Library wird in den Bereich ab $0800 gelegt.
Programm ($2000-AFFF): Normalerweise beginnt der Objektcode des Kyan-Programms einschließlich der darin enthaltenen Assemblerroutinen ab $2000. Falls jedoch hochauflösende Grafik benötigt wird, so muß man den entsprechenden Quelltext mit

```
#A
 ORG $4000
#
PROGRAM GRAFIK; usw.
```

einleiten, so daß der HGR-Bereich $2000-$3FFF als Grafik-Speicher ausgespart wird. Wird keine Grafik oder z.B. nur Lores-Grafik benutzt, so steht der gesamte Bereich $2000-$AFFF für Programm und Daten zur Verfügung (Programmspeicher ab $2000 aufwärts, Datenspeicher ab $AFFF abwärts). Übrigens befinden sich die ProDOS-I/O-Puffer für geöffnete Dateien unterhalb des Datenspeichers, also quasi zwischen Programm- und eigentlichem Datenspeicher.
LIB Teil 2 ($B000-$BEFF): Der zweite Teil der Runtime-Library wird in den Bereich ab $B000 gelegt.
Global-Page ($BF00-$BFFF): Die ProDOS-Global-Page wird wie üblich benutzt.
PRODOS ($D000-$FFFF, LC): Eigentliches PRODOS in der Language-Card. Unter Kyan-Pascal können offenbar alle ProDOS-Versionen benutzt werden. Dies ist insbesondere für diejenigen wichtig, die mit Fremdlaufwerken (Erphi-Controller usw.) arbeiten.
ROM ($D000-$FFFF, ROM): Eine Assemblerroutine kann – im Gegensatz zu UCSD-Pascal – immer davon ausgehen, daß das ROM eingeschaltet bzw. lesefähig ist. Das compilierte Kyan-Pascal benutzt selbst fast keine ROM-Routinen, doch kann man in eigenen Assemblerprozeduren von allen ROM-Routinen Gebrauch machen, wenn man danach wieder die Nullseite in den alten Zustand zurückversetzt.

4. Prozessorregister

Eine Assemblerroutine braucht weder den Akkumulator (A) noch das Y-Register (Y) zu retten. Desgleichen kann man auch das Statusregister (P) ignorieren; allerdings darf sich der Prozessor nach dem Verlassen der Assemblerroutine nicht im Dezimalmodus befinden.
Dagegen muß das X-Register stets gerettet werden, weil es von Kyan-Pascal als wichtigster interner Zeiger benutzt wird. Es empfiehlt sich, die Speicherstelle T stets für das X-Register zu reservieren. Beispiel:

```
#A
 STX T      ; X zwischenspeichern
 JSR $FC58  ; HOME-Befehl
 LDX T      ; X wiederherstellen
#
```

5. Parameterübergabe

Die Übergabe von Parametern an Assemblerprozeduren und -funktionen ist nicht trivial, denn man muß erstens wissen, in welcher Reihenfolge die Parameter im Speicher abgelegt werden, und zweitens muß man mit der internen Struktur der Datentypen vertraut sein. Das Programm **HEXTYPEN** zeigt, in welcher Reihenfolge und in welcher hexadezimalen Form alle vordefinierten Datentypen gespeichert werden. Darüber hinaus demonstriert das Programm **KYANASM** alle denkbaren Fälle der Parameterübergabe. Sie sollten deshalb beide Programme genau studieren, nachdem Sie sich mit den nachfolgenden Grundbegriffen vertraut gemacht haben.

1. Datenkopf: Wenn eine Prozedur oder Funktion aufgerufen wird, so zeigt der sog. Stack-Pointer auf den *Anfang des Datenspeichers* des Unterprogramms. **SP** ist ein vordefiniertes Zero-Page-Label (absolut $0004-$0005), das indirekt indiziert verwendet werden kann.
Die ersten 3 Bytes des Datenspeichers, die den Datenkopf bilden, können ignoriert werden.
LDA (SP), Y ;wobei Y = 0
enthält die Nummer des Levels bzw. Verschachtelungsgrades, z.B. $01 für Unterprogramme, die vom Hauptprogramm aufgerufen werden.
LDA (SP),Y ; wobei Y = 1 (LL) und
LDA (SP),Y ; wobei Y = 2 (HH)
zeigen auf das *Ende des Datenspeichers* („Local-Top").

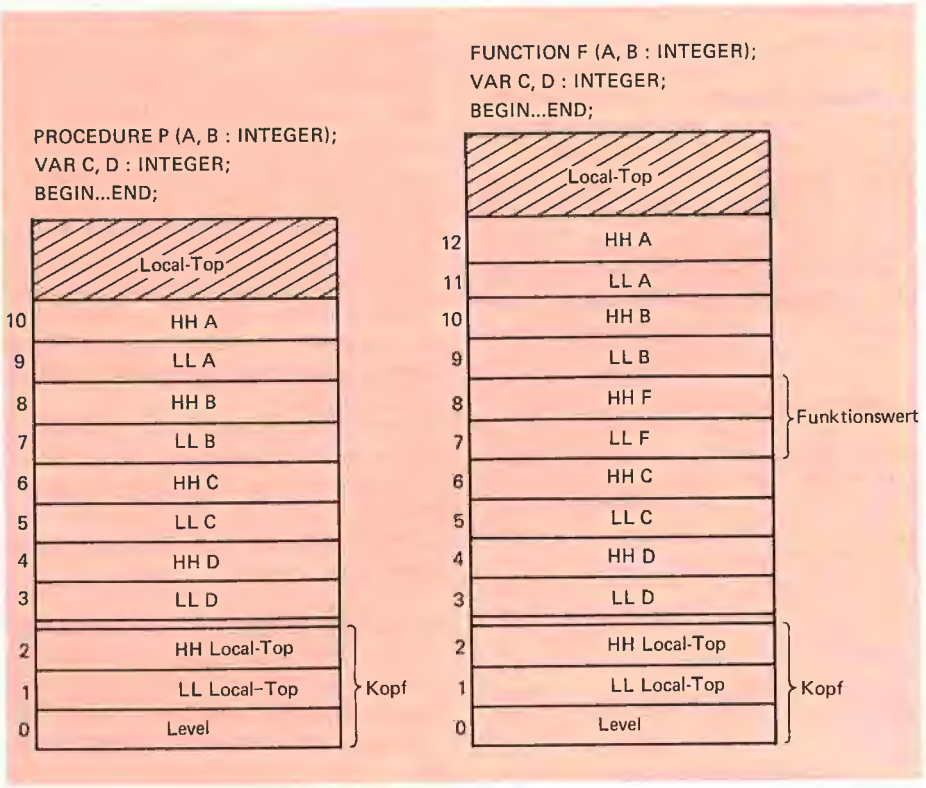


Abb. 2: Parameterübergabe

2. Eigentliche Daten: Ab dem dritten Byte, d.h. ab
LDA (SP), Y ; wobei Y = 3
werden die Daten abgelegt, und zwar bei Wertparametern als Werte und bei Variablenparametern als Zeiger auf die Werte. Aus Gründen der Vereinfachung wollen wir nur Ganzzahlen betrachten, zumal diese bei Assemblerroutinen am häufigsten benutzt werden. Ganzzahlen belegen im Speicher 2 Bytes in der Form Low Byte – High Byte. Beispiele:

```
PROCEDURE P (A: INTEGER);
BEGIN ... #A ... # ... END;
```

hat den Datenaufbau
LDA (SP), Y = 4: HH von A
LDA (SP), Y = 3: LL von A

```
PROCEDURE P (A, B: INTEGER);
BEGIN ... #A ... # ... END;
```

hat den Datenaufbau
LDA (SP), Y = 6: HH von A
LDA (SP), Y = 5: LL von A
LDA (SP), Y = 4: HH von B
LDA (SP), Y = 3: LL von B

```
PROCEDURE P (A, B: INTEGER);
VAR C: INTEGER;
BEGIN ... #A ... # ... END;
```

hat den Datenaufbau
LDA (SP), Y = 8: HH von A
LDA (SP), Y = 7: LL von A
LDA (SP), Y = 6: HH von B
LDA (SP), Y = 5: LL von B
LDA (SP), Y = 4: HH von C
LDA (SP), Y = 3: LL von C

```
FUNCTION F (A: INTEGER): INTEGER;
BEGIN ... #A ...# ...END;
```

hat den Datenaufbau
LDA (SP), Y = 6: HH von A
LDA (SP), Y = 5: LL von A
LDA (SP), Y = 4: HH von F
LDA (SP), Y = 3: LL von F

```
FUNCTION F (A, B: INTEGER): INTEGER;
VAR C, D: INTEGER;
BEGIN ... #A ... # ... END;
```

hat den Datenaufbau
LDA (SP), Y = 12: HH von A
LDA (SP), Y = 11: LL von A
LDA (SP), Y = 10: HH von B
LDA (SP), Y = 09: LL von B
LDA (SP), Y = 08: HH von F
LDA (SP), Y = 07: LL von F
LDA (SP), Y = 06: HH von C
LDA (SP), Y = 05: LL von C
LDA (SP), Y = 04: HH von D
LDA (SP), Y = 03: LL von D

Aus den Beispielen (vgl. auch **Abb. 2: Parameterübergabe**) können wir folgende Regeln ableiten:

HEXTYPEN

```
PROGRAM HEXTYPEN;

{Dieses Programm zeigt, in welcher Reihenfolge
 und in welcher Hex-Form Variablen im Speicher
 abgelegt werden. U.Stiehl/Dez. 85}

VAR
TRENN1: CHAR;
I:      INTEGER;
R:      REAL;
C:      CHAR;
TRENN2: CHAR;
IA:     ARRAY [1..3] OF INTEGER;
CA:     ARRAY [1..3] OF CHAR;
TRENN3: CHAR;
RD:     RECORD
        RI: INTEGER;
        RR: REAL;
        RS: ARRAY [1..3] OF CHAR
        END;
TRENN4: CHAR;
RA:     ARRAY [1..3] OF RECORD
        RI: INTEGER;
        RR: REAL;
        RS: ARRAY [1..3] OF CHAR
        END;
TRENN5: CHAR;
B1:     BOOLEAN;
B2:     BOOLEAN;
TRENN6: CHAR;
AUFZ:   (NULL, EINS, ZWEI);
TRENN7: CHAR;
MENGE:  SET OF 0..63; {trotzdem 32 Bytes!}
TRENN8: CHAR;

PROCEDURE ZEIGER (VAR I: INTEGER);
VAR A,B,C: CHAR;
BEGIN
A := CHR (241);       {F1}
B := CHR (242);       {F2}
C := CHR (243);       {F3}
#A
  JMP $FF69 ;Monitor-Exit
#
END;

BEGIN
TRENN1 := CHR (255);  {FF}
I  := 5;              {05 00 wird spaeter erneut initialisiert!}
R  := 1234567890.123; {01 23 45 67 89 01 23 09}
C  := '1';            {31}

TRENN2 := CHR (255);  {FF}
IA[1] := 257;         {01 01}
IA[2] := 258;         {02 01}
IA[3] := 259;         {03 01}
CA    := 'ABC';       {41 42 43}

TRENN3 := CHR (255);  {FF}
RD.RI := 513;         {01 02}
RD.RR := 123456789.0; {01 23 45 67 89 00 00 08}
RD.RS := 'DEF';       {44 45 46}

TRENN4 := CHR (255);  {FF}
FOR I := 1 TO 3 DO
BEGIN
RA[I].RI := I+512;    {01 02,02 02,03 02}
RA[I].RR := 1.0E+88;  {01 00 00 00 00 00 00 88}
RA[I].RS := 'abc';    {61 62 63}
END;

TRENN5 := CHR (255);  {FF}
B1 := TRUE;           {01}
B2 := FALSE;          {00}

TRENN6 := CHR (255);  {FF}
AUFZ := ZWEI;         {02 00}

TRENN7 := CHR (255);  {FF}
MENGE := [0,7,8,15,16,23,24,31,32,39,40,47,48,55,56,63];
         {81, 81,  81,   81,   81,   81,   81,   81,   00 usw.}

TRENN8 := CHR (255);  {FF}
I := 10;              {0A 00}
ZEIGER (I);
END.
```

Peeker-Sammeldisk #16

DOS-3.3-Format
Einzelpreis DM 28,–; Fortsetzungspreis DM 20,–

Extra für Fortsetzungsbezieher:
B 027 APPLESOFT. EDITOR

Binäres Rechnen, Teil 2:

A 002 M8TEST
T 004 T.M8RLN
B 002 M8RLN
T 004 T.M8LRN
B 002 M8LRN
T 004 T.M8RLT
B 002 M8RLT
T 004 T.M8LRT
B 002 M8LRT
A 003 M16TEST
T 005 T.M16RLN
B 002 M16RLN
T 005 T.M16LRN
B 002 M16LRN
T 004 T.M16RLT
B 002 M16RLT
T 004 T.M16LRT
B 002 M16LRT
A 002 MULT8RLN.DEMO
T 008 T.MULT8RLN
B 002 MULT8RLN
A 003 MULT16RLN.DEMO
T 008 T.MULT16RLN
B 002 MULT16RLN
A 003 D16TEST
T 005 T.D16V1
B 002 D16V1
T 004 T.D16V2
B 002 D16V2
T 004 T.D16V3
B 002 D16V3
T 004 T.D16V4
B 002 D16V4
A 003 D24TEST
T 004 T.D24V2
B 002 D24V2

A 003 DIV16.DEMO
T 007 T.DIV16
B 002 DIV16
A 003 DIV24.DEMO
T 008 T.DIV24
B 002 DIV24

Sonderzeichen auf dem Imagewriter:

A 015 FONT.LOADER
T 023 T.FONT.LOADER.OBJ
B 003 FONT.LOADER.OBJ
T 012 T.FONT.CONVERT
B 002 FONT.CONVERT
B 005 ASCII.FONT
B 005 DEUTSCH.FONT
B 005 SONDERZEICHEN.FONT

Haushaltsverwaltung:

A 038 HAUSHALT
A 038 HAUSHALT.40
T 002 DATEI

Kyan-Pascal und Assembler:

T 037 KYANASM
T 020 HEXTYPEN

Interaktive Funktionseingabe:

T 005 FUNKSTART.TEXT
T 003 FUNKDEMO.TEXT
T 002 FUNKUNIT.TEXT
T 002 FUNKEXEC.TEXT

Zweistimmige Melodien:

A 015 MUSIK.EDITOR
T 010 T.TONROUTINE
B 002 TONROUTINE
B 018 M.SONATA
A 002 SONATA

Hüthig Software Service
Postfach 10 28 69 · 6900 Heidelberg

Peeker-Quickie

DUMP80REL
(vgl. Heft 4/85. S. 33)

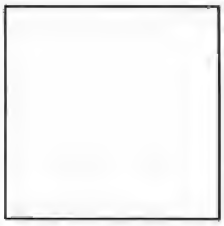
Relokativer 80-Z/Z-Screen-Dump für IIe/c. Drucker in Slot 1. Getestet auf Epson- und Imagewriter-Druckern. Aufrufen mit CALL A, wobei Adresse A beliebig festgelegt werden kann.

```
10 DATA 173,24,192,48,1,96,162,1,181,36
15 DATA 157,250,2,181,40,157,252,2,181,54
20 DATA 157,254,2,202,16,238,169,1,32,149
25 DATA 254,169,137,32,237,253,169,206,32,237
30 DATA 253,169,0,141,249,2,32,193,251,162
35 DATA 0,138,74,168,144,5,141,84,192,176
40 DATA 3,141,85,192,177,40,9,128,201,160
45 DATA 176,2,105,64,141,84,192,32,237,253
50 DATA 232,224,80,144,222,169,141,32,237,253
55 DATA 238,249,2,173,249,2,201,24,144,202
60 DATA 162,1,189,250,2,149,36,189,252,2
65 DATA 149,40,189,254,2,149,54,202,16,238,96
70 A = 768: REM A beliebig, hier z.B. 768
75 FOR X = A TO A + 120: READ Y: POKE X,Y: NEXT
```

Ampersand Verlag

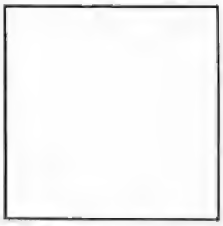
Teltower Damm 168
D-1000 Berlin 37
(030) 815 80 69

