

Nr. 9/85 September

DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

# PEEKER

MAGAZIN FÜR APPLE-COMPUTER

Software und Kopierrecht

Neues FID für DOS 3.3

Fastboot-Routine für ProDOS

Flag Monitor

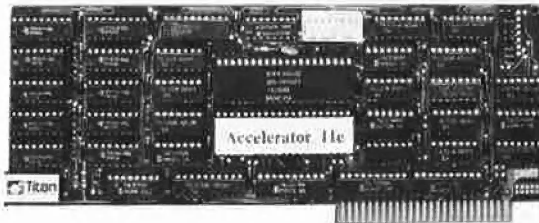
P-Code-Optimierung

Tastaturen im Test

Grafikdump für NEC



## Accelerator™ IIe macht Ihren Apple® II, II Plus oder IIe dreieinhalbmal schneller.



Jetzt laufen VisiCalc®, Apple Writer, PASCAL, BASIC, Datenbanken usw. endlich ohne langen Zeitverlust.

Stecken Sie einfach die ACCELERATOR IIe Karte in irgendeinen Slot und beobachten Sie, wir Ihr Apple loslegt!

ACCELERATOR IIe besitzt seinen eigenen schnellen 6502 Prozessor und 80 K-Byte Hochgeschwindigkeitspeicher, einschließlich einer eingebauten schnellen Sprachkarte und schnellem RAM-Speicherplatz für die ROM-Sprache.

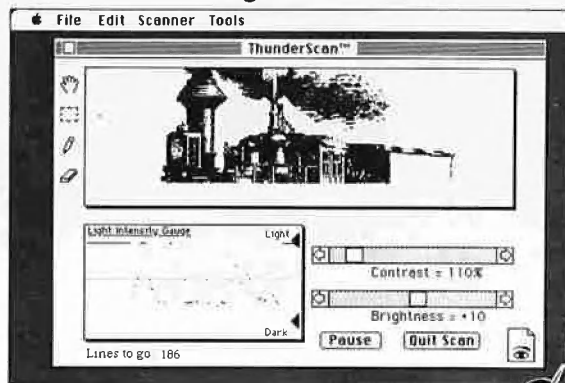
Direkt von Pandasoft (Titan Distributor für Deutschland) oder bei Ihrem Applehändler.

**pandasoft** Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo-Fr 10-18 Uhr, Sa 10-13 Uhr  
Telefon: 0 30/31 04 23 · Telex: 1 85 859

## ThunderScan.™

Ein neues optisches Lesegerät, das beliebige Vorlagen in MacPaint überträgt: Fotos, Zeichnungen, Landkarten und Illustrationen werden in den Apple-Imagewriter eingespannt und von einem Lesekopf, der das Farbband ersetzt, abgetastet.



- 32 Graustufen
- 80 Punkte/cm Auflösung
- Übertragungsmaßstab 25% - 400%
- Vorlagen bis 20 x 25 cm
- Nachträgliche Veränderung des Kontrasts und der Helligkeit.



ThunderScan

**pandasoft** Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo-Fr 10-18 Uhr, Sa 10-13 Uhr  
Telefon: 0 30/31 04 23 · Telex: 1 85 859

**Orange™**  
Printer Interface

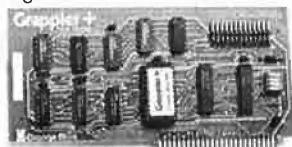
Druckerinterfaces für Apple II+/e/ c/III Interfaces auf dem **neuesten**

Stand der Technik. Kompatibel mit allen gängigen Druckern wie: APPLE, EPSON, STAR, NEC, OKIDATA usw. Passende Treiber-Software wird über Dip-Switch ausgewählt.

**Grappler +**  
Printer Interface

Grafikfähiges Druckerinterface das keine Wünsche mehr offen läßt. ermöglichen die volle Kontrolle

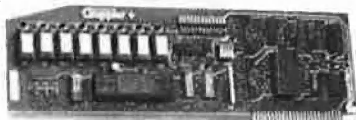
Über 2 Dutzend Kommandos über alle Möglichkeiten Ihres Druckers. Jetzt auch mit **IIe Features: Double Hires Graphics** und **80 Zeichen Dump** mittels Druckerpuffer nachrüstbar über Bufferboard.



**Grappler +**  
BUFFERED

ten 16 K Druckpuffer, der auf 32 oder 64 K aufrüstbar ist.

Besitzt alle Vorzüge des Grappler +, hat aber zusätzlich einen integrier-



**SERIAL Grappler**  
Printer Interface

Serielles Druckerinterface speziell für den **Apple Imagewriter**.

**MOTLINK**

Seriell-nach-Parallel-Wandler für den IIc im Kabel integriert.

**GRAPPLER C**

wie Hotlink, jedoch zusätzlich Imagewriter Emulation und Grafik Software-Diskette.

**pandasoft** Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo-Fr 10-18 Uhr, Sa 10-13 Uhr  
Telefon: 0 30/31 04 23 · Telex: 1 85 859

## Sie haben einen Apple...

wir haben die  
Software...



und die  
Hardware...



wir haben die  
Bücher...



und die  
Zeitschriften...



**Fordern Sie unseren Gratiskatalog an!**

ALLES FÜR DEN APPLE II+, IIe, IIc UND MACINTOSH

**pandasoft** Dr.-Ing. Eden

UHLANDSTR. 195 · D-1000 BERLIN 12  
TEL.: (030) 310 423 · TELEX: 18 58 59

Autorisierter Apple Fachhändler · MICROSOFT Distributor

Ich besitze einen Apple. Bitte schicken Sie mir Ihren kostenlosen Katalog.  
Name: \_\_\_\_\_  
Adresse: \_\_\_\_\_



## Ein Jahr Peeker

Mit dem vorliegenden September-Heft jährt sich das Erscheinen des Peekers, ein gebotener Anlaß, um über das Erreichte zu reflektieren. Als wir den Peeker im September des letzten Jahres gründeten, war uns klar, daß der Leser zunehmend auf kritische Berichte und sorgfältig redigierte Programme Wert legt:

Was die *kritischen Berichte* anlangt, so geht es im Mikrocomputer-Blätterwald gelegentlich zu wie im Opportunitätsjournalismus: Einige Kleinigkeiten werden bemängelt, aber im übrigen erstrahlt das Produkt im hellsten Sonnenschein. Als wäre jede Hardware ein Spitzenerzeugnis! Als wäre jede Software eine ingeniose Schöpfung! Schönfärbereien dieser Art haben wir nie zugelassen.

Was die Überarbeitung von Programmen betrifft, so lesen wir in H. Feichtingers „Arbeitsbuch Mikrocomputer“: „Es gibt tatsächlich Zeitschriften, die Programme vor dem Abdruck testen!“ Da kann man nur sagen: Und nicht nur das! Jedes Peeker-Programm wird überarbeitet, oftmals von dem einen Assembler in den anderen umgeschrieben, vereinheitlicht und in eine lesbare Form gegossen. Außerdem ist der Peeker wohl die einzige Zeitschrift, in der „alle“ Programme im modernen Lichtsatz gesetzt werden, so daß ein Listing, das bei manch einer Zeitschrift eine ganze Druckseite einnimmt, beim Peeker auf einer halben Seite Platz findet.

Unbefriedigend ist nach wie vor die Terminologie und die Fehlerlosigkeit der Programme:

Der Peeker kann sich nicht zum Vorreiter einer Sprachbereinigung machen. Heißt es z. B. „der oder die oder das File/Array?“ Jeder bastelt sich offenbar sein eigenes EDV-Chinesisch zusammen. Entscheiden Sie sich spaßeshalber selbst einmal für das „richtige“ Genus und schlagen Sie dann beispielsweise bei Niklaus Wirths „Algorithmen und Datenstrukturen“, S. 50 und 60 nach. Sie werden überrascht sein!

Tippfehler in den Listings und Bugs in den Programmen werden sich wohl nie völlig ausmerzen lassen. Schon allein aus zeitli-

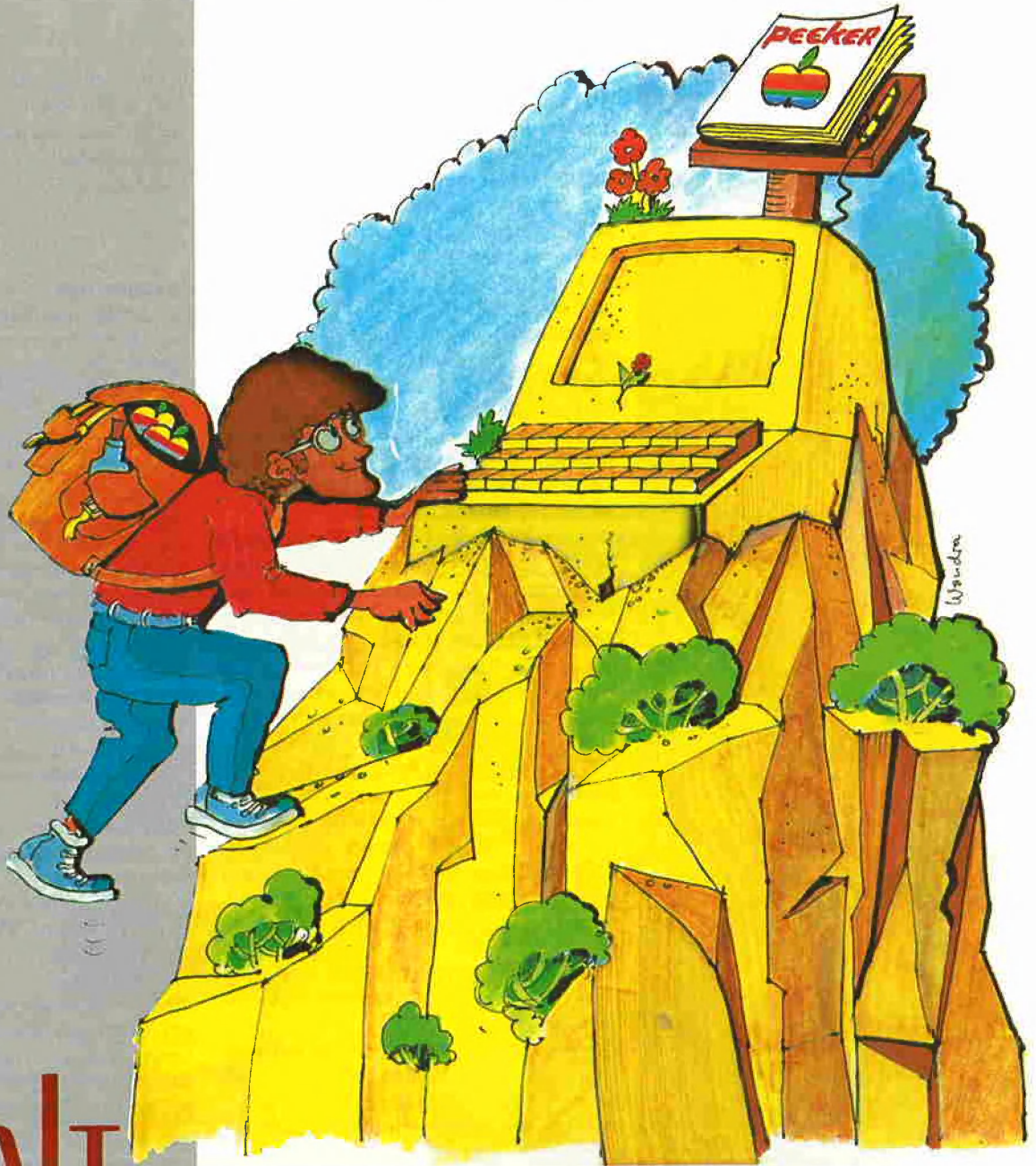
chen Gründen sehen wir uns leider außerstande, die Aufsätze und Programme derart akribisch zu durchleuchten, daß kein Schnitzer mehr stehenbleibt. Perfektionismus bleibt somit ein unerfüllbarer Traum.

### Kurzumfrage

Die letztjährige September-Ausgabe enthielt eine Kurzumfrage in Form einer Postkarte (s. a. Trennkarte in „dieser“ Ausgabe, S. 45), die über 4500 Leser ausfüllten. Damals ergaben sich bezüglich der Geräteverteilung folgende Prozentsätze: IIc (5,1 %), IIe (29,5 %), II+ (30,1 %), Kompatible (32,3 %), Macintosh (1,3 %); Apple III, Lisa u. a. Geräte zusammen ca. 1 %. Inzwischen hat sich die Apple-Szene völlig verändert: Der Apple III und die Lisa wurden aus der Produktion genommen. Der Macintosh hat sich bislang nicht durchgesetzt mit der Folge, daß ein Teil der ehemaligen Apple-II-Besitzer, die ein Gerät der höheren Leistungs- und Preiskategorie erwerben wollen, abzuwandern beginnen, weil das Nachfolgemodell zu dem Apple II immer noch auf sich warten läßt.

Auch dieses September-Heft enthält wieder eine Umfrage, die u. a. die Rubrik „Ich besitze zusätzlich folgenden Nicht-Apple-Computer“ aufweist. Sollte hier ein bestimmtes Fremdgerät sehr häufig genannt werden, so werden wir zu überlegen haben, ob wir dieses Gerät in Zukunft angemessen berücksichtigen (z. B. 4 Seiten pro Heft). Wir bitten Sie, liebe Peeker-Leser, uns die Umfrage-Karte zuzusenden, denn damit haben Sie selbst Einfluß auf die Themenverteilung.

Ulrich Stiehl



# INHALT

## 9/85

### Impressum

Peeker  
Magazin für Apple-Computer  
2. Jahrgang 1985  
ISSN 0176-9200  
© für den gesamten Inhalt  
einschließlich der Programme  
Dr. Alfred Hüthig Verlag,  
Heidelberg 1985

Verleger und Herausgeber:  
Dipl.-Kfm. Holger Hüthig  
Geschäftsführung Zeitschriften:  
Heinz Melcher  
Chefredakteur:  
Ulrich Stiehl (us) Tel. (0 62 21) 48 93 52  
(Bitte nur in redaktionellen Angelegenheiten  
anrufen)

Anzeigenleitung:  
Jürgen Maurer, Tel. (0 62 21) 48 92 18  
z. Zt. gilt Anzeigenpreisliste Nr. 3  
Vertriebsleitung:  
Ruth Biller, Tel. (0 62 21) 48 92 80  
Produktionsleitung: Gunter Sokollek  
Gestaltung: Rainer Schmitt  
Titelbild: H. Wondra

# peeker

<b>Recht</b>		<b>CP/M</b>	
Ave auctor, plagiarii te salutant Software und Zitatrecht von Ulrich Stiehl	6	MBASIC für den Applesoft-Profi von Jörg Lange	48
Urheberrechtsnovelle	11	<b>Händler-Profi</b>	
<b>DOS 3.3</b>		HIB – Der Computerladen in Nürnberg	59
Ein neues FID für DOS 3.3 Mit Auto-FID für RAM-Disk-Besitzer von Harald Grumser	12	<b>Testberichte</b>	
<b>ProDOS</b>		Erfahrungsbericht OPERATOR I von Reiner Hammerschmidt	60
ProDOS-Fastboot oder wie man ProDOS in sechs Sekunden bootet von Arne Schäpers	20	Die OPERATOR IIe getestet von Harald Grumser	60
<b>Hardware</b>		Dazzle Draw und Mousepaint Erfahrungsbericht von Dieter Charchot	61
Super-HGR für NEC und ITOH-Drucker von Rainer Hammerschmidt	30	DMP-Charger – eigene Zeichensätze auf dem Matrixdrucker getestet von Harald Grumser	62
<b>Assembler</b>		Apple II/IIe Assembler-Kurs getestet von Dr. Jürgen B. Kehrel	62
Flag Monitor Eine Taktzähler-Utility von Michael G. Schneider	32	Deutsche Sprachausgabe für den Apple getestet von Harald Grumser	63
<b>Pascal</b>		<b>Kurzberichte</b>	64
Tips und Tricks in Pascal Teil 2: Kann man den P-Code optimieren? von Dieter Geiß	40	Pressestimmen zur Wirtschaftslage von Apple	69
Pascal-Preisausschreiben	58	<b>Quickie</b>	
		Umwandlung in Großbuchstaben von Harald Grumser	31
		<b>Leserbriefe</b>	66
		<b>Inserentenverzeichnis</b>	68
		<b>Errata</b>	
		Fourier-Analyse, Formatter, PLOT 2.0	69

# Ave auctor, plagiarii te salutant

Software und Zitatrecht

von Ulrich Stiehl



Heil Dir Autor, die Geistesräuber grüßen Dich.

## 1. Kopieren, Zitieren und Bearbeiten

Im *Peeker*, Heft 4/1985 hatten wir uns bereits mit dem Thema „Software und Kopierrecht“ auseinandergesetzt. Der nachfolgende Beitrag knüpft daran an und behandelt den erheblich diffizileren Themenkomplex des Zitatrechts unter Einbeziehung der Bereiche der Bearbeitung und freien Benutzung.

**Kopieren** heißt im Juristendeutsch „Vervielfältigen“ (UrhG §53; UrhG = Urheberrechtsgesetz). In der EDV bedeutet dies ein Überspielen, d.h. ein Verdoppeln oder Eins-zu-eins-Reproduzieren eines Computerwerkes (= Programms). Der Begriff der Eins-zu-eins-Reproduktion impliziert, daß das Programm erstens *vollständig* und zweitens *unverändert* kopiert wird. Eine (unvollständige) Teilkopie ist möglich, kommt jedoch in der Praxis kaum vor. Dagegen ist es denkbar, daß *nach* der Herstellung einer Kopie an ihr Änderungen („Patches“) vorgenommen werden. Sehen wir von diesen Sonderfällen ab, so können wir festhalten, daß man unter Kopieren üblicherweise das vollständige und unveränderte Vervielfältigen eines Originalwerkstücks versteht.