Apple II Assembler Programmierung
Wagner



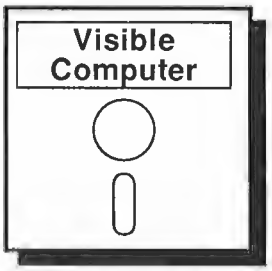
Das Assembler-Lehrbuch für BASIC-Kenner!
Roger Wagner, Autor vieler bekannter Software-Pakete, schrieb eine monatliche Kolumne über Assembler-Programmierung in der Zeitschrift SOFTALK. Der vorliegende Band faßt diese Reihe, korrigiert und erweitert, zusammen. Eine stufenweise Einführung in die Befehle und Strukturen der 6502 Assemblersprache, mit vielen Beispielen von der einfachen Tongenerierung bis zum Diskettenzugriff.
277 Seiten.
3-89058-003-3 DM 48,--
komplett mit Disk
3-89058-005-X DM 89,--

Apple II Raster Grafik
Stanton



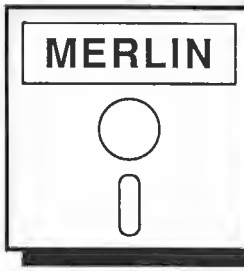
Die Qualität kommerzieller Arcadespiele läßt sich mit Applesoft BASIC alleine nicht erreichen. Das Buch führt in die Eigenarten der hochauflösenden Apple-Grafik ein und präsentiert schließlich eine Reihe extrem schneller Assembler-Routinen, mit denen Sie viele Effekte bekannter Spiele selbst programmieren können.
Gute BASIC-Kenntnisse werden vorausgesetzt, eine kurze Einführung in Assembler-Programmierung wird gegeben.
299 Seiten.
3-89058-006-8 DM 49,--
komplett mit Diskette:
3-89058-008-4 DM 89,--

Visible Computer



Ein Simulationsprogramm, das Sie in das Innere des 6502-Mikroprozessors führt.
Sie sehen auf dem Bildschirm, wie die einzelnen Instruktionen in Zeitlupe ausgeführt werden, wie sich Register und Flags verändern. Ein unverzichtbares Hilfsmittel beim Erlernen der Assemblerprogrammierung, danach ein wertvolles Werkzeug beim Testen Ihrer Programme.
Komplett mit einem 6502 Editor/Assembler und einem Lehrbuch zur Maschinenprogrammierung (deutsch, 150 Seiten).
3-89058-019-X DM 129,--

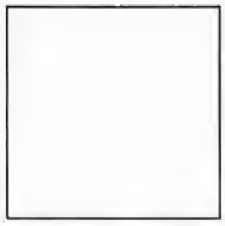
MERLIN



Ein professioneller Macro-Assembler für die Apple II-Familie. Neben allen Standard-Features bietet MERLIN u.a.:
- Editor mit globalen Such- und Ersetzfunktionen
- liest und schreibt DOS 3.3 Text und Binärfiles
- Unterstützt 65C02-Opcodes
- enthält einen Disassembler
- kompatibel mit vielen 80-Zeichenkarten und natürlich mit Apple IIe und Apple IIc.

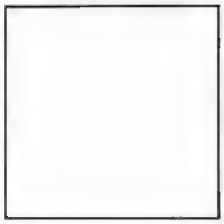
Deutsches Handbuch (170 S.)
3-89058-024-6 DM 198,--

Apple II Schaltpläne
Gayler



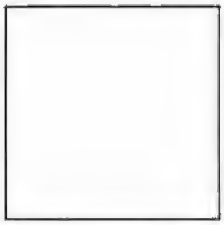
Eine detaillierte Beschreibung der Schaltungen des Apple II und Apple IIplus.
Wenn Sie Ihren Apple selbst reparieren, Interface-Karten oder Schaltungserweiterungen entwerfen oder einfach nur besser über das Innenleben Ihres Apple Bescheid wissen wollen - dieses Buch bietet Ihnen eine Fülle an Informationen, Schaltpläne und Zeitdiagramme, Theorie und praktische Tips.
215 Seiten DIN A4.
3-89058-012-2 DM 64,--

Apple Pascal Trickkiste
DeGroat



Eine Sammlung von Utilities für den Pascal-Programmierer:
P-Code-Decodierer, Systemadressen, File-Konverter (von DOS nach Pascal und zurück), DIsk-Zapper, verbesserte Ein/-Ausgabe-Prozeduren, Grafik-Routinen, Textformatierer und viele wichtige Informationen, Tips und Tricks.
248 Seiten.
3-89058-030-0 DM 48,--
komplett mit Disk:
3-89058-032-7 DM 89,--

Mikrocomputer Grafik
Myers



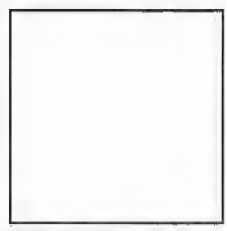
Endlich anspruchsvolle Apple-Grafik Für BASIC-Programmierer. Mikrocomputer Grafik
- enthält fast 80 lauffertige Programme, die die beschriebenen Konzepte illustrieren.
- beschreibt Hidden Line- und Hidden Surface-Techniken, Skalierung, Rotation und Translation von Grafiken.
- bietet eine Einführung in die Animationstechnik.

292 Seiten.
3-89058-000-9 DM 49,--
Komplett mit Diskette:
3-89058-002-5 DM 89,--

Rückgaberecht

Wenn Ihnen ein Buch wider Erwarten nicht gefällt, dann können Sie es innerhalb von 10 Tagen zurückschicken und bekommen, sofern das Buch unbeschädigt ist, den Kaufpreis erstattet.
Dieses Recht gilt nur bei Direktbestellung beim Verlag und nicht für Software oder Begleitdisketten.

Apple Pascal Grafik
Swan



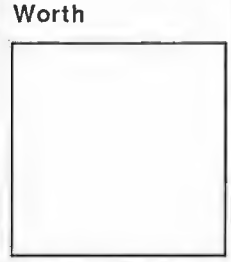
22 Pascal-Programme, mit denen Sie die Grafikmöglichkeiten Ihres Apple voll ausschöpfen:
DESIGNER ermöglicht Ihnen den Entwurf eigener Zeichensätze, mit GREDIT erstellen Sie komplexe Bildschirm-Grafiken, PRINTFOTO bringt Ihre Entwürfe aufs Papier.
Darüberhinaus bietet das Buch Fülle fertiger Prozeduren, die Sie zeitsparend in Ihre eigenen Programme einbauen können.
280 Seiten.
3-89058-009-2 DM 49,--
komplett mit Disk:
3-89058-011-4 DM 89,--

Applesoft Trickkiste
Golding

Alles, was Sie bei Applesoft bisher vermißt haben:
- Eine INPUT-Routine, mit der Sie auch Kommas und Doppelpunkte eingeben können
- Ein PRINT USING Kommando für formatierte Ausgabe
- eine schnelle GARBAGE COLLECTION

und viele andere nützliche Utilities, dazu detaillierte Informationen über die Arbeitsweise des BASIC-Interpreters, Einsprungadressen, Systemvariablen.
304 Seitem.
3-89058-033-5 DM 44,--
komplett mit Disk
3-89058-035-1 DM 89,--

Apple ProDOS Handbuch
Worth



Don Worth und Peter Lechner sind die Autoren von "Beneath Apple DOS", der Bibel aller DOS 3.3-Benutzer. In ihrem neuen Buch haben Sie sich ProDOS vorgenommen und mit der ihnen eigenen Gründlichkeit zerlegt. Ausführliche Beschreibung der Arbeitsweise, des "MLI" und des BASIC.SYSTEM. Das Standardwerk für jeden ProDOS-Anwender.
270 Seiten.
3-89058-036-X DM 46,--
komplett mit Disk
3-89058-038-6 DM 89,--

Bestellcoupon

Ampersand Verlag, Teltower Damm 168, 1000 Berlin 37

Name: .....

Anschrift: .....

.....

○ *V-Scheck liegt bei (spesenfreie Lieferung)*
○ *per Nachnahme (zzgl. DM 3,-- Versandspesen)*

Anz	Titel oder ISBN-Nummer	Preis

Datum & Unterschrift

```
Nach dem Sprung in den Monitor mit $FF69 findet man mit
0004.0005 und AF7D.AFFD folgendes vor:

0004: 7D AF:                        zeigt auf AF7D = Local-Bottom (= SP)

----------------- Lokale Variablen ------------------------------------

AF7D: 01:                           Level der Prozedur Zeiger
AF7E: 85 AF:                        zeigt auf AF85 = Local-Top + 1
AF80: F3 F2 F1:                     Lokale Variablen C, B, A
AF83: FB AF:                        zeigt auf AFFB = VAR-Parameter I

AF85: XX XX XX XX XX:               Diese 5 Bytes hier ignorieren

----------------- Globale Variablen -----------------------------------

AF8A: FF:                           TRENN8 (gleichzeitig Global-Bottom)
AF8B: 81 81 81 81 81 81 81 81:      Mengenelemente (Bits)   0 -  63
AF93: 00 00 00 00 00 00 00 00:      Mengenelemente (Bits)- 127 unbenutzt
AF9C: 00 00 00 00 00 00 00 00:      Mengenelemente (Bits) 128 - 191 unbenutzt
AFA3: 00 00 00 00 00 00 00 00:      Mengenelemente (Bits) 192 - 255 unbenutzt

AFAB: FF:                           TRENN7
AFAC: 02 00                         ZWEI          AUFZ

AFAE: FF:                           TRENN6
AFAF: 00:                           FALSE:        B2
AFB0: 01:                           TRUE:         B1

AFB1: FF:                           TRENN5
AFB2: 01 02:                        513:          RA[1].RI  1. Satz
AFB4: 01 00 00 00 00 00 00 88:      1.0E+88:      RA[1].RR
AFBC: 61 62 63:                     'abc':        RA[1].RS
AFBF: 02 02:                        514:          RA[2].RI  2. Satz
AFC1: 01 00 00 00 00 00 00 88:      1.0E+88:      RA[2].RR
AFC9: 61 62 63:                     'abc':        RA[2].RS
AFCC: 03 02:                        515:          RA[3].RI  3. Satz
AFCE: 01 00 00 00 00 00 00 88:      1.0E+88:      RA[3].RR
AFD6: 61 62 63:                     'abc':        RA[3].RS

AFD9: FF:                           TRENN4
AFDA: 01 02:                        513:          RD.RI     Record
AFDC: 01 23 45 67 89 00 00 08:      123456789.0   RD.RR
AFE4: 44 45 46:                     'DEF':        RD.RS

AFE7: FF:                           TRENN3
AFE8: 41:                           'A':          CA[1]     Char-Array CA [1..3]
AFE9: 42:                           'B':          CA[2]
AFEA: 43:                           'C':          CA[3]
AFEB: 01 01:                        257:          IA[1]     Integer-Array IA [1..3]
AFED: 02 01:                        258:          IA[2]
AFEF: 03 01:                        257:          IA[3]

AFF1: FF:                           TRENN2
AFF2: 31:                           '1':            C       Char-Variable
AFF3: 01 23 45 67 89 01 23 09:      1234567890.123  R       Real-Variable
AFFB: 05 01:                        5               I       Integer-Variable

AFFD: FF                            TRENN1 (gleichzeitig Global-Top)

AFFE: XX XX                         Diese 2 Bytes hier ignorieren
B000:                               Beginn der Kyan-Library, 2. Teil
```

Hinweis zur Version 2.0

Kurz vor Redaktionsschluß erhielten wir am 28.2.1986 das Vorausexemplar zur Version 2.0. Danach hat sich die Parameterübergabe geändert. Der Parameterkopf umfaßt jetzt 5 Bytes statt bisher 3 Bytes. Außerdem beginnen die reservierten Labels jetzt mit "_', also _SP, _T, _Lxxx usw. übrigens gibt es jetzt 25 Pseudo-Opcodes sowie diverse andere Neuerungen. Beispielsweise kann man jetzt den bislang nur ausdruckbaren Assemblerquelltext auf Diskette speichern und nach der Optimierung über den neuen AS(embler)-Befehl direkt assemblieren. us

1. Auf den Datenkopf, d.h. ab Y = 3, folgen zunächst die lokalen Variablen des Unterprogramms in der umgekehrten Reihenfolge, wie sie definiert wurden.
2. Falls eine Funktion (anstelle einer Prozedur) vorliegt, so folgt jetzt der Funktionswert.
3. Zum Schluß folgen die Parameter der Prozedur oder Funktion in der umgekehrten Reihenfolge, wie sie definiert wurden.

6. Übungen

Übung 1: Versuchen Sie mit Hilfe der Abb. 2 zu ergründen, welche Zahlen das folgende Programm in welcher Reihenfolge ausgibt.

```
PROGRAM UEBUNG1;
 FUNCTION F (A, B: INTEGER): INTEGER;
 VAR C, D: INTEGER;
 BEGIN
 F := 255; C := 3; D := 4;
 WRITELN (A, ' ', B, ' ', C, ' ', D);
#A
   STX T
   LDY #3
A1 LDA (SP),Y
   JSR $FDDA    ;Hexzahl ausgeben
   LDA #$A0     ;Leertaste
   JSR $FDED    ;ausgeben
   INY
   CPY #13      ;insgesamt 5 Hexzahlen
   BNE A1       ;ausgeben
   JSR $FD8E    ;Return ausgeben
   LDX T
#
 END;
BEGIN
 WRITELN (F (1, 2))
END.
```

Übung 2: Wenn Sie früher bereits in UCSD- oder Turbo-Pascal programmiert haben, so versuchen Sie, den Namen der folgenden Kyan-Funktion zu erraten:

```
FUNCTION Name???: BOOLEAN;
BEGIN
#A
 LDA $C000
 ROL A
 ROL A
 AND #$01
 LDY #3
 STA (SP),Y
#
END;
```

Kurzhinweise

1. Zweck:
Muster-Assemblerprogramme für Kyan-Pascal.
2. Konfiguration:
Apple II+/e/c; ProDOS; Kyan-Pascal 1.2 oder 2.0.
3. Test:
Zunächst Dateien KYANASM und HEXTYPEN von Sammeldisk auf Ihre Kyan-Arbeitsdiskette mit DOSTOPRO konvertieren und danach compilieren.
4. Sammeldisk:
KYANASM
HEXTYPEN
(Quelltexte)

KYANASM

```
{Diverse Assembler-Muster fuer Kyan-Pascal}
{U.Stiehl/Dez. 85}


PROGRAM KYANASM;

TYPE
AUFZ   = (NULL, EINS, ZWEI);
STRING = ARRAY [1..3] OF CHAR;
MENGE  = SET OF 0..7;

VAR
R:      REAL;
X,Y:    INTEGER;
S,W:    INTEGER;
C:      CHAR;
B1,B2:  BOOLEAN;
A1,A2:  AUFZ;
STR:    STRING;
M:      MENGE;

{Interner Aufbau von Kyan-Datentypen:

Real:         8 Bytes; VM MM MM MM MM MM MM EE
                       LL                   HH
                       (Naeheres siehe unten)

Integer:      2 Bytes; 76543210 FEDCA98  Bits
                       LL       HH
                       (Bit F ist Vorzeichenbit)
                       ( 0 bis  32767: positiv)
                       (-1 bis -32768: negativ)

Char:         1 Byte;  C := 'X':     ASCII 32-127
                       C := CHR (X): ASCII  0-255

Boolean:      1 Byte;  01 = TRUE, 00 = FALSE

Pointer:      2 Bytes; LL HH (16-Bit-Adresse)

Aufzaehlung: 2 Bytes; LL HH (Natuerliche Zahl)
                            (0 bis theoretisch 65535)

String:       X Bytes; C C C C C C C... (Wert)
                       1 2 3 4 5 6 7... (Index)
                       (Index: 1 bis 32767)

Menge:       32 Bytes; 256 Bits: Bit 1 = IN Menge

             (stets)             Bit 0 = NOT IN Menge
                       Beispiel fuer SET OF 0..7:
                       76543210  Bit-Nummern des 1. Bytes
                       10100010  Elemente 7, 5, 1 IN Menge}

{-----------Nuetzliche Routinen-------------}

{HOME: Bildschirm loeschen}
{Prozedur ohne Parameter: HOME;}
{Stackaufbau hier irrelevant}
{T = Vordefiniertes Label fuer Temporary}
{Auf Zero-Page sind frei: T bis T+$0E,}
{physisch: $0010 (T) bis $001E (T+$0E)}

PROCEDURE HOME;
BEGIN
#A
    STX T       ;Stets X in Temp ($0010) retten!
    JSR $FC58   ;HOME
    LDX T       ;Nachher stets X wieder laden!
#
END;            {Ersetzt RTS}


{GOTOXY (X,Y): Htab X, Vtab Y}
{Prozedur mit Wertparameter: GOTOXY (X, Y);}
{X (0-79) und Y (0-23) sind Integervariablen}
{Stackaufbau, physisch von unten nach oben:}
{Y=6: HH X ignorierbar, d.h. 0 wegen X: 0-79}
{Y=5: LL X von GOTO (Y,Y)}
{Y=4: HH Y Ignorierbar, d.h. 0 wegen Y: 0-23}
{Y=3: LL Y von GOTO (X,Y)}
{Y=2: HH Local}
{Y=1: LL Local}
{Y=0: 01 Level}
{SP = vordefiniertes Label fuer Stackpointer:}
{(SP),Y=0 hat niedrigste Adresse, z.B. $AFF9}
{(SP),Y=6 hat hoechste Adresse,   z.B. $AFFF}

PROCEDURE GOTOXY (X,Y: INTEGER);
BEGIN
#A
    STX T       ;X-Register speichern
    LDY #3
    LDA (SP),Y  ;YL: VTAB
    CMP #24
    BCS A1      ;groesser als 23!
    JSR $FB5B   ;TABV
    LDY #5
    LDA (SP),Y  ;XL: HTAB
    CMP #80
    BCS A1      ;groesser als 79!
    STA $057B   ;XL: HTAB 0-79 - 80 Z/Z IIe/c
    STA $0024   ;XL: HTAB 0-23 - 40 Z/Z II+/e/c
A1  LDX T       ;X-Register laden
#
END;            {Ersetzt RTS}


{KEYBOARD (C): Tastaturabfrage}
{Prozedur mit Variablenparameter: KEYBOARD (C);}
{C ist Char-Variable:}
{Vorher: Wenn C = 'C', dann Cursor sichtbar}
{        Wenn C = 'N', dann Cursor unsichtbar}
{Nachher:C = eingegebene Taste}
{Stackaufbau:}
{Y=4: HH Zeiger (!) auf VAR-Parameter-Wert C}
{Y=3: LL Zeiger (!) auf VAR-Parameter-Wert C}
{Y=2: HH Local}
{Y=1: LL Local}
{Y=0: 01 Level}

PROCEDURE KEYBOARD (VAR C: CHAR);
BEGIN
#A
    STX T       ;X-Register speichern
    LDY #3
    LDA (SP),Y  ;C-Zeiger LL in
    STA T+1     ;Zero-Page T+1
    INY
    LDA (SP),Y  ;C-Zeiger HH in
    STA T+2     ;Zero-Page T+2
    LDY #0
    LDA (T+1),Y ;Wert von C laden (LL)
    CMP #$43    ;und mit 'C' = Cursor vergleichen
    BNE A2
    JSR $FD0C   ;RDKEY, wenn = 'C'
    BNE A3      ;stets, weil Bit 7 stets gesetzt
A2  LDA $C000   ;Keyboard direkt, wenn <> 'C'
    BPL A2
    BIT $C010   ;Keyboard Strobe
A3  AND #$7F    ;Bit 7 auf 0 setzen
    LDY #0
    STA (T+1),Y ;Wert von C speichern (LL)
    LDX T       ;X-Register laden
#
END;            {Ersetzt RTS}

{RDKEY: Tastaturabfrage, Cursor sichtbar}
{Funktion, aufrufen mit C := RDKEY;}
{Stackaufbau:}
{Y=3: C  Funktionswert wird gepokt}
{Y=2: HH Local}
{Y=1: LL Local}
{Y=0: 01 Level}

FUNCTION RDKEY: CHAR;
BEGIN
#A
    STX T       ;X-Register speichern
    JSR $FD0C   ;RDKEY: Tastaturabfrage
    AND #$7F    ;Bit 7 auf 0
    LDY #3
    STA (SP),Y  ;in Char-Funktionswert poken
    LDX T       ;X-Register laden
#
END;            {Ersetzt RTS}


{POKE: Poken von Wert W in Speicherstelle S}
{Prozedur, aufrufen mit POKE (S,W);}
{Stackaufbau:}
{Y=6: HH von S}
{Y=5: LL von S}
{Y=4: HH von W ignorierbar}
{Y=3: LL von W}
{Y=2: HH Local}
{Y=1: LL Local}
{Y=0: 01 Level}
```

```
PROCEDURE POKE (S,W: INTEGER);
BEGIN
#A
;   STX T ist hier ueberfluessig
    LDY #5      ;Y = 5
    LDA (SP),Y
    STA T+1     ;LL von S
    INY         ;Y = 6
    LDA (SP),Y
    STA T+2     ;HH von S
    LDY #3      ;Y = 3
    LDA (SP),Y  ;LL von W
    LDY #0
    STA (T+1),Y ;poken
#
END;            {Ersetzt RTS}


{PEEK: Peeken von Wert W aus Speicherstelle S}
{Funktion; aufrufen mit W := PEEK (S);}
{Stackaufbau:}
{Y=6: HH von S}
{Y=5: LL von S}
{Y=4: HH von Funktionswert ignorierbar}
{Y=3: LL von Funktionswert (spaeter W)}
{Y=2: HH Local}
{Y=1: LL Local}
{Y=0: 01 Level}

FUNCTION PEEK (S:INTEGER):INTEGER;
BEGIN
#A
;   STX T ist hier ueberfluessig
    LDY #5      ;Y = 5
    LDA (SP),Y
    STA T+1     ;LL von S
    INY         ;Y = 6
    LDA (SP),Y
    STA T+2     ;HH von S
    LDY #0
    LDA (T+1),Y ;peeken
    LDY #3      ;Y = 3
    STA (SP),Y  ;LL von Funktionswert
    LDA #0
    INY         ;Y = 4
    STA (SP),Y  ;HH von Funktionswert loeschen
#
END;            {Ersetzt RTS}

{---------------Demo-Routinen---------------}

{Darstellung des internen Formats der 8 Bytes}
{umfassenden BCD-Real-Variablen als Demo}
{V = Vorzeichen M/E ( 1 Stelle:  0-3)}
{M = Mantisse       (13 Stellen: 0-9)}
{E = Exponent       ( 2 Stellen: 0-9)}
{Stackpointer      Y3 Y4 Y5 Y6 Y7 Y8 Y9 YA}
{Zahlenformat      VM MM MM MM MM MM MM EE}
{+1.234567890123   01 23 45 67 89 01 23 00}
{-1.234567890123   21 23 45 67 89 01 23 00}
{+0.12345          11 23 45 00 00 00 00 01}
{-0.12345          31 23 45 00 00 00 00 01}
{+1E+05            01 00 00 00 00 00 00 05}
{-1E+05            21 00 00 00 00 00 00 05}
{+1E-05            11 00 00 00 00 00 00 05}
{+1E-05            31 00 00 00 00 00 00 05}

PROCEDURE SHOWREAL (R: REAL);
BEGIN
#A
    STX T       ;X-Register speichern
    LDY #3      ;Level und Local ignorieren
A4  LDA (SP),Y
    JSR $FDDA   ;PRBYTE: Hexzahl ausgeben
    INY
    CPY #$0B    ;insgesamt 8 Bytes anzeigen
    BNE A4
    JSR $FD8E   ;CROUT: Return ausgeben
    LDX T       ;X-Register laden
#
END;            {Ersetzt RTS!}


PROCEDURE SHOWBOOLEAN (B1, B2: BOOLEAN);
BEGIN
{Stackaufbau verkuerzt}
{Y=4: B1}
{Y=3: B2}
```

```
#A
    STX T
    LDY #4
    LDA (SP), Y  ;B1
    JSR $FDDA
    LDA #$AC     ;Komma
    JSR $FDED
    LDY #3
    LDA (SP),Y   ;B2
    JSR $FDDA
    LDX T
#
END;


PROCEDURE SHOWAUFZAEHLUNG (A1, A2: AUFZ);
BEGIN
{Stackaufbau verkuerzt}
{Y=6: HH  A1}
{Y=5: LL  A1}
{Y=4: HH  A2}
{Y=3: LL  A2}

#A
    STX T
    LDY #5
    LDA (SP),Y   ;A1
    JSR $FDDA
    INY
    LDA (SP),Y
    JSR $FDDA
    LDA #$AC    ;Komma
    JSR $FDED
    LDY #3
    LDA (SP),Y  ;A2
    JSR $FDDA
    INY
    LDA (SP),Y
    JSR $FDDA
    LDX T
#
END;

PROCEDURE SHOWSTRING (STR: STRING; C: CHAR);
BEGIN
{Stackaufbau verkuerzt}
{Y=6: STR[3]}
{Y=5: STR[2]}
{Y=4: STR[1]}
{Y=3: C}

#A
    STX T
    LDY #4
A5  LDA (SP),Y    ;STR
    ORA #$80
    JSR $FDED
    INY
    CPY #7
    BNE A5
    LDA #$AC     ;Komma
    JSR $FDED
    LDY #3
    LDA (SP),Y   ;C
    ORA #$80
    JSR $FDED
    LDX T
#
END;

PROCEDURE SHOWMENGE (M: MENGE);
BEGIN
{Stackaufbau verkuerzt}
{Y=5: 3. von 32 Bytes: Elemente 16..23 usw.}
{Y=4: 2. von 32 Bytes: Elemente 8..15}
{Y=3: 1. von 32 Bytes: Elemente 0..7}

#A
    STX T
    LDY #3
A6  LDA (SP),Y ;Mengenbytes
    JSR $FDDA
    INY
    CPY #35
    BNE A6
    LDX T
#
END;
```

```
{---------Hauptprogramm----------}

BEGIN
{HOME-Demo}
HOME;
 WRITELN ('HOME');

{GOTOXY-Demo}
GOTOXY (10,10);
 WRITE ('GOTOXY (10,10)');

{KEYBOARD-Demo}
GOTOXY (10,12);
 WRITE ('KEYBOARD mit Cursor: ');
 C := 'C'; KEYBOARD (C); WRITELN (C);
GOTOXY (10,14);
 WRITE ('KEYBOARD ohne Cursor: ');
 C := 'N'; KEYBOARD (C); WRITELN (C);

{RDKEY-Demo}
GOTOXY (10,16);
 WRITE ('RDKEY: ');
 C := RDKEY; WRITELN (C);

{POKE-PEEK-Demo}
WRITELN; C := RDKEY; HOME; GOTOXY (0,2);
WRITELN ('In Bildschirm "A" poken...');
 S := 1024; W := 193;

 POKE (S, W);
WRITELN ('Aus Bildschirm peeken...');
 W := PEEK (S);
 WRITELN (W, ' = ', CHR(W));
C := RDKEY; HOME;
```

```
{----------Typen-Demos----------}

{SHOWREAL-Demo}
REPEAT
 BEGIN
 WRITELN; WRITE ('Realzahl eingeben (0=Ende): ');
 READLN (R); WRITELN (R:19); SHOWREAL (R)
 END;
UNTIL R = 0;
HOME;

{SHOWBOOLEAN-Demo}
WRITELN; WRITELN ('Boolesche Werte');
B1 := TRUE; B2 := FALSE;
SHOWBOOLEAN (B1, B2);
WRITELN; WRITELN ('TRUE,FALSE');

{SHOWAUFZAEHLUNG-Demo}
WRITELN; WRITELN ('Aufzaehlung');
A1 := EINS; A2 := ZWEI;
SHOWAUFZAEHLUNG (A1, A2);
WRITELN; WRITELN ('EINS,ZWEI');

{SHOWSTRING-Demo}
WRITELN; WRITELN ('String + Char');
STR := 'ABC'; C := 'D';
SHOWSTRING (STR, C);
WRITELN; WRITELN (STR, ',', C);

{SHOWMENGE-Demo}
WRITELN; WRITELN ('Menge [7,3,1]');
M := [7,3,1];
WRITELN ('Intern:'); {%10001010 = $8A}
SHOWMENGE (M); WRITELN;

END.
```

Interaktive Funktionseingabe

von Gerhard Röhner

Zur Untersuchung von Funktionen ist es zweckmäßig, bei einem entsprechenden Programm einen Funktionsterm während des Programmlaufs eingeben zu können. Bei interpretierenden Systemen ist dies mit selbstmodifizierenden Programmen relativ leicht machbar. Im UCSD-Pascal-System, bei dem zunächst das Programm in den P-Code kompiliert werden muß, läßt sich die Eingabe eines Funktionsterms, wenn auch etwas umständlicher, ebenfalls realisieren.
Der Grundgedanke besteht darin, die Funktion in einer UNIT zu definieren, diese dann automatisch zu kompilieren und in ein Funktionsuntersuchungsprogramm einzubinden. Dazu wird eine Exec-Datei benutzt, wie sie vom Pascal-System zur Verfügung gestellt wird.

Die erste Teilaufgabe besteht darin, die Unit zu erstellen, welche den Funktionsterm enthält. Dies geschieht mit dem Programm **FUNKSTART**. Die Dateivariable D kennzeichnet eine Textdatei, welche durch
REWRITE(D,'#4:FUNKUNIT.TEXT')
zum Schreiben eröffnet wird. In diese Textdatei wird die gesamte Unit FUNKTION (s. FUNKUNIT) geschrieben. Der Interface-Teil beinhaltet die Schnittstelle zwischen aufrufendem Programm und der Unit. Die Funktion F erwartet einen Real-Parameter, der Funktionswert ist ebenfalls eine Real-Zahl. Hier sind Erweiterungen auf mehrere Parameter ohne weiteres möglich. Im Implementation-Teil wird die Funktion F definiert. Der Funktionsterm wird über die Tastatur eingegeben und dann in die Unit geschrieben. Nach
CLOSE(D,LOCK)
ist die Unit FUNKTION als Text-Datei verfügbar.
Zum automatischen Kompilieren und Einbinden in ein Funktionsuntersuchungsprogramm wird als nächstes eine Exec-Datei erzeugt. Sie wird als **FUNKEXEC** ebenfalls auf Volume 4: geschrieben. Der Inhalt der Datei ist in **FUNKUNIT** zu sehen.

Als erstes Zeichen erhält die Exec-Datei einen Terminator. Im Beispiel ist es das Prozentzeichen. Das folgende F steht für den Aufruf des Filers. Mit N(ew wird die aktuelle SYSTEM.WRK-Datei gelöscht. Falls diese noch nicht mit dem Filer-Befehl S(ave gespeichert wurde, wird mit Y(es die Frage 'Throw away current workfile?' beantwortet. Der Befehl G(et mit anschließendem FUNKUNIT kennzeichnet diese Datei als neue Arbeitsdatei. Das nächste Zeichen der Exec-Datei ist Q zum Verlassen des Filers. C steht für den Aufruf des Compilers. Nach Erzeugung der Code-Datei von FUNKUNIT wird durch L der Linker aufgerufen. Als sog. Hostfile wird **FUNKDEMO** angegeben. Die nächste Frage des Linkers nach dem Libfile wird mit SYSTEM.WRK beantwortet. Die beiden WRITELN(D)-Anweisungen erzeugen zwei Wagenrücklauf-Zeichen als Reaktion auf die Fragen „Lib file?" und „Map file?" des Linkers. Abschließend wird mit **FUNKFERTIG** der Dateiname des Outputfiles angegeben.
Nachdem der Linker die Unit FUNKTION in das FUNKDEMO-Programm eingebunden hat, liegt mit FUNKFERTIG ein lauffähiges Programm vor. Mit dem Befehl eXecute FUNKFERTIG wird es zur Ausführung gebracht. Nach Beendigung des Programmlaufs befinden sich noch die 4 Terminatoren %%%% in der EXEC-Datei. Sie zeigen an, daß die EXEC-Datei nun vollständig abgearbeitet ist. Das System geht danach auf die Hauptkommandoebene zurück.
Damit sind alle Bestandteile der Exec-Datei erläutert. Gestartet wird sie vom FUNKSTART-Programm, welches dazu die Unit CHAINSTUFF benutzt. Mit
SETCHAIN('EXEC/#4:FUNKEXEC')
wird die Abarbeitung der Exec-Datei aufgerufen.