**Zitieren** bedeutet demgegenüber in der EDV die Übernahme eines Fremdprogramms in ein eigenes Programm. Der Regelfall ist das sog. *Kleinzitat*, d.h. die Übernahme eines *Teils* eines Fremdprogramms in ein eigenes Programm. Nur selten stößt man auf ein sog. *Großzitat*, d.h. die Übernahme eines *vollständigen* Programms in ein eigenes. Während ein kopiertes Originalwerk nach erfolgter Kopie ein Werkstück *für sich* darstellt, bewirkt das Zitat stets eine Einverleibung einer „Stelle“ aus einem fremden Werk in ein eigenes Werk. Normalerweise wird eine Stelle unverändert unter Nennung des Werktitels und Urhebers wiedergegeben, um den urheberrechtlichen Grundregeln über das Änderungsverbot (UrhG §53) und die Quellenangabe (UrhG §63) Rechnung zu tragen. Oft findet man jedoch auch veränderte Stellen und/oder ungenügende oder fehlende Quellenangaben. Ein literarisches Paradebeispiel ist Goethes Werk, dessen zahlreiche obszöne Passagen entweder bereinigt („purgiert“) oder als „— — —“ zitiert werden (s. z.B. Faust, Walpurgisnacht usw.). Wenn man die Quellenangabe „zu erwähnen vergißt“, liegt meist ein Plagiat vor (plagiar = fremdes geistiges Eigentum rauben;

danach plagiierten, Plagiat; plagiator und plagiarius = Räuber fremden geistigen Eigentums; danach Plagiator).

Neben dem Zitat gibt es noch die Möglichkeit der mehr oder weniger engen Ausrichtung an einem fremden Werk. Hier spricht man dann von der zustimmungspflichtigen **Bearbeitung**, die sich an eine Fremdvorlage eng anlehnt (UrhG §23), sowie von der nicht-zustimmungspflichtigen **freien Benutzung**, die die Fremdvorlage nur als Anregung nimmt (UrhG §24). Wenn man die diversen Spielarten von der freien Benutzung über die Bearbeitung und die modifizierten Zitate mit und ohne Quellenangabe bis zu den wörtlichen, exakten Zitaten Revue passieren läßt, wird offenbar, daß das Zitatrecht erheblich komplizierter und undurchsichtiger als das Kopierrecht ist, denn im Kopierrecht braucht man zur Ermittlung eines (Straf-)tatbestandes in der Regel nur Gleiches mit Gleichem zu vergleichen, während im Zitatrecht oft Gleiches mit Ähnlichem oder Ungleichem verglichen werden muß.

## 2. Das begriffliche Gerippe

Nachfolgend zitieren wir aus dem Urheberrechtsgesetz die relevanten Paragraphen: UrhG §13 **Anerkennung der Urheberschaft**: Der Urheber hat das Recht auf Anerkennung seiner Urheberschaft am Werk...

UrhG §23 **Bearbeitungen** und Umgestaltungen: Bearbeitungen oder andere Umgestaltungen des Werkes dürfen nur mit Einwilligung des Urhebers des bearbeiteten oder umgestalteten Werkes veröffentlicht oder verwertet werden...

UrhG §24 **Freie Benutzung**: Ein selbständiges Werk, das in freier Benutzung des Werkes eines anderen geschaffen worden ist, darf ohne Zustimmung des Urhebers des benutzten Werkes veröffentlicht und verwertet werden....

UrhG §51 **Zitate**: Zulässig ist die Vervielfältigung, Verbreitung und öffentliche Wiedergabe, wenn in einem durch den Zweck gebotenen Umfang

1. einzelne Werke nach dem Erscheinen in ein selbständiges wissenschaftliches Werk zur Erläuterung des Inhalts aufgenommen werden,
2. Stellen eines Werkes nach der Veröffentlichung in einem selbständigen Sprachwerk angeführt werden...

UrhG §62 **Änderungsverbot**: (1) Soweit nach den Bestimmungen dieses Ab-

schnitts die Benutzung eines Werkes zulässig ist, dürfen Änderungen an dem Werk nicht vorgenommen werden...

UrhG §63 **Quellenangabe**: (1) Wenn ein Werk oder ein Teil eines Werkes in den Fällen des ...§51... vervielfältigt wird, ist stets die Quelle deutlich anzugeben...

UrhG §106 **Unerlaubte Verwertung** urheberrechtlich geschützter Werke: Wer in anderen als den gesetzlich zugelassenen Fällen vorsätzlich ohne Einwilligung des Berechtigten ein Werk oder eine Bearbeitung oder Umgestaltung eines Werkes vervielfältigt, verbreitet oder öffentlich wiedergibt, wird mit Geldstrafe oder mit Gefängnis bis zu einem Jahr bestraft.

Die in den Paragraphen erwähnten Begriffe lassen sich folgendermaßen definieren: Eine **Bearbeitung** ist ein selbständiges Werk, das sich an ein anderes Werk stark anlehnt. Typische Beispiele für Bearbeitungen sind Übersetzungen (von Büchern in andere Sprachen), Vertonungen, Dramatisierungen und Verfilmungen (im UrhG gesondert geregelt). Die Bearbeitung setzt die *vorherige* Zustimmung = Einwilligung des Urhebers voraus.

Bearbeitungen von Computerwerken könnten Übersetzungen sein, doch ist mir hierzu keine Rechtsprechung bekannt. Bei natürlichen Sprachen gibt es bekanntlich unkomplizierte Übersetzungen (z.B. aus dem Englischen ins Deutsche) und komplizierte Übersetzungen (z.B. aus dem Sanskrit ins Chinesische). Ähnlich gibt es bei den Computersprachen leichte (z.B. von BASIC in Pascal oder vom Quellcode in den Objektcode) und schwere Übersetzungen (z.B. von BASIC in Assembler oder vom Objektcode in den Quellcode). Der Schwierigkeitsgrad kann also nicht als Richtschnur dienen. Dieser ganze Komplex ist offenbar noch nicht in der Rechtsliteratur durchdiskutiert worden.

Eine **freie Benutzung** ist ein selbständiges Werk, das sich an ein anderes Werk schwach anlehnt. Typische literarische Beispiele sind die Travestie und Parodie, die beide Satiren auf ein Originalwerk darstellen. Bei der freien Benutzung ist die Zustimmung des Urhebers des Originalwerkes nicht erforderlich.

In der EDV würde eine freie Benutzung dann vorliegen, wenn man z.B. eine in BASIC geschriebene Sortieroutine zum Anlaß nimmt, um eine eigene Routine etwa in Pascal oder Assembler zu erstellen.

Ein **Zitat** ist die Übernahme eines kompletten fremden Werkes oder eines Teils

eines fremden Werkes in ein eigenes Werk. Das Zitat muß exakt, d.h. bei Schriftwerken wörtlich sein (Änderungsverbot). Außerdem muß stets die Quelle (Urheber und Werkname) angegeben werden. Wie bei der freien Benutzung ist auch hier die Zustimmung des Urhebers des Originalwerkes nicht erforderlich.

Das *Großzitat* eines kompletten vollständigen Werkes setzt die Quellenangabe mit Verlagsangabe sowie das Erscheinen des fremden Werkes voraus. Ferner sind Großzitate nur in wissenschaftlichen (Sprach)werken zulässig, zu denen auch die Computerwerke gehören. Bild- und Musikzitate rechnet man in der Regel zu den Großziten (von Bedeutung bei Computerspielen).

Ein Großzitat läge z.B. dann vor, wenn man in einem Buch über ein Betriebssystem (etwa ProDOS) neben einer allgemeinen Beschreibung eine Kommentierung des ggf. zuvor disassemblierten Quellcodes vornähme (s. z.B. „ProDOS-Analyse“ von A. Schäpers).

Das *Kleinzitat* eines Teils eines fremden Werkes (= einer Stelle) setzt die Veröffentlichung des fremden Werkes voraus. Kleinzitate sind auch in nichtwissenschaftlichen Sprachwerken möglich.

Beispielsweise enthält das Kopierprogramm in meinem ProDOS-Buch, Band 2, die Formatierungsroutine aus dem ProDOS-FILER als Kleinzitat.

Das **Plagiat** ist die bewußte Aneignung fremden Geistesgutes, d.h. praktisch ein verändertes oder unverändertes Zitat, bei dem man die Quellenangabe „zufällig vergessen hat“. Dies ist stets unzulässig. In der Regel zulässige, wenngleich oft verpönte Spezialformen des Plagiats sind

1. das Zitat aus einem gemeinfreien (= nicht mehr geschützten) Werk ohne Quellenangabe;
2. das Zitat aus einem eigenen Werk (= Selbstplagiat);
3. das unbewußte Zitat infolge eines „eidetischen Gedächtnisses“;
4. die unbewußte Doppel- oder Parallelschöpfung.

Bei Büchern, Zeitschriften und Quellcodes kann man problemlos die Quellenangaben deutlich vermerken. Beim Objektcode kann die Quellenangabe entfallen, wenn der Quellcode mitgeliefert wird. Andernfalls muß man ihn im Objektcode selbst unterbringen. Bis zum heutigen Tage habe ich allerdings noch niemals eine Quellenangabe in einem Objektcode gesehen, was wohl nicht gerade auf ein ausgepräg-

tes Rechtsbewußtsein bei Programmierern schließen läßt.

### 3. Das „geplagte“ Plagiat

Das Kernproblem des Zitatrechts sind nicht die korrekten Kleinzitate, sondern die Zitate ohne Quellenangaben sowie die mehr oder weniger stark angelehnten Bearbeitungen von bereits vorliegenden Werken anderer Autoren. Nehmen wir einmal an, daß wir in einer Bücherstube ein Lyrikbändchen aufschlagen und dabei zufällig auf ein Gedicht stoßen, das so beginnt:

1 Über allen Bergen  
2 ist Ruh,  
3 in allen Särgen  
4 hörst du  
5 kaum einen Laut...