Um das Programm FUNKDEMO in Betrieb nehmen zu können, läßt man zunächst FUNKSTART laufen, um die Unit und die EXEC-Datei zu erhalten. Nach dessen Ablauf befindet man sich im Filer! Dies rührt daher, daß ursprünglich keine Code-Version des Programms FUNKDEMO vorliegt und der Linker seine Aufgabe nicht wahrnehmen kann. Der erste Programmlauf hat aber bewirkt, daß die UNIT erstellt und kompiliert wurde. Man muß sie nun noch als solche mit dem Befehl Save sichern und kann dann sein eigenes Programm FUNKDEMO erfolgreich kompilieren und sichern. Diese etwas umständliche Prozedur ist nur beim erstenmal nötig, da FUNKDEMO nur bei vorhandener Unit FUNKTION erfolgreich kompiliert werden kann.
Ist all dies geschehen, reicht der Aufruf von FUNKSTART aus. Man gibt seinen Funktionsterm ein und schaut zu, wie alles automatisch erledigt wird.

Kurzhinweise

1. Zweck:
Eingabe von Funktionen im laufenden Pascal-Programm
2. Konfiguration:
Apple II+/e/c, 2 Laufwerke, Apple-Pascal 1.1/1.2
3. Test:
(a) Dateien FUNKSTART.TEXT und FUNKDEMO.TEXT mit GETDOS-Programm (Disk #15) von DOS-Sammeldisk auf Pascal-Diskette konvertieren; (b) FUNKSTART.TEXT zu FUNKSTART.CODE kompilieren; (c) eXecute FUNKSTART; endet zunächst im Filer; danach S(ave) von FUNKUNIT; nun FUNKDEMO kompilieren; (e) später nur noch jeweils eXecute FUNKSTART; dann Formel eingeben.
4. Sammeldisk:
FUNKSTART.TEXT
FUNKDEMO.TEXT
ferner:
FUNKUNIT.TEXT
FUNKEXEC.TEXT

FUNKSTART

```
PROGRAM FUNKSTART;
{von G.Roehner, 15.8.85}

USES CHAINSTUFF;

VAR    D:TEXT;
    TERM:STRING;

BEGIN
REWRITE(D,'#4:FUNKUNIT.TEXT');
WRITELN(D,'{$S+}');
WRITELN(D,'UNIT FUNKTION;');
WRITELN(D,'INTERFACE');
WRITELN(D,' USES TRANSCEND;');
WRITELN(D,' FUNCTION F(X:REAL):REAL;');
WRITELN(D,'IMPLEMENTATION');
WRITELN(D,' FUNCTION F;');
WRITELN(D,' BEGIN');
PAGE(OUTPUT);
WRITELN('Geben Sie die',
        ' Funktionsgleichung ein');
WRITELN;
WRITELN('Beispiel: y=sin(x)+2*x');
WRITELN;
WRITE  ('           y=');READLN(TERM);
WRITELN(D,'      F:=',TERM);
WRITELN(D,'    END;');
WRITELN(D,'END.');
CLOSE(D,LOCK);
REWRITE(D,'#4:FUNKEXEC.TEXT');
WRITELN(D,'%FNYGFUNKUNIT');
WRITELN(D,'QCLFUNKDEMO');
WRITELN(D,'SYSTEM.WRK');
WRITELN(D);
WRITELN(D);
WRITELN(D,'FUNKFERTIG');
WRITELN(D,'XFUNKFERTIG');
WRITELN(D,'%%%');
CLOSE(D,LOCK);
SETCHAIN('EXEC/#4:FUNKEXEC')
END.
```

FUNKUNIT

```
{$S+}
UNIT FUNKTION;
  INTERFACE
    USES TRANSCEND;
    FUNCTION F(X:REAL):REAL;
  IMPLEMENTATION
    FUNCTION F;
    BEGIN
      F:=sin(x)+2*x
    END;
END.
```

FUNKEXEC

```
%FNYGFUNKUNIT
QCLFUNKDEMO
SYSTEM.WRK


FUNKFERTIG
XFUNKFERTIG
%%%
```

FUNKDEMO

```
PROGRAM FUNKDEMO;

USES TRANSCEND,
 {$U #4:FUNKUNIT.CODE} FUNKTION;

VAR X,VON,BIS,DELTA:REAL;

BEGIN
  PAGE(OUTPUT);
  REPEAT
    WRITELN('Funktionsintervall');
    WRITE('von: ');READLN(VON);
    WRITE('bis: ');READLN(BIS);
    WRITELN;
  UNTIL VON < BIS;
  WRITE('Schrittweite: ');
   READLN(DELTA);
  WRITELN;
  X:=VON;
  REPEAT
    WRITELN('x= ',X,'  f(x)= ',F(X));
    X:=X+DELTA
  UNTIL X > BIS
END.
```

Zweistimmige Melodien

Ein intelligentes Tonprogramm für den Apple II

von Jörg Schmidt

Der Apple II besitzt im Gegensatz zu vielen anderen (neueren) Computern keinen Sound-Generator. Statt dessen muß die Membran des eingebauten Lautsprechers durch gezieltes Ansprechen der Speicherstelle $C030 zum Schwingen gebracht werden. Gemessen am heutigen Standard (siehe Mockingboard) eine vorsintflutliche Methode, doch immerhin lassen sich so einfache Töne (z.B. zur Untermalung von Spielen) und primitive Melodien realisieren – vorausgesetzt man benutzt Maschinensprache. Schwieriger ist es dann schon, unter den ärmlichen Hardware-Voraussetzungen zweistimmige Musik erklingen zu lassen – aber es geht!

1. Das Prinzip

1.1. Einstimmige Töne

Adressiert man die Speicherstelle SPKR (Speaker = $C030), so wird der Zustand der Lautsprechermembran umgekehrt: Ist sie in der Ruhelage, so wird sie herausgedrückt (ein „Knack" entsteht). Nochmaliges Adressieren bewirkt, daß die Membran wieder in die Ruhelage zurückkehrt. Zusammenhängende Töne werden durch periodisches Ansprechen von $C030 erzeugt (z.B. innerhalb einer Programmschleife). Integriert man in diese Schleife eine durch einen Zahlenwert einstellbare Wartezeit, so lassen sich verschiedene Frequenzen erzeugen: Ist die Wartezeit (im folgenden als „Frequenz-Wert" bezeichnet) groß, entstehen tiefe Töne; ist sie klein, entstehen hohe Töne. Leider hat dieser Algorithmus einen Nachteil: Bei gleicher Anzahl der Schleifendurchläufe (Dauer des Tones) hält ein tiefer Ton länger an als ein hoher.

Ein anderes Funktionsprinzip ist also notwendig:
Die Dauer des Tones ist von der Anzahl der Schleifendurchläufe abhängig. Nun ist aber die Frequenz der Schleifendurchläufe konstant hoch.

Bei jedem Schleifendurchlauf wird deshalb ein Zähler erniedrigt, der zuvor mit dem Frequenz-Wert initialisiert wurde. Ist der Zähler bei Null angelangt, so wird er wiederum mit dem Frequenz-Wert belegt und $C030 adressiert. Ist der Frequenz-Wert hoch (tiefer Ton), so wird er innerhalb der Schleife seltener auf Null herabgezählt werden können, d.h. ein dunkler Ton entsteht. Ist der Frequenz-Wert klein (hoher Ton), so wird der damit belegte Zähler häufiger Null erreichen; es entsteht eine höhere Frequenz.

1.2. Zweistimmige Töne

Soll der Ton zweistimmig sein, werden in der Schleife zwei Programmteile zum Abzählen und Knacken hintereinander angeordnet, jeweils für Frequenz 1 und Frequenz 2. Nun hört sich das Ergebnis noch sehr verzerrt an, da sich beide Frequenzen gegenseitig beeinflussen. Deshalb ergreift man zwei Maßnahmen:

– Zusätzlich zu Frequenz 1 und Frequenz 2 werden Frequenz 1 und 2 durch 16 berechnet. Ist die Membran herausgedrückt worden, so wird nach einem 16tel der Durchlaufzeit für den nächsten Knack die Membran wieder hineingezogen. Die Schwingungen werden somit kontrolliert abgeschlossen.

– Es wird ein „Membran-Flag" eingerichtet (hier Akkumulator), das ständig darüber Auskunft gibt, ob sich die Membran in der Ruhelage oder im herausgedrückten Zustand befindet. So kann verhindert werden, daß eine Amplitude direkt wieder durch einen weiteren Versuch, die Membran herauszudrücken, unterbunden wird.

2. Handhabung

Die Tonroutine für zwei Stimmen liegt ab $0300 im Speicher und ist somit vor Applesoft-Programmen geschützt. Die Musik-Files, die die Notenwerte für ganze Musikstücke enthalten, können in jeden beliebigen Speicherbereich geladen werden. Die Anfangsadresse des Musikstücks wird im Low/High-Format in den Speicherstellen 6 und 7 abgelegt. Der Aufruf erfolgt mit CALL 768 ($0300). Das Stück wird abgespielt, bis man eine Taste drückt oder es zu Ende ist.

3. Der Musik-Editor

Der Musik-Editor ist ein Applesoft-Hilfsprogramm zum Komponieren von eigenen Musikstücken, um Partituren einzugeben usw. Die Stücke können eingetippt, geladen, abgespeichert, editiert und gespielt werden.

3.1 Befehlssatz

Der Musik-Editor arbeitet nicht menügesteuert, sondern durch Eingabe von „Ein-Buchstabe-Kommandos". Das Programm zeigt als Prompt das Größer-Zeichen („>") und erwartet nun folgende Befehle, auf die gegebenenfalls noch weitere Eingaben folgen können:

– **Ctrl-D**: DOS-Befehle ausführen (CATALOG etc.).
– **Ctrl-L**: Musikstück laden (Filename eingeben).
– **Ctrl-S**: Musikstück abspeichern (Filename eingeben).
– **-Ctrl-A**: Musikstück an das im Speicher befindliche anhängen (Filename eingeben).
– **S**: Musikstück im Speicher abspielen (Start-Zeilennummer angeben).
– **L**: Musikstück listen (Zeilenbereich eingeben). Während das Listing abläuft, kann es mit der Leertaste angehalten und mit dem Schrägstrich („/") abgebrochen werden.
– **A**: Zeilen ans Ende des Musikstücks anfügen.
– **E**: Zeilen editieren: Geben Sie zunächst den Zeilenbereich an, den Sie editieren wollen. Ist dies getan, gibt das Programm die erste Zeile des Bereichs aus und erwartet eine Zeile tiefer die neue Eingabe dieser Zeile. Dies wiederholt sich bis zur letzten Zeile des Bereichs.
– **N**: Neues Musikstück beginnen. Das vorhandene Stück wird gelöscht und die Eingabe beginnt bei Zeile 0.
– **T**: Trace-Modus ein- bzw. ausschalten. Ist Trace aktiv, wird bei der Ein- und Ausgabe jede Zeile, die ausgedruckt wird, „mitgespielt".
– **?**: Gibt in Kurzform Informationen über die einzelnen Befehle aus.
– **Esc**: Beendet das Programm (zieht den Verlust des Musikstückes im Speicher nach sich).

3.2 Ausgabeformat der Musikstücke

Die Stücke sind zeilenweise aufgebaut. Ganz links steht die Zeilennummer, daneben die Dauer (ein Zahlenwert), dann der Notenwert der 1. Stimme und ganz rechts der Notenwert der 2. Stimme.
Der Notenwert selber besteht aus einer Ziffer für die Oktave (1-5), daran anschließend die Note selber (A-G). Ist die Note um einen Halbtonschritt erhöht, so schließt sich noch ein Doppelkreuz („#") an. Soll eine Stimme gar nicht ertönen, werden anstelle ihres Notenwertes drei Striche („---") eingegeben. Eine Pause ergibt sich, wenn bei einem gewünschten Längenwert beide Stimmen stumm sind.
Als Besonderheit ist zu beachten, daß die Note „H" entsprechend der amerikanischen Schreibweise als „B" einzugeben ist. Außerdem beginnt eine Oktave immer bei A und endet mit G (normalerweise C bis H).

3.3 Eingabe von Zeilen

Links steht wiederum die Zeilennummer. Geben Sie nun die Dauer, den Notenwert der 1. Stimme und den Notenwert der 2. Stimme, jeweils durch Kommas getrennt, ein und schließen Sie mit Return ab. Das Format der Dauer- und Notenwerte ist dasselbe wie bei der Ausgabe.
Nach Eingabe einer Zeile dauert es einige Sekunden, bis das Ergebnis auf dem Bildschirm bestätigt wird und das Programm zur Eingabe der nächsten Zeile bereit ist.
Möchten Sie keine weiteren Zeilen eingeben, tippen Sie nur zwei Kommas („,,"), gefolgt von Return. Das Musikstück endet dann an dieser Stelle.

3.4 Awendungsbeispiele

Beim Dauer-Wert haben sich Vielfache von drei als günstig erwiesen. Eine Achtelnote wäre dann z.B. 6, eine Viertelnote 12, eine halbe Note 24 usw. Wer es etwas langsamer liebt, sollte entsprechende Vielfache von 4 wählen.
Hier einige konkrete Beispiele:

„12,3A,--- ": Eine Viertelnote, erste Stimme A in der 3. Oktave und zweite Stimme stumm.
„3,2C#,3F": Eine Sechzehntelnote, erste Stimme Cis in der 2. Oktave und zweite Stimme F in der 3. Oktave.
„24,---,---": Eine Pause von einer halben Note.

Weitere Beispiele können dem **Listing eines Musikstückes** entnommen werden, das die ersten Takte einer Sonate wiedergibt, die sich auf der Sammeldisk befindet.

Kurzhinweise

1.Zweck:
Programm zur Ein- und Wiedergabe von zweistimmigen Musikstücken
2. Konfiguration:
II+ (mit G/K), IIe oder IIc;
DOS 3.3 oder ProDOS (nicht bei MUSIK-.EDITOR);
3. Test:
RUN MUSIK.EDITOR
RUN SONATA (nur auf Sammeldisk)
4. Sammeldisk:
MUSIK.EDITOR
(Applesoft-Musik-Editor)
T.TONROUTINE
(Big-Mac-Quelltext)
TONROUTINE
(Maschinenprogramm zur zweistimmigen Tonerzeugung)
SONATA
(Applesoft-Startprogramm)
M.SONATA
(Editiertes Musikstück als Binär-File)

Listing eines Musikstückes

(die ersten Takte der Sonate auf der Sammeldisk)

```
 0:   16   2C   3C
 1:   16   2G   3C
 2:   16   2E   3C
 3:   16   2G   3C
 4:   16   2C   3E
 5:   16   2G   3E
 6:   16   2E   3G
 7:   16   2G   3G
 8:   16   2D   3B
 9:   16   2G   3B
10:   16   2F   3B
11:    8   2G   3C
12:    8   2G   3D
13:   16   2C   3C
14:   16   2G   3C
15:   16   2E   ---
16:   16   2G   ---
17:   16   2C   4A
18:   16   3A   4A
19:   16   2F   4A
20:   16   3A   4A
21:   16   2C   3G
22:   16   2G   3G
23:   16   2E   4C
24:   16   2G   4C
25:   16   2B   3G
26:   16   2G   3G
27:    4   2D   3G
28:    4   2D   3F
29:    4   2D   3G
30:    4   2D   3F
31:    8   2G   3E
32:    8   2G   3F
33:   16   2C   3E
```

MUSIK.EDITOR

```
1000 REM --------------------
1010 REM     Musik-Editor
1020 REM  von Jörg Schmidt
1030 REM     Februar 1986
1040 REM --------------------
1050 :
1060 TEXT : HOME : SPEED= 255
1070 HIMEM: 16384: CLEAR
1080 D$ = CHR$ (4):MA = 16384
1090 PRINT D$"BLOAD TONROUTINE"
1100 D = 0:F1 = 1:F2 = 2:ML = 6:MH = ML + 1
1110 PL = 768:TN = PL + 40:BL = 43616
1120 DIM FR$(255)
1130 FOR I = 0 TO 255:FR$(I) = "???": NEXT
1140 RESTORE : FOR I = 1 TO 5
1150 I$ = STR$ (I)
1160 FOR J = 65 TO 71:A$ = CHR$ (J)
1170 READ A:FR$(A) = I$ + A$
1180 IF A$ = "B" OR A$ = "E" THEN 1200
1190 READ A:FR$(A) = I$ + A$ + "#"
1200 NEXT J,I
1210 DATA 255,240,228,216,204,192,180,172,160,152,144,136
1220 DATA 128,120,114,108,102,96,90,86,80,76,72,68
1230 DATA 64,60,57,54,51,48,45,43,40,38,36,34
1240 DATA 32,30,28,27,25,24,22,21,20,19,18,17
1250 DATA 16,15,14,13,12,12,11,10,10,9,9,8
1260 FR$(0) = "---"
1270 POKE MA,0:LN = 1
1280 HOME : INVERSE
1290 HTAB 9: PRINT SPC( 22): PRINT
1300 HTAB 9: PRINT "     MUSIK-EDITOR     "
1310 HTAB 9: PRINT SPC( 22): PRINT
1320 HTAB 9: PRINT "  VON: JOERG SCHMIDT  "
1330 HTAB 9: PRINT "     FEBRUAR 1986     "
1340 HTAB 9: PRINT SPC( 22): NORMAL
1350 PRINT : PRINT
1360 PRINT "Befehle:": PRINT
1370 PRINT "<Ctrl-D>...DOS-Operationen"
1380 PRINT "<Ctrl-L>...Musik-File laden"
1390 PRINT "<Ctrl-S>...Musik-File speichern"
1400 PRINT "<Ctrl-A>...Musik-File anhängen"
1410 PRINT "<S>........Musikstück spielen"
1420 PRINT "<L>........Musikstück listen"
1430 PRINT "<A>........Zeile(n) anfügen"
1440 PRINT "<E>........Zeile(n) editieren"
1450 PRINT "<N>........Neues Stück beginnen"
1460 PRINT "<T>........Trace ein/aus"
1470 PRINT "<?>........Diese Informationen"
1480 PRINT "<Esc>......Ende des Programms"
1490 PRINT : PRINT ">";
1500 POKE - 16368,0: GET A$
1510 IF A$ = CHR$ (4) THEN 1640
1520 IF A$ = CHR$ (12) THEN 1700
1530 IF A$ = CHR$ (19) THEN 1760
1540 IF A$ = CHR$ (1) THEN 1810
1550 IF A$ = "S" THEN 1910
1560 IF A$ = "L" THEN 1990
1570 IF A$ = "A" THEN 2070
1580 IF A$ = "E" THEN 2120
1590 IF A$ = "N" THEN 2190
1600 IF A$ = "T" THEN 2220
1610 IF A$ = "?" THEN 1360
1620 IF A$ < > CHR$ (27) THEN PRINT CHR$ (7);: GOTO 1500
1630 HOME : END
1640 REM ---DOS-Operationen---
1650 ONERR GOTO 1870
1660 INPUT "DOS-Befehl:";A$
1670 IF A$ = "" THEN 1490
1680 PRINT D$A$
1690 POKE 216,0: GOTO 1490
1700 REM ---Musik-File laden---
1710 ONERR GOTO 1870
1720 INPUT "Laden: ";F$: IF F$ = "" THEN 1490
1730 PRINT D$"BLOAD"F$",A"MA
1740 LN = ( PEEK (BL) + 256 * PEEK (BL + 1)) / 3
1750 POKE 216,0: GOTO 1490
1760 REM ---Musik-File speichern---
1770 ONERR GOTO 1870
1780 INPUT "Speichern: ";F$: IF F$ = "" THEN 1490
1790 PRINT D$"BSAVE"F$",A"MA",L"LN * 3
1800 POKE 216,0: GOTO 1490
1810 REM ---File anhängen---
1820 ONERR GOTO 1870
1830 INPUT "Anhängen: ";F$: IF F$ = "" THEN 1490
1840 PRINT D$"BLOAD"F$",A"MA + LN * 3 - 3
1850 LN = LN + ( PEEK (BL) + 256 * PEEK (BL + 1)) / 3 - 1
1860 POKE 216,0: GOTO 1490
1870 REM ---DOS-Fehler---
1880 POKE 216,0: PRINT
1890 PRINT "DOS-Fehler Nr. " PEEK (222) CHR$ (7)
1900 GOTO 1490
1910 REM ---Stück spielen---
1920 INPUT "Spielen ab Zeile: ";ZN
1930 IF ZN > = LN - 1 THEN 1490
1940 N = MA + ZN * 3: GOSUB 2260
1950 POKE ML,L: POKE MH,H: CALL PL
1960 PRINT "Stop in Zeile ";
1970 PRINT (( PEEK (ML) + 256 * PEEK (MH)) - MA) / 3
1980 GOTO 1490
1990 REM ---Stück listen---
2000 INPUT "Listen (Start,Ende): ";S,E
2010 IF S > E OR S > = LN - 1 THEN 1490
2020 IF E > = LN - 1 THEN E = LN - 2
2030 FOR ZN = S TO E: GOSUB 2290
2040 IF PEEK ( - 16384) = 175 THEN 1490
2050 IF (T) OR PEEK ( - 16384) = 160 THEN POKE - 16368,0:
     WAIT - 16384,128
2060 NEXT : GOTO 1490
2070 REM ---Zeilen anfügen---
2080 ZN = LN - 1: PRINT "Zeile(n) anfügen:"
2090 GOSUB 2380
2100 IF NOT PEEK (AD) THEN 1490
2110 LN = LN + 1:ZN = ZN + 1: GOTO 2090
2120 REM ---Zeilen editieren---
2130 INPUT "Editieren (Start,Ende): ";S,E
2140 IF S > E OR S > = LN - 1 THEN 1490
2150 IF E > = LN - 1 THEN E = LN - 2
2160 FOR ZN = S TO E: GOSUB 2290
2170 GOSUB 2380: IF NOT PEEK (AD) THEN LN = ZN + 1: GOTO 1490
2180 NEXT : GOTO 1490
2190 REM ---Neues Stück---
2200 PRINT "Neues Musikstück beginnen:"
2210 POKE MA,0:ZN = 0:LN = 1: GOTO 2090
2220 REM ---Trace ein/aus---
2230 T = 1 - T: PRINT "Trace ";
2240 IF T THEN PRINT "ein": GOTO 1490
2250 PRINT "aus": GOTO 1490
2260 REM ---Lo/Hi-Teilung---
2270 H = INT (N / 256):L = N - 256 * H
2280 RETURN
2290 REM ---Zeile ausgeben---
2300 AD = MA + 3 * ZN
2310 PRINT TAB( 5 - LEN ( STR$ (ZN)))ZN":";
2320 PRINT TAB( 10) PEEK (AD);
2330 PRINT TAB( 20)FR$( PEEK (AD + 1));
```

```
2340 PRINT TAB( 32)FR$( PEEK (AD + 2))
2350 IF NOT T THEN RETURN
2360 POKE D, PEEK (AD): POKE F1, PEEK (AD + 1):
     POKE F2, PEEK (AD + 2)
2370 CALL TN: RETURN
2380 REM ---Zeile eingeben---
2390 PRINT TAB( 5 - LEN ( STR$ (ZN)))ZN":";
2400 INPUT "";Z$,F1$,F2$:AD = MA + ZN * 3
2410 POKE AD, VAL (Z$): IF NOT VAL (Z$) THEN RETURN
2420 FOR I = 0 TO 255: IF F1$ = FR$(I) THEN W1 = I
2430 IF F2$ = FR$(I) THEN W2 = I
2440 NEXT : POKE AD + 1,W1: POKE AD + 2,W2
2450 VTAB PEEK (37): GOTO 2290
```

SONATA

(Beispiel für den Aufruf eines editierten Musikstückes. M.SONATA befindet sich nur auf der Sammeldisk)

```
10 HOME : PRINT "Es wird gespielt ..."
20 PRINT CHR$ (4)"BLOAD TONROUTINE"
30 PRINT CHR$ (4)"BLOAD M.SONATA,A$2000"
40 POKE 6,0: POKE 7,32
50 PRINT : PRINT "... SONATA": CALL 768
60 GET A$: REM löscht Tastatur
```

TONROUTINE

BSAVE TONROUTINE, A$0300, $00A6

```
                    1  ********************************
                    2  * Routine für zweistimmige Töne *
                    3  * ============================ *
                    4  * von Jörg Schmidt, Februar 86  *
                    5  ********************************
                    6
                    7           ORG  $300
                    8
                    9  * Aufruf:
                   10  * In MUSL/MUSH die Startadresse des
                   11  * Musik-Files ablegen (POKE 6,Low
                   12  * POKE 7,High). Dann CALL 768
                   13  * zum Abspielen
                   14
                   15  * Achtung, keine NOPs weglassen!
                   16
                   17  DAUER    EQU  $00        ;Einzelton-Dauer
                   18  FREQU1   EQU  $01        ;Frequenz 1
                   19  FREQU2   EQU  $02        ;Frequenz 2
                   20  CTR      EQU  $03        ;Zähler
                   21  FR1D16   EQU  $04        ;Frequenz 1 /16
                   22  FR2D16   EQU  $05        ;Frequenz 2 /16
                   23  MUSL     EQU  $06        ;Musikstück Low
                   24  MUSH     EQU  $07        ;High
                   25
                   26  KBD      EQU  $C000      ;Tastatur
                   27  SPKR     EQU  $C030      ;Lautsprecher
                   28
                   29  * Musikstück abspielen:
                   30
0300: A0 00        31  PLAY     LDY  #0
0302: B1 06        32           LDA  (MUSL),Y   ;Dauer
0304: F0 21        33           BEQ  PLAYEND    ;0 = Ende
0306: 85 00        34           STA  DAUER
0308: C8           35           INY
0309: B1 06        36           LDA  (MUSL),Y   ;Frequenz 1
030B: 85 01        37           STA  FREQU1
030D: C8           38           INY
030E: B1 06        39           LDA  (MUSL),Y   ;Frequenz 2
0310: 85 02        40           STA  FREQU2
0312: 20 28 03     41           JSR  TON        ;Ton spielen
0315: AD 00 C0     42           LDA  KBD        ;Taste gedrückt
0318: 30 0D        43           BMI  PLAYEND    ;-> Ende
031A: A5 06        44           LDA  MUSL       ;nächste Noten
031C: 18           45           CLC
031D: 69 03        46           ADC  #3
031F: 85 06        47           STA  MUSL
0321: 90 DD        48           BCC  PLAY
0323: E6 07        49           INC  MUSH
0325: B0 D9        50           BCS  PLAY
0327: 60           51  PLAYEND  RTS             ;Ende
                   52
                   53  * Zweistimmigen Einzelton spielen:
                   54  * (Ist ein Frequenzwert 0, wird
                   55  * die jew. Stimme nicht gespielt)
                   56
```

```
0328: A5 01        57  TON      LDA  FREQU1     ;Frequenz 1
032A: 4A           58           LSR             ;durch 16
032B: 4A           59           LSR             ;berechnen
032C: 4A           60           LSR
032D: 4A           61           LSR
032E: 85 04        62           STA  FR1D16
                   63
0330: A5 02        64           LDA  FREQU2     ;Frequenz 2
0332: 4A           65           LSR             ;durch 16
0333: 4A           66           LSR             ;berechnen
0334: 4A           67           LSR
0335: 4A           68           LSR
0336: 85 05        69           STA  FR2D16
                   70
0338: A9 00        71           LDA  #0         ;Zähler
033A: 85 03        72           STA  CTR        ;initialisieren
033C: A2 10        73           LDX  #$10
033E: A0 F0        74           LDY  #$F0
                   75
                   76  * --- 1. Stimme ---
                   77
0340: CA           78  TONLOOP  DEX
0341: D0 10        79           BNE  TON1
0343: A6 01        80           LDX  FREQU1
0345: F0 1F        81           BEQ  TON8
0347: C9 01        82           CMP  #1         ;Membran in
0349: F0 1B        83           BEQ  TON8       ;Ruhelage?
034B: 2C 30 C0     84           BIT  SPKR       ;Membran raus
034E: 49 01        85           EOR  #1         ;Membran-Flag
0350: 4C 6A 03     86           JMP  TON3       ;umkehren
                   87
0353: E4 04        88  TON1     CPX  FR1D16     ;1/16 später?
0355: D0 0C        89           BNE  TON5
0357: C9 00        90           CMP  #0         ;Membran
0359: F0 0B        91           BEQ  TON8       ;rausgedrückt?
035B: 2C 30 C0     92           BIT  SPKR       ;Membran rein
035E: 49 01        93           EOR  #1         ;Membran-Flag
0360: 4C 6A 03     94           JMP  TON3       ;umkehren
                   95
0363: EA           96  TON5     NOP             ;Timing-NOPs
0364: EA           97           NOP
0365: EA           98           NOP
0366: EA           99  TON8     NOP
0367: EA          100           NOP
0368: EA          101           NOP
0369: EA          102           NOP
036A: EA          103  TON3     NOP
036B: EA          104           NOP
                  105
                  106  * --- 2. Stimme ---
                  107
036C: 88          108           DEY
036D: D0 10       109           BNE  TON2
036F: A4 02       110           LDY  FREQU2
0371: F0 1F       111           BEQ  TON9
0373: C9 01       112           CMP  #1         ;Membran in
0375: F0 1B       113           BEQ  TON9       ;Ruhelage?
0377: 2C 30 C0    114           BIT  SPKR       ;Membran raus
037A: 49 01       115           EOR  #1         ;Membran-Flag
037C: 4C 96 03    116           JMP  TON4       ;umkehren
                  117
037F: C4 05       118  TON2     CPY  FR2D16     ;1/16 später?
0381: D0 0C       119           BNE  TON6
0383: C9 00       120           CMP  #0         ;Membran
0385: F0 0B       121           BEQ  TON9       ;rausgedrückt?
0387: 2C 30 C0    122           BIT  SPKR       ;Membran rein
038A: 49 01       123           EOR  #1         ;Membran-Flag
038C: 4C 96 03    124           JMP  TON4       ;umkehren
                  125
038F: EA          126  TON6     NOP             ;Timing-NOPs
0390: EA          127           NOP
0391: EA          128           NOP
0392: EA          129  TON9     NOP
0393: EA          130           NOP
0394: EA          131           NOP
0395: EA          132           NOP
0396: EA          133  TON4     NOP
0397: EA          134           NOP
                  135
0398: C6 03       136           DEC  CTR
039A: D0 05       137           BNE  TON10
039C: C6 00       138           DEC  DAUER
039E: D0 A0       139           BNE  TONLOOP
03A0: 60          140           RTS             ;Ende
                  141
03A1: EA          142  TON10    NOP             ;Timing-NOPs
03A2: EA          143           NOP
03A3: 4C 40 03    144           JMP  TONLOOP

166 Bytes
```

Das Modembuch zur DFÜ
Datenfern-Übertragung für Anfänger und Profis
von Manfred und Bruno Hurth
2. Auflage 1985, 314 S., kart., DM 29,80
Selbstverlag Bruno Hurth, Essen
Gliederung
Derzeitige Möglichkeiten der DFÜ im Bereich der DBP (Übersicht) – Datenübertragung im öffentlichen Fernsprechnetz – BTX – Mailboxen – DATEX-L – DATEX-P – Öffentliches Direktrufnetz – Internationale Mietleitungen – Anhang – Lexikon
Bemerkungen
Sehr nützliches Vademekum für alle Modem-Fans, insbesondere wegen des umfangreichen Informationsanbieterverzeichnisses.

Basic: Mathematik per Computer
Eine Software-Sammlung in Basic
von Rudolf Busch
1984, 108 S., 32 Abb., kart., DM 19,80
Franzis-Verlag, München
Gliederung
Auftakt – Das Potenzieren – Das Wurzelziehen – Die Reihenfolge von Rechenoperationen – Ein erstes Computerprogramm – Das Rechnen mit Exponenten – Neue Typen von Variablen – Das Vor-Definieren von Variablen-Typen – Die mathematischen Funktionen Ihres Computers – Die Ganzzahl-Funktion – Die „Verwandten" von INT – Der Zufallsgenerator Ihres Computers – Winkelfunktionen – Logarithmus und Exponent – Zusammenfassung – Liste der Operatoren – Übungsaufgaben
Bemerkungen
Gute und unkomplizierte Einführung in die Programmierung mathematischer Aufgaben. Für den Schulgebrauch geeignet. Leider wird das heute veraltete TRS-80-BASIC verwendet.