Nach dem ersten Durchlesen beginnt man zu stutzen, beim zweiten Durchlesen kommt dann das Déjà-vu-Erlebnis: Irgendwo habe ich dies doch schon einmal gesehen! Jetzt kommt das Aha-Erlebnis und man merkt, daß es sich hierbei um eine Verballhornung eines Goethe-Gedichtes handelt, das im Original so beginnt:

1 Über allen Gipfeln  
2 ist Ruh,  
3 in allen Wipfeln  
4 spürest du  
5 kaum einen Hauch...

Offenkundig handelt es sich weder um ein korrektes, wörtliches Zitat noch um ein für den Schulgebrauch („ad usum Delphini“) bereinigtes Gedicht. Somit bleibt nur noch entweder die einwilligungspflichtige Bearbeitung oder die zustimmungsfreie Benutzung. Die Zeile 2 („ist Ruh“) ist bei beiden Gedichten identisch, doch sind weder „ist“ noch „Ruh“ für sich noch deren Kombination schutzfähig. Alle anderen Zeilen unterscheiden sich in mindestens *einem* Wort, und ein gleicher Endreim findet sich nur in den Zeilen 2 und 4. Auch der Umstand, daß erstens die Anzahl der Wörter sowie sogar die Anzahl der Silben und zweitens das Versmaß identisch sind, begründet noch keinen Urheberschutz, denn ein Urheber kann weder Silben noch Wörter noch Reime noch Versform noch Satzbau exklusiv beanspruchen. Dies gilt sowohl für die gehobene Dichtung wie für die Alltagsprosa. Dem Autor des Gedichtes könnte man juristisch nichts am Zeug flicken. Trotzdem beschleicht uns hier ein unwohles Gefühl,

denn das gesunde Rechtsempfinden würde hier ein Plagiat vermuten. Daß dieses Empfinden überhaupt aufkommt, ist indessen nur darauf zurückzuführen, daß Goethes Gedicht allgemein bekannt ist.

Betrachten wir deshalb nach diesem fiktiven Beispiel einen „echten“ Fall. Angenommen wir würden in derselben Bücherstube nunmehr das Werk „H.Schierenbeck: Grundzüge der Betriebswirtschaftslehre, 5. Aufl. 1980“ auf der Seite 207 aufschlagen und folgendes lesen:

#### *Bei Schierenbeck*

(1) Deren Gegenstand bilden vor allem die zukünftige Entwicklung von Markt- und Absatzpotential, Markt- und Absatzvolumen sowie des Marktanteils einer Unternehmung.

(2) Das Marktpotential umschreibt die Aufnahmefähigkeit eines Marktes (Gesamtheit möglicher Absatzmengen) für ein bestimmtes Produkt.

(3) Das Absatzpotential ist der maximal mögliche Anteil am Marktpotential, den die Unternehmung auf sich vereinigen zu können glaubt...

(4) Infolge des nicht ausgeschöpften Marktpotentials ist es für die einzelne Unternehmung möglich, hohe Steigerungsraten des Absatzvolumens auch dann zu erzielen, wenn sich die Marktanteile nur unwesentlich verändern.

Wohl kaum einer würde hier ein ungutes Gefühl haben, weil sich kein Déjà-vu-Erlebnis einstellt. Deshalb helfe ich Ihnen auf die Sprünge. Vergleichen Sie nunmehr einmal die obigen Sätze mit den nachfolgenden, die ich der Seite 186 von „H.Meffert: Marketing. Einführung in die Absatzpolitik, 2. Aufl. 1977“ entnommen habe.

#### *Bei Meffert*

(1) Gegenstand von Absatzprognosen sind vor allem der zukünftige Zustand bzw. die Entwicklung von Markt- und Absatzpotential, Markt- und Absatzvolumen sowie des Marktanteils einer Unternehmung (Fuchs 1963, Rogge 1972).

(2) Marktpotential ist die Gesamtheit möglicher Absatzmengen eines Marktes für ein bestimmtes Produkt (Aufnahmefähigkeit eines Marktes).

(3) Absatzpotential ist der Anteil am Marktpotential, den das Unternehmen maximal erreichen zu können glaubt (Zielsetzung)...

(4) Infolge des nicht ausgeschöpften, rasch wachsenden Marktpotentials sind für die einzelne Unternehmung große Zu-



# Wollen Sie mit Ihrem **MACINTOSH** grafisch arbeiten?



**Das MacTablett von Summagraphics – mit entsprechender Software – macht es möglich.**

**Informieren Sie sich bei Ihrem Apple Händler, oder direkt unter Telefon 089/1415077**



**Summagraphics** <sup>®</sup>  
LIMITED

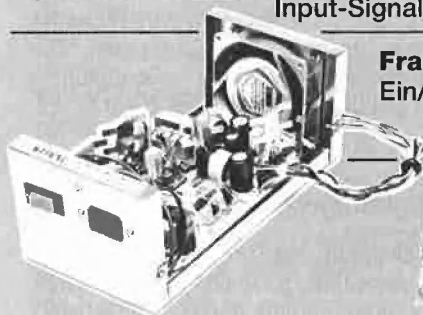
Niederlassung Deutschland

Georg-Brauchle-Ring 68  
D-8000 München 50  
Telefon 089/1415077  
Telex 5214793

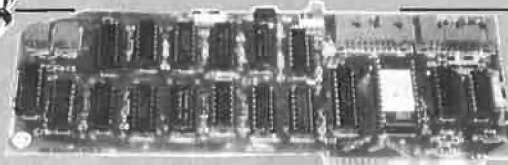


**7" Monitorchassis.** Auflösung 18 MHz. Grüner Schirm. Videoeingang. Lieferung kpl. steckfertig. Stromversorgung 12 V. **BN 94011 DM 58,50**  
Netztrafo 2 x 12 V/2 x 2 Amp. **BN 94012 DM 7,80**  
Monitorgehäuse 7-14" **BN 94015 DM 19,50**

**Monitor Grün 9".** Industriegehäuse. High-Resolution größer 20 MHz, Schirm grün, entspiegelt. Input-Signal 0,5-2,0 V. Stromversorgung 220 V/50 Hz. **BN 65032 Monitor VM-91/9 DM 199,90**



**Franklin-Schaltenteil.** Im geschlossenen Metallgehäuse mit eingebautem Lüfter. Ein/Aussch. u. Steckdose.  $\pm 5$  V/3 Amp. und  $\pm 12$  V/0,9 Amp. Applefähig. Maße 128 x 90 x 225 mm. **BN 94014 DM 69,50**

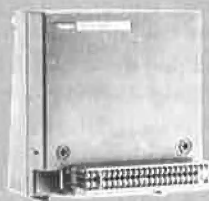


**Apple II und IIe.** Interface Card für 2 Disc Drive. Franklin Qualität. **BN 94014 DM 58,90**



**Trickball.** Handballen-Joystick für Commodore 20/64, Atari u. v. a. **BN 94031 Paarpreis DM 24,50**  
**BN 94032 ab 3 Paar DM 19,90**

**Kostenlosen Katalog anfordern!**



**Original Sinclair 16 K RAM.** Steck-Modul. Verwendbar nach Umbau auch für andere Computer. Bestückung: 8 x 4116 N-4/4 x 74 LS 157 N/1 x 74 LS 32/1 x 74 LS 00/1 x 74 LS 393. **BN 94033 DM 34,90**  
**BN 94034 ab 3 Stück DM 29,90**

**Kostenlosen Katalog anfordern!**

**Bühler Elektronik · Postfach 32 · 7570 Baden-Baden · Telefon (07221) 7 1004**

wachsraten im Umsatz auch dann erreichbar, wenn sich die Marktanteile nur unwesentlich verändern.

Bei wohlwollender Beurteilung der Sachlage würde man sagen, daß sich das Schierenbeck-Buch an das Meffert-Buch „geringfügig angelehnt“ habe. Bei nicht-wohlwollender Beurteilung würde man sagen, daß Schierenbeck „schamlos abgekupfert“ hat. Schierenbeck hätte eine ganze Seite aus dem Meffert-Buch wörtlich mit vollständiger Quellenangabe zitieren können. Dann wäre die Rechtslage eindeutig gewesen. So aber wurden die Sätze aus dem Meffert-Buch lediglich geringfügig umformuliert (und an einer anderen Stelle bei einer exakt reproduzierten Schemazeichnung der Ausdruck „nach Meffert“ hinzugefügt). Liegt damit bereits eine freie Benutzung vor? Wohl kaum, doch läßt sich nicht verleugnen, daß das hier praktizierte Verfahren in vielen wissenschaftlichen Büchern gang und gäbe ist. Neue Fachbücher entstehen heutzutage häufig dadurch, daß man Passagen aus früheren Büchern extrahiert, kondensiert und transformiert. Ironischerweise hat gerade die moderne Linguistik gezeigt, wie man hier rein schematisch vorgehen kann. Chomsky und die anderen Verfechter der generativen Transformationsgrammatik hätten hier frohlockt! Nehmen wir an, ein Ausgangsgedanke lautete:

„Wir schreiben jetzt diesen Satz.“

Daraus lassen sich dann auf dem Wege der Transformation u.a. die folgenden Sätze bilden:

1. Jetzt schreiben wir diesen Satz.
2. Dieser Satz wird jetzt von uns geschrieben.
3. Jetzt wird dieser Satz von uns geschrieben.
4. Von uns wird jetzt dieser Satz geschrieben.

Wenn man zusätzlich Synonyme und syntaktische Konstruktionen mit anderen Wörtern zuläßt, können ferner gebildet werden:

5. Nunmehr bringen wir diese Aussage zu Papier.
6. Der Satz muß nun von uns niedergeschrieben werden.  
usw.

Derselbe Gedanke läßt sich folglich in vielfältige mehr oder weniger gelenkte Sätze kleiden; neuer wird er dadurch jedoch nicht.

Wenn derartige syntaktische Transformationen und Synonym-Substitutionen bereits ein neues Sprachwerk begründen,

dann haben wir es bei Computerwerken noch viel einfacher. So läßt sich etwa der „Computergedanke“

LDA \$2000 STA \$4000

folgendermaßen transformieren:

1. LDX \$2000 STX \$4000
2. LDY \$2000 STY \$4000
3. LDX \$2000 TXA STA \$4000

Man brauchte dann also bei einem fremden Computerwerk lediglich Registervertauschungen und Befehlsumstellungen und anschließend eine Neuassemblierung mit geänderter Ursprungsadresse vorzunehmen, so daß beispielsweise  
LDA \$2000 STA \$4000

in

LDX \$2222 STX \$4222

transformiert wird mit der Folge, daß (scheinbar) ein neues selbständiges Werk entsteht.

Bei den obigen Zitaten aus den Büchern von Schierenbeck und Meffert ist zu unterscheiden zwischen den sog. definitivischen oder analytischen Aussagen (2 und 3) und den nicht-definitivischen oder synthetischen Aussagen (1 und 4). Bei analytischen Aussagen ist der Formulierungsspielraum begrenzt, weil der korrekt definierte Begriff die Verwendung bestimmter Termini erzwingt. Bei synthetischen Aussagen ist der Formulierungsspielraum jedoch sehr groß. Deshalb springt beispielsweise der Satz (4) sofort als „angelehnt“ ins Auge.

Im Gegensatz zu natürlichen Sprachwerken ist bei Computerwerken der Spielraum für Variationen teils enger und teils weiter gezogen: weiter insofern, als künstliche Umformungen in der oben skizzierten Art leicht vorgenommen werden können, enger insofern, als Algorithmen eine relativ rigide Programmierung erfordern.

#### 4. Das Kaspar-Hauser-Syndrom

Goethe gilt bekanntermaßen als unser größter Dichter, der seiner Nachwelt ein beeindruckendes Gesamtwerk hinterlassen hat. Demgegenüber ist weniger bekannt, daß er eine riesige Gelehrtenbibliothek mit angeblich über 5000 Bänden besaß. Sie werden sich fragen, was das eine mit dem anderen zu tun hat. Hier kommen wir jedoch zur Begründung zu dem oben zitierten Satz (4): Hätte Schierenbeck niemals das Meffert-Buch gelesen, dann wäre er wohl auch niemals zu dem „angelehnten“ Satz gekommen. Zu Anfang des letzten Jahrhunderts, also noch zu Lebzeiten Goethes, vegetierte ein ausgesetztes

Findelkind namens Kaspar Hauser sprichwörtlich in einem Kellerloch dahin, bis es eines Tages entdeckt wurde. Es wurde zwar mehr schlecht als recht ernährt, hatte jedoch keinen nennenswerten (Sprach-)kontakt zu seiner Umwelt, geschweige denn zu seinen Eltern aus den höchsten Kreisen des Adels, die sich auf diese Weise eines Erben entledigen wollten. Es ist einleuchtend, daß Hauser im Gegensatz zu Goethe niemals ein Werk geschrieben hat. Nun wäre es jedoch falsch, annehmen zu wollen, daß Goethe seine Werke geschrieben hat, weil er 5000 Bücher besaß. Richtig ist vielmehr, daß Goethe niemals ein Buch hätte schreiben können, wenn er niemals ein Buch gelesen hätte. Die mehr oder weniger umfangreiche Verarbeitung früheren Geistesgutes ist damit eine *notwendige*, aber keine *hinreichende* Bedingung für die Schaffung eigener Werke. (Für Logiker: die notwendige Bedingung ist eine Replikation, die hinreichende eine Implikation.) Jeder, der ein Werk schaffen will, sei er nun Dichter oder Programmierer, muß also notwendigerweise in bestimmtem Umfang auf das Schaffen anderer Urheber zurückgreifen. Rechtlich relevant ist jedoch allein die hinreichende Bedingung. Wenn es einem Urheber nicht gelingt, die letzte Sprosse der Schöpfungshöhe zu erklimmen, so sollte er sich seine Unzulänglichkeit eingestehen und zum wörtlichen Kleinzitat greifen, das im Rahmen eines wissenschaftlichen Sprachwerkes, wozu auch die Computerwerke gehören, stets erlaubt ist. Wie der obige Satz (4) zeigt, ist es mit der Substitution von „groß“ durch „hoch“ und dem Einfügen von Flickwörtern wie „rasch“ usw. nicht getan. Zur „persönlichen geistigen Schöpfung“ (UrhG §2) gehört leider etwas mehr.

#### 5. Folgerungen für Computerwerke

Das Zitatrecht ist urheberrechtlich für Computerwerke deshalb relevant, weil Computerprogramme den (wissenschaftlichen) Schriftwerken oder allgemeiner formuliert den Sprachwerken zugeordnet werden. Die Zitatparagraphen kommen indessen erst dann zum Tragen, wenn eigene Programme mit fremden Programmteilen *erscheinen*, also vervielfältigt und verbreitet werden (verkaufen, verschenken, tauschen). Solange man lediglich in seinem stillen Kämmerlein fremde Programme in eigene einarbeitet, kann das Urheberrecht noch nicht verletzt werden. Selbst wenn man es mit dem Zitieren in wissenschaftlichen Werken oftmals nicht

allzu genau nimmt, wie die Gegenüberstellung aus den Büchern von Schierenbeck und Meffert gezeigt hat, empfehle ich dem Pecker-Leser, im Zweifelsfalle wörtlich mit korrekter Quellenangabe zu zitieren, zumal das Kleinzitat vergütungsfrei ist. Auch wenn augenscheinlich künstliche Transformationen eines fremden Werkes rein rechtlich ausreichen, um ein neues Werk zu schaffen, ist dieses Verfahren ethisch mehr als fragwürdig.

Vorsicht ist stets bei Großzitatat geboten, weil hier die Grenzen enger gezogen sind. So ist es beispielsweise in einem Buch über ProDOS, das dieses Betriebssystem bis zum letzten Byte kommentiert, ohne weiteres zulässig, ProDOS in Form eines Großzitatats in den Text zu integrieren. Umgekehrt wäre es nicht zulässig, auf eine Diskette mit einem beliebigen Apple-Programm ProDOS oder DOS 3.3 mit aufzunehmen, weil hierfür zitatrechtlich keine

Veranlassung besteht. (Nachtrag: Die Schlagzeile „Ave auctor, plagiarum te salutant“ = „Heil Dir Autor, die Plagiatoren grüßen Dich“ ist in Analogie zu dem Gladiatorenspruch „Ave imperator, morituri te salutant“ = „Heil Dir Imperator, die Todgeweihten grüßen Dich“ entstanden.)



## Urheberrechtsnovelle

Am 27.06.1985 ist das „Gesetz zur Änderung von Vorschriften auf dem Gebiet des Urheberrechts“ im „Bundesgesetzblatt“ Nr. 33 veröffentlicht worden, das nunmehr erstmals konkret von „Programmen für die Datenverarbeitung“ spricht, die den Sprachwerken zugeordnet werden. Nachfolgend geben wir einen Auszug aus den geänderten Paragraphen des UrhG, soweit sie für EDV-Programme relevant sind. Man beachte, daß die Zitatrechtparagraphen nicht geändert wurden.

§2 Abs. 1 Nr. 1: (Zu den geschützten Werken ... gehören insbesondere:)

1. Sprachwerke, wie Schriftwerke und Reden, sowie Programme für die Datenverarbeitung.

§53 Vervielfältigungen zum privaten und sonstigen eigenen Gebrauch

(1) Zulässig ist, einzelne Vervielfältigungsstücke eines Werkes zum privaten Gebrauch herzustellen. Der zur Vervielfältigung Befugte darf die Vervielfältigungsstücke auch durch einen anderen herstellen lassen; doch gilt dies für die Übertragung von Werken auf Bild- oder Tonträger ... nur, wenn es unentgeltlich geschieht.

(2) Zulässig ist, einzelne Vervielfältigungsstücke eines Werkes herzustellen oder herstellen zu lassen

1. zum eigenen wissenschaftlichen Gebrauch, wenn und soweit die Vervielfältigung zu diesem Zweck geboten ist,

2. zur Aufnahme in ein eigenes Archiv, wenn und soweit die Vervielfältigung zu diesem Zweck geboten ist und als Vorlage für die Vervielfältigung ein eigenes Werkstück benutzt wird,

...