Basic: Zahlen-Umwandlungen
Eine Software-Sammlung in Basic
Von der Dezimal-Binär-Wandlung zur Spezial-Rechenmaschine
von Rudolf Busch
1984, 67 S., 9 Abb., kart., DM 16,80
Franzis-Verlag, München
Gliederung
Zahlensysteme – Zahlenumwandlungen – Universalprogramme – Sonderprobleme – Anwendungen – Anhang
Bemerkungen
Leichtverständliche Einführung in die Zahlensysteme mit Beispielprogrammen in TRS-80-BASIC. Applesoft-Programmierer müssen deshalb die Programme umschreiben.

Basic: Alles über Peek und Poke
Eine Software-Sammlung in Basic
von Heiko Requardt
1984, 70 S., 9 Abb., kart., DM 19,80
Franzis-Verlag, München
Gliederung
Der Peek-Befehl und seine Anwendung – Die Tastatur und der Tastaturspeicher – Der Poke-Befehl und seine Anwendung – Empfehlungen und Ausblick
Bemerkungen
Ein allgemeines Buch über Peeks und Pokes ist völlig nutzlos, da diese Befehle genaue Kenntnisse der Speicherorganisation des jeweiligen Rechners voraussetzen. Vom Kauf muß daher abgeraten werden.

Programmierung des 68000
von C. Vieillefond
1985, 453 S., kart. DM 64,-
Sybex-Verlag, Düsseldorf
Gliederung
Grundlagen – Allgemeine Organisation von Datenübertragungen – Ausnahmezustände – Die Speicherverwaltung – Der Befehlssatz des 68000 – Anwenderprogramme – Die anderen Prozessoren der 68000-Familie
Bemerkungen
Wie wir dies inzwischen vom Sybex-Verlag gewöhnt sind, ist auch dieses Buch typographisch aufwendig und damit ansprechend produziert worden. Leider ist auch hier wie bei dem „Basic Handbuch" der Titel nicht ganz zutreffend gewählt, denn das 68000-Buch stellt mehr eine Beschreibung der Befehle denn eine Einführung für Programmierer dar. Man muß schon lange blättern, bis man ab Seite 365 einige wenige Miniprogramme vorfindet.

Basic: Die perfekte Behandlung von Zeichenketten
Eine Software-Sammlung in Basic
von Rudolf Busch
1984, 90 S., 13 Abb., kart., DM 19,80
Franzis-Verlag, München
Gliederung
Was ist eine Zeichenkette? – Die Deklaration von Zeichen(ketten) – Wie lang darf ein String sein? – Die Bedeutung von CLEAR – Die Bedeutung von DIM – Die Konkatenation von Strings – Der ASCII-Code – String-Funktionen – Der Graphikzeichensatz Ihres Computers – Control-Codes und Steuerzeichen – Exotisches – Übungsprogramme
Bemerkungen
Enthält TRS-80-Programme zur String-Verarbeitung, die Apple-MBASIC-Programmierer zum großen Teil übernehmen können.

ISIS Personal Computer Report
Unternehmen, Produkte, Programme
Nomina-Verlag, München
Bemerkungen
Dieses regelmäßig erscheinende Nachschlagewerk – zur Besprechung lag die Ausgabe von Juli-Dezember 1984 vor – bietet auf über 1000 Seiten im DIN-A4-Format detaillierte Informationen über EDV-Firmen und deren Produkte im PC-Bereich. Die Firma Apple wird beispielsweise mit zwei vollen Seiten gewürdigt.

Basic: Programme für Kaufleute
Eine Software-Sammlung in Basic
von Rudolf Busch
1985, 109 S., kart., DM 19,80
Franzis-Verlag
Gliederung
Soll und Haben – Sie werden Unternehmer – Investieren und Abschreiben – Sie betreiben Datenverarbeitung – Sie analysieren Ihre Aktivitäten – Die Dokumentation
Bemerkungen
Ein nützliches Buch für kaufmännisch interessierte MBASIC-Programmierer, die hier zu vielen betriebswirtschaftlichen Anwendungen wie Zinseszinsrechnung, Break-even-Analyse usw. leichtverständliche Programme vorfinden werden.

Softwareführer 1986
für Personal-Computer
1985, 736 S., kart., DM 28,-
Rossipaul-Verlag, München
Bemerkungen
Ein sehr umfangreiches Nachschlagewerk, insbesondere für Anwender-Software. Man beachte jedoch, daß die Programm-Charakteristiken von den Software-Häusern selbst erstellt werden. Es handelt sich also nicht um objektive Darstellungen, geschweige denn um Testberichte.

Grundkenntnisse Pascal
Zeichen, Erklärungen, Beispiele
von Wolfgang J. Weber und Michael Mrowka
1984, 78 S., kart., DM 5,95
Giradet-Verlag, Essen
Bemerkungen
Dieses interessante Büchlein im Format einer Zigarettenschachtel (King Size!) paßt in das Jackett eines jeden Pascal-Programmierers. Eine wirklich gute und preiswerte Idee, in dieser kompakten Form alle Befehle von Standard- und UCSD-Pascal verfügbar zu machen.

Apple II Basic Handbuch
von D. Hergert
1984, 299 S., kart., DM 32,-
Sybex-Verlag, Düsseldorf
Bemerkungen
Der Titel ist unglücklich gewählt, denn es handelt sich nicht um ein (systematisches) Handbuch, sondern um ein (alphabetisches) Lexikon. Wer nähere Informationen zu einem bestimmten Applesoft-Befehl sucht, wird in diesem sauber gesetzten Nachschlagewerk rasch fündig werden. Mitübersetzer dieses amerikanischen Buches ist übrigens der ehemalige Vorsitzende W. Dederichs des deutschen Apple-Clubs AUGE.

bücher

DOS 3.3 – das Diskettenbetriebssystem des Apple-II
von Bernd Ruhland
1985, 256 S., geb., DM 48,-
Franzis-Verlag, München
Wenn Sie jemals Fragen zum inneren Ablauf von DOS 3.3 haben, hier ist das Buch, um Ihnen auf die Sprünge zu helfen. In seiner ganzen Länge disassembliert und kommentiert liegt das Diskettenbetriebssystem vor Ihnen. Sogar das Boot-Programm im EPROM der Controller-Karte wurde bearbeitet. Bis in das letzte Byte auseinandergenommen, in Tabellen verschiedenster Art (Einsprungadressen, Ablaufpläne usw.) gefaßt und mit kleinen erläuternden Texten versehen, läßt sich DOS 3.3 hinter seine Karten schauen. Ruhland geht weiter als Worth/Lechner in „Beneath Apple DOS" oder alle anderen Veröffentlichungen. Aus einer Hochschularbeit heraus ist hier ein Buch entstanden, das allen eine Hilfe bietet, die Änderungen am DOS machen, DOS-Routinen für eigene (Maschinen-)Programme nutzen oder einfach nur verstehen wollen, was dort abläuft. Um vollen Nutzen daraus ziehen zu können, sollten Sie Assemblerkenntnisse besitzen. „Nur-BASIC"-Programmierer werden die Informationen nicht entschlüsseln können.
Vorschläge, welche Teile des DOS Sie wie verändern können, macht Ruhland nicht. Er beschränkt sich auf die minutiöse Beschreibung des Vorhandenen. Hier ist Ihre eigene Phantasie gefordert. DOS 3.3 ist ein Arbeitsbuch und keine Gutenacht-Lektüre.

Betriebssystem CP/M
von Jürgen Plate
1984, 351 S., geb., DM 56.-
Franzis-Verlag, München
CP/M gilt immernoch als eines der weitestverbreiteten Betriebssysteme für Microcomputer. Das Buch wendet sich an alle, die sich ausführlich über das Betriebssystem CP/M der Version 2.2, welches auch für die Apple-Computer verfügbar ist, informieren wollen.
Zunächst werden alle Kommandos der Version 2.2, sowie ihre genaue Syntax beschrieben, und durch Beispiele ergänzt. Weiterhin wird eine Beschreibung der Syntax und des Befehlsvorates der auf der CP/M-System-Diskette vorhandenen Dienstprogramme ED (Editor), ASM (8080-Assembler) und DDT (Dynamic Debugging Tool) gegeben. Bis zu diesem Punkt unterscheidet sich das Buch auch nicht wesentlich von einer Vielzahl auf dem Büchermarkt vorhandener Bücher über das Betriebssystem CP/M.
Wer sich jedoch über die interne Struktur und den Aufbau des CP/M-Betriebssystems informieren möchte, wird nur wenige geeignete Darstellungen finden.
Jürgen Plate ist hier eine ausführliche und übersichtliche Darstellung der internen Struktur des CP/M-Betriebssystems gelungen.
Die Funktion des BDOS, des BIOS, sowie der allgemeine Aufbau des Diskettenbetriebssystems werden ausführlich erklärt und durch Beispiele ergänzt, so daß der Leser in der Lage ist, auf elementarer Ebene zu programmieren. Das Buch enthält ebenfalls ein komplettes Monitor-Programm für Z80-Computer, das eine Programmierung auf Maschinenebene ermöglicht, wenn noch kein CP/M geladen wurde, sowie Beispielprogramme für Urlader, GETSYS, PUTSYS und SYSGEN.
Die Beispielprogramme beziehen sich oft auf den MC-CP/M-Computer und lassen sich deshalb nicht immer direkt auf das Apple-CP/M-System übertragen, liefern aber dennoch wertvolle Anregungen etwa zum Anpassen des CP/M-Systems an eigene Bedürfnisse. Die Assembler-Listings sind größtenteils mit dem Assembler Macro-80 oder mit dem CP/M-Assembler erstellt. Unverständlich ist es, daß verwendete Labels oftmals nur in den Befehlen des Quellcodes enthalten sind, aber sonst nirgendwo im Listing auftauchen, was nicht gerade zum Verständnis des abgedruckten Programmes und dessen einfache Implementierung auf einem anderen Rechner beiträgt. Ein Beispiel:

```
F6E8 18 F4 JR LOOP
```

Das verwendete Label LOOP taucht sonst nirgends im Quelltext auf, so daß Ihnen nichts anderes übrig bleibt, als anhand des Bytes F4 das Sprungziel selbst auszurechnen, um die Position des Labels im Quelltext zu bestimmen, falls Sie sich die Mühe machen wollen, diesen mit Ihrem eigenen Assembler zu verarbeiten.

Am Ende des Buches findet sich noch eine Kurzbeschreibung des Multitasking-/Multiuser-Betriebssystems MP/M, wo auch Begriffe wie Warteschlangen, Bankswitching und virtueller Speicher erläutert werden. Der Anhang des Buches enthält eine Kommandokurzübersicht über alle Directiven, die unter CP/M 2.2 zur Verfügung stehen, sowie eine Liste der CP/M-Fehlermeldungen.
Das Buch ist alles in allem didaktisch gut aufgebaut und geht sehr ins Detail, so daß man es fast als CP/M-Handbuch bezeichnen könnte. Empfehlenswert für alle, die sich mit CP/M beschäftigen, sei es als Anwender oder als aktiver Systemprogrammierer.

Anwenderprogramme
für Apple IIc und Apple IIe
Grafikprogramme, Dateiverwaltungen, Datenstrukturen, Geschäftsprogramme
von E. Floegel
1984, 138 S., kart., DM 19,80
Hofacker-Verlag, Holzkirchen
Gliederung
Aufteilen von Zeichenketten – Verkettete Listen – Hashing – Registrierkasse – Notizbuch – Stichwort-Kartei – AVL-Bäume – Zeichnen eines Balkendiagramms – Zeichnen eines Tortendiagramms – Darstellung ebener Funktionen – Darstellung räumlicher Funktionen – Computer-Grafiken – Artikelverwaltung – ADREKART
Bemerkungen
Das Buch enthält eine vermischte Sammlung von Programmen für kaufmännische Anwendungen. Den Titelzusatz „für IIc und IIe" hätte man weglassen können, denn die Programme laufen auch auf dem alten Apple II+ bzw. auf Nachbauten.

Ein Apple macht sich fein

Staubschutz-Abdeckungen

getestet von Thomas Bühner

Wenn wichtiger Besuch kommt, wirft man sich in Schale. Damit auch der Computer dazu paßt, gibt es für Apple IIe und Macintosh jetzt einen Anzug mit Fliege für festliche Gelegenheiten.

Ausführungen

Diese ausgefallene Bekleidung ist Teil eines kompletten Textil-Programms, das dafür sorgen soll, daß die gesamte Mikrocomputer-Anlage staubfrei bleibt. Ob Laufwerk, Drucker oder der Computer selbst – für alle Komponenten steht eine passende Hülle zur Verfügung. Auch an sparsame Gemüter wurde gedacht: Wer nur die Tastatur vor Staub schützen will, kann das tun und muß entsprechend weniger tief in die Tasche greifen. Insgesamt sind augenblicklich über dreißig verschieden geschnittene Abdeckungen erhältlich.

Design

Außer dem eleganten Anzug steht für weniger formelle Anlässe ein zurückhaltenderes Design zur Verfügung. Der Stoff ist hier uni in den Farben Beige, Blau oder Rot. Von vorn nach hinten verlaufen in der Mitte der Abdeckung drei breite Streifen in unterschiedlichen Tönen der Hauptfarbe.

Verarbeitung

Für alle Teile wurde Baumwolle verwendet, so daß keine Probleme mit statischer Elektrizität auftreten. Mit Ausnahme des Anzugs mit Fliege, der innen gefüttert ist, besteht jede Schutzhülle aus einer Lage Baumwolle. Der Stoff ist sauber verarbeitet worden; am Rand ist er jedoch nur abgesteppt. Das verhindert ein Ausfransen zwar auch zuverlässig, sieht aber nicht so schön aus wie ein umgeschlagener Saum. Beim Waschen (Temperatur 30 Grad) traten bei dem Uni-Design keine Schwierigkeiten auf; für den Anzug mit Fliege empfiehlt sich aber die chemische Reinigung, da hier die Gefahr besteht, daß die schwarzen auf die weißen Teile abfärben.

Wenn der Computer regelmäßig nach Gebrauch abgedeckt wird, kann man ihn auch vor Kaffee und Cola schützen, indem die Schutzhülle mit einem handelsüblichen Spray imprägniert wird.

Ein-Blick

Produkt:	Staubschutz-Abdeckungen für Apple IIe, Macintosh und Peripherie
Gesamtwertung:	****
Optischer Eindruck:	****
Stoffqualität:	****
Verarbeitung:	****
Preis/Leistung-Verhältnis:	***
Preis: (Tastatur-Schutz)	$ 10,-
(Anzug mit Fliege)	$ 48,-
Hersteller:	Soft Wear Company, Ganges, Canada

(Die beste Bewertung entspricht 5 Sternen)

Bildschirm-Dump auf Tastendruck

Fingerprint Plus

getestet von Thomas Bühner

Mittlerweile sind der Imagewriter-Drucker und das Super-Serial-Card-Interface fast zum Standard der Apple-Welt geworden. Hersteller anderer Interfaces können sich nur durch Dumping-Preise oder zusätzliche Extra-Funktionen Marktanteile sichern. Mit „Fingerprint Plus" liegt ein Interface vor, das den zweiten Weg beschreitet.

Funktion

Fingerprint Plus ist ein Interface mit eigenem RAM- und ROM-Speicher, das vor allem für die Zusammenarbeit mit Matrix-Druckern entwickelt wurde. Es besitzt einen seriellen und einen parallelen Ausgang. Die serielle Schnittstelle ist auch zum Anschluß von Modems, Grafik-Tabletts und ähnlichen Geräten geeignet. Beide Ausgänge können gleichzeitig oder unabhängig voneinander benutzt werden. Anschließbar sind Imagewriter, C.Itoh, Epson und etwa zwanzig weitere Drucker, darunter auch Farbdrucker. Laufende Programme kann man jederzeit unterbrechen, um den Inhalt des Bildschirms zu Papier zu bringen.

Lieferumfang

Im Preis inbegriffen sind: das Interface selbst, ein druckempfindliches Sensorplättchen mit Anschlußkabel, ein Kabel für den Drucker, eine Diskette mit Testprogrammen und ein Handbuch. Da für den Test eine knappe Vorab-Version des Handbuchs zur Verfügung stand, kann es nur mit Vorbehalt beurteilt werden.