(4) Die Vervielfältigung

...

b) eines Buches oder einer Zeitschrift, wenn es sich um eine im wesentlichen vollständige Vervielfältigung handelt, ist, soweit sie nicht durch Abschreiben vorgenommen wird, stets nur mit Einwilligung des Berechtigten zulässig oder unter den Voraussetzungen des Absatzes 2 Nr. 2 oder zum eigenen Gebrauch, wenn es sich um ein seit mindestens zwei Jahren vergriffenes Werk handelt. *Ebenso ist die Vervielfältigung eines Programms für die Datenverarbeitung (§2 Abs. 1 Nr. 1) oder wesentlicher Teile davon stets nur mit Einwilligung des Berechtigten zulässig.*

(5) Die Vervielfältigungsstücke dürfen weder verbreitet noch zu öffentlichen Wiedergaben benutzt werden...

§108a: Gewerbsmäßige unerlaubte Verwertung

(1) Handelt der Täter in den Fällen des Vervielfältigens oder des Verbreitens im Sinne des §106 oder des §108 gewerbsmäßig, so ist die Strafe Freiheitsstrafe bis zu fünf Jahren oder Geldstrafe.

(2) Der Versuch ist strafbar.

### Rechtsprechung zum Software-Urheberrecht

Unter diesem Titel erscheint im nächsten Pecker ein Beitrag von RA D. Naber, der die Gerichtsurteile der letzten Zeit resümiert.

### Lotus und Urheberrecht

In der amerikanischen Zeitschrift „Byte“ wurde von mehreren Prozessen der Firma Lotus gegen Käufer von „1-2-3“ usw. berichtet. Wegen der Herstellung illegaler Kopien sollen angeblich Konventionalstrafen in Millionenhöhe gezahlt worden sein. Da die Firma Lotus jetzt auch eine Niederlassung in Deutschland hat, empfehle ich deshalb allen Lesern, den Kaufvertrag durch den Passus zu erweitern, daß nur das neue deutsche Urheberrechtsgesetz gilt und davon abweichende Vereinbarungen null und nichtig sind. Die Empfehlung gilt im übrigen nicht nur in bezug auf Lotus, sondern im Hinblick auf alle Software-Verträge. Es ist Ihr gutes Recht, wenn Sie sich das ausbedingen, was im Gesetz steht. us

# Ein neues FID für DOS 3.3

## Mit Auto-FID für RAM-Disk-Besitzer

von Harald Grumser

Ein im Umgang mit Disketten immer wieder auftretendes Problem ist die Erstellung von Dateikopien, sei es, um die Folgen eines „Disketten-Crashes“ durch sog. Backups zu mindern oder um alle wichtigen Arbeitsdateien auf eine RAM-Disk zu übertragen.

Eine Automatisierung dieses Vorgangs war bisher nur unter Zuhilfenahme des FID-Programms (File Developer) möglich, wobei der recht umständliche Weg über eine EXEC-Datei gewählt werden mußte. Das hier vorgestellte Programm zeigt, wie man aus einem Applesoft-Programm heraus Dateien aller Art (A-, I-, B- und T-Files) kopieren kann, ohne daß dabei Probleme wie Garbage-Collection oder „EXTRA IGNORED“-Meldungen speziell bei der Übertragung von Textdateien auftreten. Es bietet sich somit auch die Möglichkeit, die RAM-Disk durch ein geeignetes HELLO-Programm als Arbeitsdiskette vorzubereiten und nach Beendigung einer Sitzung alle wichtigen Daten per Programm auf eine physische Diskette zu retten.

### 1. Anforderungen

Bei der Erstellung dieses Programms wurden folgende Ziele ins Auge gefaßt:

**Kompatibilität** – Da mittlerweile etliche DOS-Varianten existieren, die z.T. erhebliche Verbesserungen gegenüber dem Standard-DOS-3.3 zeigen, wurde auf die Benutzung aller nicht über die Page 3 zugänglichen DOS-Routinen verzichtet. Somit läuft dieses Programm auch unter Diversi-DOS und gemovten DOS-Versionen.

**Flexibilität** – Es versteht sich von selbst, daß sowohl der Betrieb mit einem Laufwerk als auch die Ansteuerung von Laufwerken in verschiedenen Slots unterstützt wird. Die Möglichkeit, unterschiedliche

Quell- und Zielnamen anzugeben, vereinfacht die Anwendung in vielen Fällen, da dadurch auch Dateien auf *einer* Diskette dupliziert werden können. Der Umfang des Datenpuffers kann vom Benutzer gewählt werden.

**Einfache Handhabung** – Der Aufruf des Kopierprogramms erfolgt mit Hilfe des Ampersand-Vektors. Die zuletzt angegebenen Laufwerksparameter bleiben als Ersatzwerte erhalten, wodurch Slot und Drive nicht immer wieder neu eingegeben werden müssen.

**Neutralität** – Da diese Utility in Anwenderprogramme eingebaut werden kann, besteht die Möglichkeit, alle Meldungen (Fehleranzeige, Aufforderung zum Diskettenwechsel usw.) abzufangen und durch eigene Menütexe oder Bildschirmmasken zu ersetzen. Dies wurde durch die Übergabe aller Meldungen als Applesoft-Fehler realisiert, die durch „ONERR GOTO“ bearbeitet werden können. Die Programmierung gestaltet sich somit ähnlich wie bei DOS-Befehlen in umfangreicheren Anwenderprogrammen.

Darüber hinaus besteht die Möglichkeit, das Programm auch im Direktmodus aufzurufen.

### 2. Der Programmstart

Vor der Erläuterung der Interna soll zunächst die Initialisierung und der Aufruf des Programms, das **AS.FILER** (= Applesoft-Filer) genannt wurde, beschrieben werden. Das Programm wird durch BRUN AS.FILER gestartet und setzt als Ersatzwerte für Quell- und Ziellaufwerk die Parameter des Startlaufwerks ein. Der Benutzer muß danach den Pufferumfang durch Angabe des HIMEM-Wertes festlegen; dieser Puffer bestimmt die Datenmenge, die auf einmal in den Rechner eingelesen

werden kann, und sollte, um mehrfache Kopierschübe zu vermeiden, so groß wie möglich gewählt werden. Im Direktmodus kann HIMEM z.B. auf 4096 (\$1000) gesetzt werden, d.h. je tiefer die angegebene Adresse, desto größer wird der Puffer. Im Programm sollte dieser Wert jenseits der Variablenobergrenze liegen und noch etwas „Luft“ für Strings lassen (siehe die beiden Beispielprogramme). Das Programm selbst beginnt im Speicher ab \$8EA4 (bzw. \$8ED8, da die Initialisierungsroutine überschrieben werden kann) und bildet die Obergrenze des Datenpuffers.

### 3. Der Aufruf

Der eigentliche Aufruf des Kopierprogramms erfolgt dann durch den Befehl „& Quelldatei, Zieldatei“, wobei die beiden Dateiangaben als Strings angegeben werden müssen. Die Dateiangabe besteht aus dem Dateinamen und den durch Komma getrennten Laufwerksparametern. Wird die Laufwerksangabe weggelassen, so setzt das Programm die zuletzt benutzten Werte ein. Die Laufwerksangabe besteht aus Slot und Drive (die Reihenfolge spielt dabei keine Rolle) und muß nicht vollständig sein, d.h., daß z.B. der Slot nicht immer bestimmt werden muß.

Die Angabe der Zieldatei kann entfallen; in diesem Fall wird der Quellname und das zuletzt angegebene Ziellaufwerk eingesetzt. Das Komma zwischen Quell- und Zieldatei darf in diesem Fall nicht getippt werden. Soll der Quellname benutzt werden, jedoch ein anderes Ziellaufwerk, so kann die Zieldateiangabe nur aus der Bestimmung des Laufwerks bestehen, wobei jedoch aus syntaktischen Gründen ein Komma vorangehen muß.

Um dieser formalen Beschreibung etwas Farbe zu geben, sind im folgenden einige

Beispiele mit Erläuterungen aufgelistet. Dabei seien folgende Eingaben vorangegangen:

```
BRUN AS.FILER,S6,D1
HIMEM: 3000
N$ = "DATEI"
DRIVE = 2
```

& "ORIGINAL", "FAELSCHUNG,D2" – kopiert die Datei ORIGINAL von Laufwerk S6, D1 als Datei FAELSCHUNG nach S6, D2.

& "TEXT" – kopiert die Datei TEXT von Laufwerk S6, D1 unter gleichem Namen nach S6, D2.

& "PROGRAMM,D2", "POGRAMM.1" – kopiert die Datei PROGRAMM von Laufwerk S6, D2 mit dem Namen PROGRAMM.1 auf das gleiche Laufwerk.

& N\$ + ",S3", ",D1,S3" – kopiert die Datei mit Namen DATEI von Laufwerk S3, D2 mit demselben Namen nach S3, D1.

& N\$ + ".COPY", "DATEI.BAK,D" + STR\$ (DRIVE) – kopiert die Datei namens DATEI.COPY von Laufwerk S3, D2 mit dem Namen DATEI.BAK auf dasselbe Laufwerk.

Es ist zu beachten, daß Leerzeichen nur innerhalb des Dateinamens stehen dürfen.

#### 4. Die Fehlermeldungen

Wenn während der Übertragung mit nur einem Laufwerk ein Diskettenwechsel notwendig wird, meldet sich das Programm mit einer Fehlermeldung und Fehlernummer. Nach dieser Meldung kann mit dem &-Befehl fortgefahren werden (es versteht sich, daß die Diskette vorher gewechselt werden muß). Besteht die Zieldatei bereits oder ist sie gesperrt, so wird ebenfalls eine Warnung in Form eines Fehlers ausgegeben. In diesem Fall kann durch „& Name“ ein neuer Zielname angegeben oder durch einen einfachen &-Befehl die alte Datei überschrieben werden.

Es gibt jedoch auch einige weniger erfreuliche Fehlermeldungen, die die Kopie abbrechen. In der **Tabelle der Fehlermeldungen und Hinweise** sind diese zusammengestellt.

Daraus wird auch ersichtlich, daß eine Kopie bei jeder Meldung durch die Eingabe von „&“ abgebrochen werden kann.

Diese Verfahrensweise ermöglicht zwar eine große Flexibilität, es ist jedoch davor zu warnen, die Antwort auf eine Meldung erst nach mehreren anderen Aktionen zu

geben. Dagegen kann während einer Kopie durchaus mittels „CATALOG“ die Existenz von Dateien nachgeprüft werden (z.B. zur Festlegung eines Ersatz-Zielnamens)

Innerhalb eines Programmablaufs sollte auf die Meldungen durch „ONERR GO TO“-Konstruktionen eingegangen werden. Dies geschieht auf die übliche Weise; der Fehlercode kann wie bisher durch PEEK (222) ermittelt werden. Näheres hierzu ist aus den beiden Beispielprogrammen zu entnehmen.

Der Kopiervorgang sei noch einmal an einem kurzen Dialog verdeutlicht, bei dem die (sehr lange) Datei XXX mit nur einem Laufwerk kopiert werden soll:

```
& "XX,S6,D1", ",S6,D1" <RETURN>
#6 ERROR (bedeutet: vertippt – Datei existiert nicht)
& "XXX" <RETURN>
#2 ERROR (bedeutet: Zieldiskette einlegen)
& <RETURN>
#4 ERROR (bedeutet: Datei existiert bereits)
CATALOG <RETURN> (neuen Dateinamen suchen)
& "YYY" <RETURN>
#1 ERROR (bedeutet: Quelldiskette einlegen)
& <RETURN>
#2 ERROR (bedeutet: erneut Zieldiskette einlegen)
& <RETURN>
```

#### 5. Technische Details

Der Aufruf des File-Managers wurde schon im Peeker, Heft 5/85 beschrieben und soll nur noch einmal kurz zusammengefaßt werden. Zum Einlesen einer Datei muß diese zunächst *eröffnet* werden. Dabei wird deren Existenz überprüft und der (erste) TSL-Sektor eingelese. Bevor nun die Daten gelesen werden können, ist die Datei zu positionieren, d.h. ein interner Dateizeiger wird im allgemeinen auf den Dateianfang gesetzt. Nach dieser Positionierung kann der Inhalt der Datei in den Rechner gelesen werden.

Aufgabe des Programmierers ist die Bereitstellung des Speicherbereiches, der die Daten aufnehmen soll. Da Daten stets nur sektorweise (1 Sektor = 256 Bytes) eingelese werden, muß ein Puffer bereitgestellt werden, der darüber hinaus auch den TSL-Sektor aufnehmen muß. Weiterhin benötigt der File-Manager einen Puf-

fer, um alle im Zusammenhang mit dieser Datei relevanten Daten abzulegen (dies ist die Voraussetzung für die gleichzeitige Bearbeitung mehrerer Dateien).

Paßt die eingelesene Datei nicht auf einmal in den Speicher, so bricht der File-Manager den Lesevorgang ab. An dieser Stelle kann nun der gepufferte Dateinhalt auf die Zieldatei geschrieben werden, die auch zunächst eröffnet und positioniert werden muß. Bei der Eröffnung der Zieldatei muß die Möglichkeit bestehen, diese neu anzulegen. Dies wird durch das sog. Allocation-Flag (X-Register) mitgeteilt. Nachdem der Puffer auf die neue Datei geschrieben wurde, kann in einem weiteren Durchlauf der nächste Teil des Files eingelese werden, um auch diesen zu übertragen. Das Ende der Datei ist erreicht, wenn beim Einlesen ein „END OF DATA“-Fehler auftritt. Nach dem letzten Beschreiben der Zieldatei muß diese geschlossen werden. Dabei wird die TSL auf der Diskette aktualisiert und der letzte Datensektor als belegt gekennzeichnet. Die Quelldatei muß im Gegensatz zum DOS-Aufruf auf Applesoft-Ebene nicht geschlossen werden (Dieses Schließen im Applesoft-Programm bewirkt nur, daß der Dateipuffer freigegeben wird).

Beim Lesen und Schreiben wird die Länge des Speicherbereichs in der File-Manager-Parameter-Liste übergeben. Es ist dabei zu berücksichtigen, daß die Länge beim Schreiben um 1 vermindert werden muß.

#### 6. Programm-Interna

Das Programm AS.FILER untergliedert sich in vier Teile:

- Im Initialisierungsteil wird der &-Vektor gesetzt und die Lage der File-Manager-Parameter-Liste und des Input/Output-Blocks über eine Page-3-Routine ermittelt. Danach werden die Ersatzwerte für Slot und Drive aus dem IOB übernommen und ein Status-Register auf Null gesetzt. Dieses Register gibt während des Kopiervorgangs den Zustand an, um nach einem Wiederaufruf (z.B. nach Diskettenwechsel) an der richtigen Stelle fortfahren zu können.

- Der zweite Teil bildet die Benutzerschnittstelle. Nach Aufruf durch den Ampersand-Vektor werden hier zunächst die Übergabeparameter ausgewertet; die Kontrolle der Syntax richtet sich nach dem derzeitigen Status. Bestehen keine Bean-

standungen, so wird in den entsprechenden Teil der Hauptroutine verzweigt.

● Im Hauptteil erfolgt die eigentliche Kopie. Er besteht im wesentlichen aus wiederholten Aufrufen des File-Managers, wobei dessen Fehlermeldungen den Ablauf regeln. Da diese Routine mehrfach für einen Kopiervorgang aufgerufen werden kann, müssen einige Werte gesichert werden:

Zunächst wird ermittelt, ob nur ein Laufwerk vorliegt (SAMESD) um vor dem Umschalten einen Diskettenwechsel zu ermöglichen. Außerdem muß festgelegt werden, ob beim Lesen eine „END OF DATA“-Meldung auftrat, um zum einen den Pufferumfang entsprechend zu verkleinern und zum anderen den letzten Durchgang zu markieren (LSTPASS), da nach dem letzten Schreiben die Zieldatei geschlossen werden muß und somit die Kopie abgeschlossen ist. Darüber hinaus muß durch die Variable OPNFLG sichergestellt werden, daß die Zieldatei beim zweiten Schreibdurchgang nicht nochmals eröffnet wird.

Eine detaillierte Schilderung dieser Routine würde hier zu weit führen; es sei daher auf die reichhaltigen Kommentare im Programm verwiesen.

● Der vierte Teil beinhaltet die Unterprogramme zum Aufruf des File-Managers und der Auswertung des Übergabestrings. Diese Routinen zeichnen sich nicht durch eine gesteigerte Durchsichtigkeit aus, weil versucht wurde, den Code so kurz wie möglich zu halten.

Die Parameterauswertung gestaltet sich zum einen wegen der vielen optionalen Angaben etwas umständlich, zum anderen muß die Integrität des Namens (dies beinhaltet auch die maximale Länge von 30 Zeichen) sichergestellt werden.

Vor der Beschreibung der Beispielprogramme noch einige allgemeine Erläuterungen zum Programm:

Die Variable PAGES legt die Anzahl der minimalen Pufferseiten fest. Umfaßt der Speicherbereich zwischen HIMEM und dem COPY-Programm weniger als die in dieser Variablen festgelegten Seiten, so erfolgt eine „OUT OF MEMORY“-Fehlermeldung.

Bricht das Programm wegen eines „DISK FULL“-Fehlers ab, muß die Zieldatei in jedem Fall geschlossen werden, da sonst beim Löschen dieser Datei die belegten

Sektoren in der VTOC nicht mehr freigegeben würden.

Das Lockbit im Filetyp-Byte wird stets zurückgesetzt, da das Programm ansonsten eine Überschreibung der neu eröffneten (geschlossenen) Datei verweigern würde. Vor dem Überschreiben einer existenten Datei wird diese zuerst gelöscht, da sonst die alte Länge und der Filetyp erhalten bliebe.

Das Programm verwendet keine DOS-Puffer, sondern stellt selbst ca. 1200 Bytes unterhalb von \$9600 bereit.

Unter Standard-DOS kann sich die hier vorgestellt Utility geschwindigkeitsmäßig nicht mit FID messen, da FID im Gegensatz zu diesem Programm nicht „seriös“ programmiert wurde und deshalb z.B. nicht mit gemovtem DOS ohne Patch läuft. Benutzt man hingegen z.B. Diversi-DOS, so halten sich die Übertragungsgeschwindigkeiten in etwa die Waage.