Installation

Als erstes muß man auf dem Interface kleine DIP-Schalter einstellen, die der Karte anzeigen, welcher Computer (Apple II+ oder IIe) und welcher Drucker angeschlossen werden sollen. Die Anleitung des Handbuchs dafür ist einfach zu befolgen. Anschließend wird das Kabel für den Drucker angebracht.
Bevor das Interface schließlich in einen der Steckplätze geschoben wird, muß man noch das flache, hauchdünne, aber stabile Kabel für das 25 x 25 mm große Sensorplättchen anbringen. Zum Schluß wird eine Folie von der Unterseite des Plättchens entfernt, damit es gut erreichbar in der Nähe der Tastatur auf das Apple-Gehäuse geklebt werden kann. Das flache Kabel hindert nicht beim Schließen der Gehäusehaube.
Wenn die Installation abgeschlossen ist, überprüft man mit Hilfe der Programme auf der Testdiskette, ob der ROM- und RAM-Speicher des Interfaces in Ordnung sind und ob die Einstellung der DIP-Schalter stimmt.

Kompatibilität

Soll Fingerprint Plus mit einem handelsüblichen Programm zusammenarbeiten, so hat man meist Erfolg, wenn bei der Auswahl im Programm angegeben wird, es werde das Grappler+ oder ein anderes Interface der Firma Orange Micro verwendet. Im allgemeinen klappt es bei Software mit großer Interface-Auswahl – wie etwa „The Print Shop" – immer, wenn man alle angegebenen Typen durchprobiert. Mouse Paint arbeitet durch Fingerprint Plus mit fast allen gängigen Matrixdruckern, nicht nur – wie vom Hersteller Apple vorgesehen – mit dem Imagewriter.

Sensorplättchen

Eine Besonderheit dieses Interfaces ist es, daß jedes laufende Programm mit einem Druck auf das Sensorplättchen unterbrochen werden kann. Es erscheint dann ein Bildschirm-Menü mit den zur Verfügung stehenden Optionen. Jetzt kann der Drucker in Aktion gesetzt werden. Ist alles Nötige getan, so tippt man auf die ESC-Taste, und das Programm fährt an der gleichen Stelle fort, an der es unterbrochen wurde. Nach beliebig vielen Unterbrechungen konnte nie festgestellt werden, daß ein Funktionsfehler des laufenden Programms auftrat. Es kam lediglich hin und wieder zu Störungen, wenn man in einem Augenblick unterbrach, in dem das Diskettenlaufwerk in Betrieb war.

Bildschirm-Menü

Nach einem Fingerdruck auf das Sensorblättchen erscheint ein Bildschirm-Menü. Betätigt man jetzt nur die Return-Taste, so wird das Bild, das vor der Unterbrechung des Programms zu sehen war, ausgedruckt. Ob es sich dabei um Text mit 40 oder 80 Zeichen Breite, um niedrig- oder hochauflösende Grafik, um volle Grafik oder solche mit 4 Zeilen Text handelte, ist unerheblich. Was im Augenblick

der Aktivierung des Interface zu sehen war, wird gedruckt. Danach fährt man mit dem Programm durch ein Tippen auf ESC fort.
Statt sofort etwas auszudrucken, kann man sich auch zunächst ansehen, was auf den beim Apple zur Verfügung stehenden Schirmen zu sehen ist. Das sind: 2 x Text (TEXT1 und TEXT2), 1 x niedrigauflösende Grafik ohne und mit vier Zeilen Text (LORES1) und 2 x hochauflösende Grafik ohne und mit vier Zeilen Text (HIRES1 und HIRES2). Der niedrigauflösende Grafikschirm LORES2 ist vom Menü aus nicht sichtbar zu machen. Mit Hilfe dieser Option läßt sich z.B. feststellen, ob ein Hires-Programm beide Grafikschirme benutzt oder nur einen.

Will man einen der angezeigten Schirme auf Papier festhalten, so kann man sofort den Drucker starten. Wohlgemerkt, es ist gleichgültig, ob dieser Schirm im Augenblick des Programm-Stopps tatsächlich zu sehen war.

Statt die Schirme anzusehen oder sie zu drucken, kann man einen beliebigen Text an den Drucker oder das Interface schicken. Dieser Text kann Steuercodes für den Drucker oder das Interface enthalten.

Für außergewöhnliche Anwendungen wird auch die Möglichkeit geboten, mit dem gestoppten Programm nicht mehr fortzufahren, sondern statt dessen in das Apple-Monitor-Programm oder in das Fingerprint-Plus-RAM zu springen. So kann man auch Programme inspizieren, bei denen das normalerweise nicht möglich ist oder eine grafische Darstellung eines solchen Programms auf Kassette abspeichern. (Was übrigens auch für Computer-Neulinge nicht schwierig ist.) Die Abspeicherung auf Diskette ist ohne zusätzliche Insider-Tricks nicht möglich, da das Disketten-Betriebssystem nach der Ankunft im Monitor nicht mehr arbeitet.

Als weitere Option bietet sich an, die Druck-Parameter zu verändern, die auf dem Bildschirm-Menü angezeigt werden. Das geschieht durch die abwechselnde Betätigung der Pfeiltasten und der Return-Taste. Nachdem die nötigen Parameter verändert sind, kann gedruckt werden, oder das unterbrochene Programm wird fortgesetzt mit ESC.

Druck-Optionen

Fast alle folgenden Optionen können direkt im Bildschirm-Menü geändert werden. Nur bei wenigen ist es nötig, eine Folge von Befehlen wie oben beschrieben an das Interface zu schicken. Fast alle Optionen sind auch von selbstgeschriebenen Programmen aus aufrufbar, z.B. bewirkt die Anweisung

```
1020 PRINT DRUCK$;"R"
```

in einem BASIC-Programm, daß zukünftig das Bild um 90 Grad gedreht wird, wenn man Grafik ausdrucken läßt („R" steht für „Rotieren"). Die Variable DRUCK$ ist ein besonderes Zeichen, das dem Interface sagt: „Hier kommt ein Befehl!". Welches Zeichen (Buchstabe, Ziffer, Sonderzeichen, meist jedoch ein Ctrl-Zeichen) das sein soll, kann mit dem Interface frei vereinbart werden. Im folgenden sind alle zur Verfügung stehenden Optionen aufgeführt.

Optionen

Grafik- und Textauswahl:

– Schwarze Flächen für den Druck mit einer Farbe füllen (für Farbdrucker)
– Weiße Flächen für den Druck mit einer Farbe füllen (für Farbdrucker)
– Farbiges Drucken wählen (für Farbdrucker)
– Inverses Drucken (Schwarz und Weiß vertauscht) wählen
– Beliebigen Ausschnitt aus der hochauflösenden Grafik wählen
– Grafik in doppelter Größe wählen
– Grafik um 90 Grad gedreht wählen
– Grafikschirme 1 und 2 nebeneinander wählen
– Grafikschirm 1 wählen
– Grafikschirm 2 wählen
– Doppelt hochauflösende Grafik wählen
– Niedrig- oder hochauflösende Grafik wählen
– Grafik ohne oder mit vier Zeilen Text wählen
– Angezeigte Grafik drucken
– Textschirm drucken
– Videx-80-Zeichen-Text drucken (für Apple II+ mit Videx-80-Zeichenkarte)

Seiten-Layout:

– Seiten numerieren
– Seitennummer für Start festlegen
– Seitenüberschrift festlegen
– Linken Rand einstellen
– Buchstaben pro Zeile einstellen
– Druckzeilen pro Seite einstellen

Drucker-Steuerung:

– Line Feed nach Carriage Return einstellen
– Bit 7 einstellen
– Druckertreiber des Benutzers aktivieren
– Serielle Schnittstelle auf 300..9600 Baud einstellen
– Befehlszeichen für Interface ändern (Ausgangswert: CHR$(9))
– Alle Interface-Funktionen ausschalten, Zeichen direkt an Drukker weitergeben
– Serielles Interface an/aus
– Paralleles Interface an/aus
– DIP-Schalterstellung für Drukkertyp ändern
– Alle Interface-Parameter auf Ausgangswerte setzen
– Status aller Parameter einfrieren, Software-Änderungen der Parameter ignorieren

Fehler und Schwächen

In Anbetracht der langen Liste der Fähigkeiten von Fingerprint Plus finden sich erstaunlich wenig Schwachpunkte. Die doppelt niedrigauflösende Grafik kann nicht ausgedruckt werden. Das Interface stellt nicht automatisch fest, ob einfach oder doppelt hochauflösende Grafik auf dem Bildschirm angezeigt wird; dieser Punkt muß vor dem Ausdruck per Menü oder programmgesteuert geklärt werden.
Schließlich treten manchmal Probleme auf, wenn Anweisungen direkt eingegeben wurden, die eine Zahl beinhalten: Z.B. kann es vorkommen, daß eine Menü-Steuerung des Seiten-Layouts nicht mehr möglich ist, nachdem das Layout – wie oben beschieben – direkt als Befehl an das Interface gegeben wurde. In einem solchen Fall muß die „Notbremse" gezogen werden, indem man alle Parameter auf ihre Ausgangswerte zurücksetzen läßt.

Ein-Blick

Name:	Fingerprint Plus
Einsatz:	Serielles und paralleles Interface mit Sofort-Bildschirm-Druck
Gesamtwertung:	****
Dokumentation:	(** für Vorab-Handbuch)
Zweckdienlichkeit:	*****
Optischer Eindruck:	****
Verarbeitung:	*****
Allgemeine Bedienbarkeit:	*****
Verhalten bei Bedienungsfehlern:	****
Preis/Leistung-Verhältnis:	****
Preis:	$ 150,–
Hersteller:	Thirdware Products, Miami, USA

(Die beste Bewertung entspricht 5 Sternen)

Die bessere Maus

Maus für Apple-II-Rechner

getestet von Thomas Bühner

Wer sich eine Apple-Maus angeschafft hat, um die Bedienung vieler Programme einfacher zu gestalten, kennt die Probleme, die bei ihrer Benutzung auftreten: Man braucht eine freie ebene Fläche direkt neben dem Computer mit einer Größe zwischen DIN A5 und DIN A4, um mit der Maus darauf hin- und herfahren zu können. Der Untergrund darf weder zu rauh noch zu glatt sein, damit die Kunststoff-Kugel, die an der Unterseite der Maus angebracht ist, gut rollen kann. Bewegt man sie auf dem blanken Schreibtisch oder einem Blatt Papier, wie es meist geschieht, verrutscht sie – und damit der Cursor – oft ein wenig, wenn sie losgelassen wird. Auch wenn man sie hochhebt, um mit der Bewegung an einer anderen Stelle des Untergrunds weiterzufahren, verrutscht der Cursor auf dem Bildschirm meist beim Anheben und Aufsetzen, weil die Kugel sich dabei leicht verschiebt.

Arbeitsweise

Diese Probleme treten weniger stark auf bei der Benutzung der „A+ Mouse", die nach einem optischen Prinzip arbeitet. Statt die Bewegung einer rollenden Kugel zu verfolgen, nimmt eine Leuchtdiode wahr, um welchen Betrag die Maus auf einem definierten Untergrund verschoben wird.
Als Arbeitsfläche dient eine 19 x 23 cm große Metallplatte, auf der sich

vertikal blaue und horizontal graue Linien im Abstand von einem Millimeter befinden. Ihre Oberfläche ist mit einem unempfindlichen Kunststoff versiegelt, der in der Haftung etwa einer gewachsten Schreibtischplatte entspricht. An der Unterseite der Maus sind zwei Filzstreifen angebracht, die für eine gute Beweglichkeit sorgen.

Kompatibilität

Die A+ Mouse ist 100% hardware- und software-kompatibel zur Apple-Maus.

Platzersparnis

Die Reibung zwischen dem Filz und der Kunststoff-Oberfläche der Arbeitsplatte ist recht groß, so daß die Maus erst bei einer Neigung des Untergrunds von 20 Grad zu rutschen beginnt. Somit braucht für den Betrieb auf dem Schreibtisch kein Platz mehr freigehalten zu werden. Wenn die Maus gebraucht wird, zieht man einfach unter den Arbeitsunterlagen die Metallplatte hervor und legt sie einigermaßen eben auf die Papierstapel.

Genauigkeit

Wegen der hohen Reibung verrutscht die A+ Mouse nicht, wenn man sie losläßt, um mit der Tastatur weiterzuarbeiten. Die kleinste Strecke, die man mit der optischen Maus fahren kann, beträgt etwa 1/3 mm, was somit erheblich unter der Auflösungsstärke des Strichabstandes von 1 mm liegt. Das hat zur Folge, daß eine sehr exakte und sensible Führung möglich ist, was jeder zu schätzen weiß, der mit Grafik-Programmen arbeitet. Auch beim Anheben und Absetzen resultieren geringere Cursor-Bewegungen als bei der gewohnten mechanischen Apple-Maus.

Kein Interface

Der große Nachteil der A+ Mouse ist jedoch, daß kein Interface mitgeliefert wird. Da sie vor allem für enttäuschte Apple-Maus-Besitzer gedacht ist, gehen die Hersteller wohl davon aus, daß jeder Käufer bereits das originale Maus-Interface sein eigen nennt. Für Apple-IIc-Besitzer stellt das kein Problem dar: In ihrem Computer ist es ohnehin schon serienmäßig eingebaut.

Ein-Blick

Name:	A+ Mouse
Einsatz:	Optische Maus mit besserer Bedienbarkeit
Gesamtwertung:	****
Dokumentation:	**** (5 Seiten)
Zweckdienlichkeit:	*****
Optischer Eindruck:	*****
Verarbeitung:	*****
Allgemeine Bedienbarkeit:	*****
Preis/Leistung-Verhältnis:	** (fehlendes Interface)
Preis:	$ 99,-
Hersteller:	Mouse Systems, Santa Clara, USA

(Die beste Bewertung entspricht 5 Sternen)

Joystick einfacher anschließen

Game-Socket-Extender

getestet von Thomas Bühner

Zwei Probleme können bei dem Anschluß von Joysticks an einen Mikrocomputer der Apple-II-Serie auftreten. Zum einen hat der gekaufte Joystick vielleicht einen „falschen" Anschlußstecker. Beim Apple II+ kann nur auf der Hauptplatine rechts hinten ein 16-Pin-DIL-Stecker angeschlossen werden. Man kann hier also keinen Joystick mit einem der mittlerweile beim Apple IIe und IIc üblichen kompakteren 9-Pin-Stecker (trapezförmig) verwenden.
Doch selbst wenn ein Joystick einen 16-Pin-DIL-Stecker hat, besteht immer noch das Problem, daß der Anschluß dafür sowohl beim Apple II+ als auch beim IIe auf der Hauptplatine des Computers sitzt. Das bedeutet, daß jedesmal, wenn der Joystick gegen Paddles, Koala Pad oder ein anderes Gerät ausgetauscht wird, die Gehäusehaube abgenommen werden muß, weil der Anschluß von außen nicht zugänglich ist.

Funktionsweise

Diese Sorgen ist man los, wenn man über einen der „Happ Game Port Extender" verfügt, eine Anschlußstelle für Joysticks, die außen am Computer sitzt. Das Prinzip ist simpel: Eine Buchse für 9-Pin- oder 16-Pin-Stecker wird an der Seite des Apple angebracht und über ein Verlängerungskabel mit der Hauptplatine verbunden. Nun braucht die Gehäusehaube nicht mehr abgenommen zu werden, wenn man Joysticks und Paddles tauscht.

Montage

Befestigt wird der Extender entweder mit zwei Schrauben an den Ventilation-Schlitzen des Apple oder man zieht ein Stück Folie ab und klebt ihn auf eine beliebige Stelle des Gehäuses. Entscheidet man sich für die Ventilation-Schlitze, muß beim Apple IIe das Gehäuse abgeschraubt werden, um die Montage zu ermöglichen, da sich sein Gehäuse etwas von dem des Apple II+ unterscheidet.

Ausführungen

Für die Buchse des Extenders stehen verschiedene Möglichkeiten zur Wahl. Begnügt man sich mit einem Anschluß, so sind eine 9-Pin- und eine 16-Pin-Version verfügbar, die auf dem 45 x 45 mm großen Extender untergebracht sind. Möchte man aber gleichzeitig Paddles und Joystick am Computer anschließen, braucht man zwei Buchsen. Hier kann man sich nun aussuchen, ob man auf einer Fläche von 65 x 75 mm lieber zwei 16-Pin-Anschlüsse oder eine 9-Pin- und eine 16-Pin-Buchse haben will. Auch für den Apple IIc gibt es zwei spezielle Versionen, die eine bzw. zwei 9-Pin-Buchsen haben.
Bei der Version mit zwei Buchsen kann mit einem Schalter zwischen den angeschlossenen Geräten hin- und hergeschaltet werden. Für die wenigen Action-Spiele, die den Betrieb von zwei Joysticks – und somit zwei Spielern – gleichzeitig erlauben, ist auch eine entsprechende Schalterstellung vorhanden.

Sicherheit

Beim 16-Pin-Anschluß besteht nicht wie sonst die Gefahr, daß sich ein Beinchen des Steckers verbiegt oder sogar abbricht: Der Stecker muß nicht in die Fassung gedrückt werden, sondern man legt ihn dort nur locker ein. Festgehalten wird er erst, wenn man einen kleinen Hebel kippt.

Aussehen

Leider sind diese Extender nicht sehr angenehm zu betrachten. Ihr verhältnismäßig rohes, technisches Aussehen paßt nicht zum eher weichen Design des Apple-Gehäuses. Die einzige Möglichkeit, den Extender außer Sicht unterzubringen, besteht darin, ihn auf die Rückseite des Apple zu kleben, wobei dann beim Apple IIe allerdings zwei bis drei Kabel-Öffnungen unbrauchbar werden.

Ein-Blick

Name:	Happ Game Socket Extender
Einsatz:	Leichtere Bedienung des Apple-Game-Ports
Gesamtwertung:	****
Zweckdienlichkeit:	*****
Optischer Eindruck:	**
Verarbeitung:	****
Allgemeine Bedienbarkeit:	*****
Preis/Leistung-Verhältnis:	****
Preis: (1 Anschlußbuchse)	$ 19,-
(2 Anschlußbuchsen)	$ 35,-
Hersteller:	Happ Electronics, Inc., Oshkosh, USA

(Die beste Bewertung entspricht 5 Sternen)

CP/M-Karte für Apple IIc

getestet von Harald Grumser

Ein Grund für den Erfolg des Apple II+ und IIe liegt in der Möglichkeit, gleichzeitig sowohl 6502- als auch Z80-Programme benutzen zu können, wodurch sich das riesige Angebot der CP/M-Software erschließt. Der IIc erlaubt normalerweise keine Aufrüstung mit Zusatzkarten. Die englische Firma Cirtech bietet für dieses Gerät eine Huckepack-Platine an, die nach dem Einbau (s. Peeker 2/86, S. 63) die Benutzung von CP/M 3.0 (CP/M Plus) und CP/M 2.23 gestattet. Die Karte wird in Deutschland exklusiv von der Firma Semjan in Frankfurt vertrieben.

1. Software

1.1. CP/M 3.0 beim IIc

Unter CP/M 3.0, das auf Wunsch mitgeliefert wird, werden 128K RAM verwaltet, und ein im Lieferumfang enthaltenes Patch-Programm für MBASIC erlaubt auch hier den Zugriff auf die höhere Speicherkapazität. Zu den Besonderheiten dieser IIc-Implementierung gehört u. a. die Unterstützung der Maus in allen CP/M-Programmen (s. u.).
Die Einrichtung von Zusatzmenüs, die über eine Apfeltastenkombination zugänglich werden, bietet einen fast mac-ähnlichen Komfort. So können zu jeder Zeit – also auch während der Ausführung eines Anwenderprogramms – Disketten formatiert und kopiert oder ein Dump des Bildschirms auf den Drucker ausgegeben werden.
Durch eine raffinierte Interrupt-Steuerung (auch die Maus wird vom Z80 verwaltet) wurden verschiedene Puffer realisiert, die eine reibungslose Ein/Ausgabe gewährleisten. Selbst der Drucker ist mit 12.000 Zeichen gepuffert, wodurch z. B. ein Bildschirm-Dump ausgegeben werden kann, während man die normale Arbeit fortsetzt.

1.2. Erweitertes BIOS

Der wesentliche Teil der Vorzüge des CP/M 3.0 beim IIc liegt in dem erweiterten BIOS. Die Bildschirm- und Tastatur-Ein/Ausgabe erfolgt über den Z80-Prozessor, wodurch sich eine bedeutend größere Verarbeitungsgeschwindigkeit ergibt. Nur für den Diskettenzugriff wird auf den 65C02 zurückgeschaltet.
In das BIOS aufgenommen wurden auch Routinen zur Maussteuerung. Wie bereits oben erwähnt, wird die Maus durch den Z80 bedient und kann durch ein Assembler-Programm über BIOS-Aufrufe abgefragt werden.
Auch sind BIOS-Aufrufe implementiert worden, die z. B. ein Fenster in den Bildschirm einblenden und den ursprünglichen Bildschirminhalt retten. Zur Programmierung dieser zusätzlichen Funktionen kann ein Programmierpaket für ca. DM 500 erworben werden, das neben Assembler und Debugger von Digital Research drei umfangreiche Manuals bietet, die im Detail auf die Besonderheiten von CP/M 3.0 eingehen.

2. Hardware

Der Einbau der Karte dauert länger als eine halbe Stunde und sollte nicht nur wegen der Garantie-Ansprüche – die diesbezügliche Angabe im Peeker 2/86, S. 63 war falsch – von einem Apple-Händler vorgenommen werden. Es kann nämlich nicht verschwiegen werden, daß die Karte nicht auf jedem IIc läuft. So konnte auf dem IIc in der Peeker-Redaktion weder die ursprünglich zugesandte Karte, noch die Zweitkarte, die Herr Semjan persönlich vorbeibrachte, zufriedenstellend zum Laufen gebracht werden, weshalb sich auch der Testbericht verzögert hatte. Offenbar versetzt die Karte den IIc beim Einschalten nicht immer in den normalen Kaltstart-Zustand.

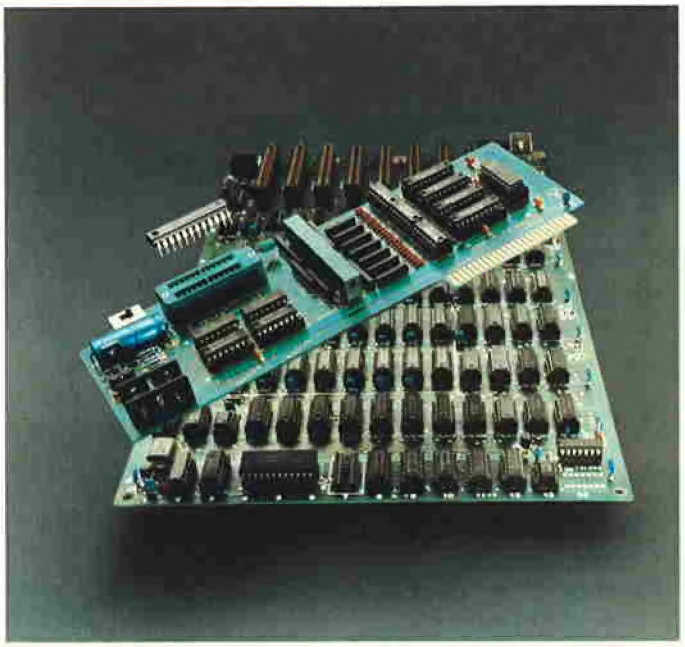
Beispiel: Wenn man den Strom einschaltet, läßt sich das Programm „Superquick", das aufgrund des uns vorliegenden Quellcodes nur legale Softswitches betätigt, nicht korrekt starten, obwohl es die Z80-Karte gar nicht benutzt. Drückt man bei unserem bereits eingeschalteten IIc jedoch mehrmals Ctrl-Apfel-Reset, so funktioniert das Programm „meistens"; ohne die Karte funktioniert es auf unserem IIc immer. Leider bietet das mitgelieferte Handbuch keine detaillierten technischen Informationen, und auch Herr Semjan konnte sich nicht erklären, warum ausgerechnet bei unserem IIc die Karte nicht funktioniert.
Die Cirtech-Karte verfügt nicht über einen eigenen Quarz. Nach Angabe von Herrn Semjan wird die Frequenz durch einen freischwingenden Oszillator abgeleitet. Es dürfte daher nicht möglich sein, extrem zeitkritische Routinen, wie z. B. Disk-I/O, in Z80 zu realisieren. Diese Frequenz liegt laut Herstellerangaben zwischen 4 und 5,7 MHz, obwohl der verwendete Z80 noch mit 8 MHz arbeiten könnte.

Fazit

Wer mit seinem IIc auch in die Welt der CP/M-Programme (Wordstar, Turbo-Pascal...) einsteigen möchte, findet mit dieser Karte zu einem recht günstigen Preis von unter DM 1000,– (inkl. Betriebssystem) eine unkomplizierte Möglichkeit. Die umfangreiche Anpassung des Betriebssystems an die Hardware-Gegebenheiten des Apple bieten einen zusätzlichen Komfort, der anderen CP/M-Benutzern vorenthalten bleiben muß.

Die Anwendungsmöglichkeiten des IIc werden durch diese Karte in jedem Fall vervielfacht.

PAL-Programmer für Apple



Ein vollständiges Paket für die Programmierung von PAL-ICs (Programmable Array Logic) bietet die Firma nuclear interface in Münster (Tel. 02 51 / 27 35 85) an. Es besteht aus einer Steckkarte für den Slot des Apple II oder Apple IIe und einer umfangreichen Software. Der PAL-Programmer ermöglicht das Auslesen, Kopieren und Brennen von 20-poligen und 24-poligen PALs sowie das Brennen der Security-Fuse (das PAL kann dann nicht mehr ausgelesen oder kopiert werden: Schutz vor Nachbau!). Der im Softwarepaket enthaltene PAL-Assembler (PALASM von MMI) ermöglicht die Übersetzung von Logikfunktionen in die entsprechenden Fuse-Pattern (Programmiermuster). Ein Screen-Editor für Fuse-Pattern, mit dem die gesamte Fuse-Matrix auf dem Bildschirm oder dem Drucker ausgegeben werden kann, steht zusätzlich zur Verfügung. Das gesamte Paket kostet etwas über 1.100,– DM.

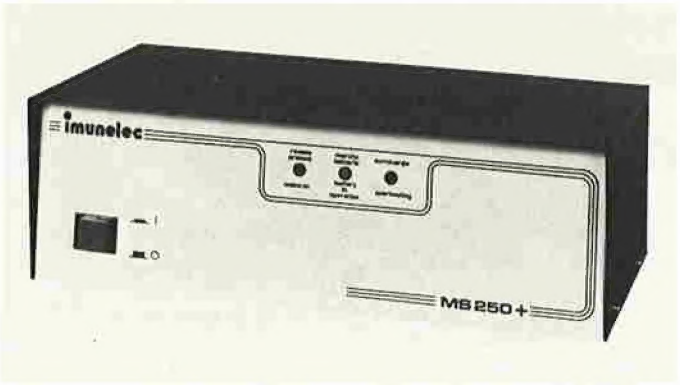
Roll-Maus

Als Low-cost-Version für den Apple II bietet sich die Roll-Maus der Firma W. Dederichs an. Sie hat eine Lauffläche von 10 × 10 cm und arbeitet auf mechanischer Basis. Die Maus verfügt über zwei Push-Buttons und ist daher für viele Programme einsetzbar, die normalerweise mit zwei Paddles arbeiten. Angeschlossen wird die Maus an den Game-I/O. Unterschiedliche Konfektionierungen für Apple II, Apple II+ bzw. Apple IIe sowie Apple IIc sind erhältlich.

Wirft man einen neugierigen Blick ins Innere der Maus, so wird man feststellen, daß die Übertragung der Rollbewegungen der Kugel durch eine Übersetzungsschnecke auf die Potis erfolgt. Hierdurch liefert die Maus sehr präzise und konstante Werte zwischen 0 und 255.
Mit der Maus wird eine Softwareanpassung geliefert, um das Programm Mouse-Paint auch mit dem Game-I/O betreiben zu können. Weitere Programme, wie z. B. KoalaPad-Software lassen sich ebenfalls gut mit dieser Maus anwenden.
Bei einem Preis von DM 195,00 für diese Maus inkl. Software-Anpassung für Mouse-Paint dürfte dies eine interessante Lösung für grafikbegeisterte Computeranwender sein.

USV: Unterbrechungsfreie Stromversorgung

Microstal ist eine USV-Anlage, speziell entwickelt zum Schutz von Mikrocomputern vor Netzstörungen wie
– totalem Stromausfall,
– Mikrounterbrechungen,
– Stromschwankungen,
– Frequenzschwankungen.
Bei Microstal wird der Netzstrom gleichgerichtet, in der eingebauten gasdichten wartungsfreien Batterie „gelagert" und über den Wechselrichter wieder neu erzeugt, unabhängig vom Stromnetz. Somit ist gewährleistet, daß nach einem Netzausfall die für den Computer so tückische Frequenzverschiebung ausgeschlossen ist. Eingang vom Netz und Ausgang der USV-Anlage sind absolut unabhängig voneinander. Microstal ist in Verbindung mit Apple II und III verwendbar.



Z80+ Card

Mit dieser Z80-Karte erhält man zwei Z80-Karten in einer, da sie softcard-kompatibel (2.046 MHz) und ALS-CP/M-kompatibel (8 MHz ohne Waitzyklen) betrieben werden kann. Die mitgelieferte Software ermöglicht es, aus einem vorhandenen Softcard-CP/M 2.20 eine CP/M-2.2-Version zu erzeugen, die im 8-MHz-Modus dieser Karte läuft. Dieses CP/M bietet u.a. schnelleren Disk-I/O, Ehring- und Erphi-Kompatibilität (bis 640K), integrierte RAM-Disk und Unterstützung anderer gängiger RAM-Karten. Natürlich ist auf der Z80+ Card auch CP/M 3.0 einsetzbar, da die Z80H-CPU insgesamt 120K RAM ansprechen kann. Preis DM 696,- für Z80+ Card, Diskette und 40seitiges Manual.

Inserentenverzeichnis Peeker 4/86

Computerbücher
die gehen, für Computer die kommen.

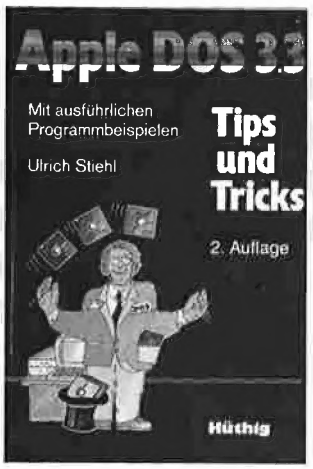
Arne Schäpers
ProDOS-Analyse
Versionen 1.0.1, 1.0.2, 1.1.1
1985, 470 S., kart., DM 68,–
ISBN 3-7785-1134-3

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

MEGA-CORE

läßt keine Wünsche offen:

- 10 Megabytes in Ihrem Apple
- 4 Betriebssysteme im Zugriff
- Superstarkes Netzteil mit Kühlung

MEGA-CORE macht durch Einbau eines 10 MB Festplattenlaufwerks Ihren Apple //e zu einem XT.

MEGA-CORE besteht aus:

- 3 1/2" Festplatte mit 10 MB
- Harddiskcontroller auf einer Slotkarte
- 70 W Netzteil (12 W f. Platte / 58 W f. Apple)
- Lüfter zur Kühlung des Rechners
- Software zur Anpassung von vier Betriebssystemen, DOS, CP/M, PASCAL u. ProDOS
- Ausführliches deutsches Handbuch

MEGA-CORE macht Ihren Apple zum Profisystem:

- Jedes Betriebssystem bootet von der Platte
- Alle Betriebssysteme gleichzeitig auf der Platte bei extrem schnellem Zugriff
- Die ganzen 10 MB sind frei konfigurierbar

MEGA-CORE ist ein Produkt von:

FRANK & BRITTING
Elektronik Entwicklungs GmbH
Langestr. 4, Postfach 1129, 7529 Forst
Telefon 07251/103068-69.
Telex: 7822452 fub d

Die Harddiskcontroller-Spezialisten

Seit 1. Januar 1986 können Sie MEGA-CORE auch mit 20 MB haben.
Fragen Sie den Fachhandel oder uns nach Preisen und Bezugsquellen, oder holen Sie sich gegen Voreinsendung von DM 5,– eine Demo-Diskette.