## 7. STARTUP zum Kopieren von Diskette auf RAM-Disk

Beim Kaltstart des Rechners ist es wünschenswert, wenn eine RAM-Disk installiert wird und alle wichtigen Dateien dort hin übertragen werden. Das Programm STARTUP demonstriert eine Möglichkeit mit der Poor Man's RAM-Disk aus Peeker, Heft 1-2/85, S. 8; es kann jedoch jede andere RAM-Disk benutzt werden, deren Initialisierung als Unterprogramm aufgerufen werden kann (Zeile 120 und 130 sind dann zu ändern). Die hier verwendeten POKEs verhindern einen DOS-Warmstart und das Löschen des Bildschirms; der Einsprung erfolgt dann hinter dem Installierungsmenü.

Nach dem Starten des AS.FILER wird HIMEM auf den niedrigsten Wert (Variablenende + 7 Strings für die Dateinamen) gesetzt, um einen maximal großen Puffer zu erhalten. Die zu übertragenden Dateien werden in einer DATA-Zeile festgehalten und nacheinander kopiert. Da beim Kaltstart des Rechners nicht allzuvielen Fallunterscheidungen nötig sind, reduziert sich die Fehlerbehandlung auf wenige Zeilen. Erwähnenswert ist die Zeile 340, die in jeder Fehlerbehandlungsroutine nach „ONERR GOTO“ eine ganz normale Fehlerausgabe produziert. Dieses Verfahren bietet sich beim Austesten von Programmen an, wenn nicht sichergestellt ist, ob alle möglichen Fehler abgefangen werden (was in dem Programm STARTUP nicht der Fall sein sollte).

Tritt bei der Übertragung ein „DISK FULL ERROR“ auf (die Poor Man's RAM-Disk umfaßt nur 61 reine Datensektoren), so wird die letzte Datei gelöscht, da sie ohnehin nicht komplett kopiert wurde und somit nutzlos wäre.

## 8. ENDUP zum Kopieren von RAM-Disk auf Diskette

Der Inhalt einer RAM-Disk geht nach dem Ausschalten des Rechners verloren – es sei denn, die Dateien werden auf eine physische Diskette kopiert, was insbesondere für die eigentlichen Datendateien erforderlich ist. Eine Automatisierung dieses Vorgangs wäre wünschenswert.

Das Programm ENDUP zeigt ein weiteres Beispiel für die Anwendung des AS.FILER. Hierbei wird im Gegensatz zum FID-Programm automatisch ein Suffix angehängt, wenn eine Datei gleichen Namens auf der Zieldiskette bereits existiert. Das Programm besteht im wesentlichen aus drei Teilen. Das Hauptprogramm (Zeile 1000 ff.) regelt die Übertragung der Dateien und die Bearbeitung der Fehlermeldungen und Warnungen. Das Unterprogramm ab Zeile 2000 liest aus der Catalog-Spur der Diskette bei jedem Aufruf einen Dateinamen ein und übergibt diesen an das Hauptprogramm. Es bedient sich dabei des Unterprogramms ab Zeile 3000, das den angegebenen Sektor in den Eingabepuffer einliest.

Im Hauptprogramm werden zunächst die Laufwerksparemeter von Quell- und Zieldiskette abgefragt. Nach Angabe dieser Werte läuft die komplette Übertragung automatisch ab, wobei die Namen der kopierten Dateien ausgegeben werden. Besteht auf der Zieldiskette bereits eine Datei mit demselben Namen, so wird der Zieldatei solange eine „.COPY“ angehängt, bis ein noch nicht existierender Name entsteht. (Dieses Verfahren gerät jedoch in eine Endlos-Schleife, wenn der Dateiname eine Länge von 30 Zeichen überschreitet.)

Das Programm kann selbstverständlich auch genutzt werden, um Dateien zwischen zwei physischen Laufwerken zu transferieren. Soll eine selektive Übertragung erfolgen, so kann in Zeile 2145 der Dateityp durch die Anweisung TYP = PEEK (514 + IND) ermittelt werden; zur Kopie reiner Applesoft-Dateien ist dann folgende Zeile einzufügen:

```
1215 IF TYP <> 2 THEN 1200
```

(zu den Dateitypen s. Peeker, Heft 5/85, Seite 9: DOS-File-Manager).

Obwohl die Kopie einer Diskette auf sich selbst möglich wäre, sei vor diesem Spiel gewarnt. Solange weniger als 7 Dateien zu übertragen sind, findet das Programm ein Ende, wenn nie mehr als 6 Dateien auf der Diskette existierten. Ansonsten wird die erste Datei mit dem Anhängsel „.COPY .COPY“ so lange kopiert, bis ein „DISK FULL ERROR“ auftritt.

Das Unterprogramm, das jeweils den nächsten Catalog-Eintrag holt, liest beim ersten Aufruf den ersten Catalog-Sektor in den Eingabepuffer ein. Danach wird ein Index auf den ersten Eintrag gebildet und der Dateiname herausgepeekt. Dabei muß von hinten das erste Nicht-Leerzeichen gesucht werden, da ansonsten jeder Name 30 Zeichen umfassen würde und ein Suffix bedeutungslos wäre. Um Konflikte mit der internen String-Darstellung des Applesoft-Interpreters zu vermeiden (DOS 3.3 legt die Dateinamen mit Bit 7 = 1 ab, während der Interpreter und der AS-.FILER positive ASCII-Werte bearbeitet), wird durch Subtraktion von 128 das Bit 7 gelöscht.

Beim nächsten Aufruf dieses Unterprogramms wird der Index um 35 erhöht und ein weiterer Name ausgewertet. Dabei muß geprüft werden, ob dieser Eintrag nicht gelöscht wurde (Zeiger zur ersten Track/Sektor-Liste enthält \$FF), um ihn gegebenenfalls zu übergehen. Dieses Byte wird auch zur Erkennung des Catalog-Endes herangezogen: ein nie referiertes Feld enthält hier eine Null. Die Variable ERR wird dann vor dem Rücksprung in das Hauptprogramm auf 1 gesetzt. Überschreitet der Index die Sektorgrenze, so wird der nächste Catalog-Sektor eingelesen, falls es sich nicht schon um Sektor 1 der Catalog-Spur handelt.

Das Einlesen eines Sektors mit Hilfe eines (fast) reinen Applesoft-Programms ist wenig geeignet, um den hochsprachlichen Charakter des BASIC zu demonstrieren – das hier gezeigte Unterprogramm kann dennoch in vielen Fällen nützlich sein.

Vom Hauptprogramm müssen die Sektorparameter (SLOT, DRIVE, TRACK, SECTOR) übergeben werden. Beim Aufruf wird zunächst die Lage des IOB und der RWTS bestimmt. Da auch hier jegliche Kompatibilität gewahrt werden soll, ist die Adresse des IOB nur durch die umständliche Konstruktion mit den indirekten PEEKs möglich. Bevor die RWTS aufgerufen wird, müssen die Parameter in diesen

Kontrollblock übertragen werden. (Auf die Bedeutung der einzelnen POKES soll hier nicht weiter eingegangen werden).

Beim Sprung in die RWTS müssen das A- und Y-Register die Adresse des IOB enthalten; daher kann nur über ein kurzes gepoktes Maschinenprogramm dorthin gesprungen werden.

Vor der Rückkehr in das aufrufende Programm wird letztendlich ein Fehlercode in die Variable ERR eingelesen, wobei ERR = 0 den erfolgreichen Lesevorgang signalisiert.



#### AS.FILER Hex-Dump

BSAVE AS.FILER, A\$8EA4, L1884

```

$8EA0: 00 00 00 00 A9 D8 A0 8E
$8EA8: 8D F6 03 8C F7 03 20 DC
$8EB0: 03 84 ED B5 EE 20 E3 03
$8EB8: 84 EB 85 EC A0 00 8C 65
$8EC0: 91 C8 B1 EB 4A 4A 4A 4A
$8EC8: 8D 4B 91 8D 59 91 C8 B1
$8ED0: EB 8D 4A 91 8D 58 91 60
$8ED8: C9 2C D0 06 20 B1 00 4C
$8EE0: E5 8F AE 65 91 F0 4A E0
$8EE8: 01 D0 06 20 26 8F 4C 8A
$8EF0: 8F E0 02 D0 06 20 26 8F
$8EF8: 4C AB 8F E0 03 D0 0F 20
$8F00: B7 00 D0 19 A2 0E A9 08
$8F08: 20 E2 90 4C 13 8F 20 B7
$8F10: 00 D0 0A A2 0E A9 05 20
$8F18: E2 90 4C B0 8F 20 03 90
$8F20: 20 26 8F 4C B0 8F 20 B7
$8F28: 00 D0 01 60 A9 05 4C 2C
$8F30: 91 A9 00 8D 67 91 8D 66
$8F38: 91 38 A9 D8 E5 73 85 FA
$8F40: A9 8E E5 74 85 FB 90 04
$8F48: E9 01 B0 03 4C 10 D4 20
$8F50: EB 8F AD 4B 91 CD 59 91
$8F58: D0 06 AD 4A 91 ED 58 91
$8F60: 8D 68 91 A2 01 8E 69 91
$8F68: CA 20 E0 90 F0 03 4C 30
$8F70: 91 A0 07 B1 ED 29 7F 8D
$8F78: 5A 91 A2 00 20 D1 90 F0
$8F80: 09 AD 68 91 D0 04 A9 01
$8F88: D0 3F 20 99 90 F0 13 8D
$8F90: 67 91 18 A0 06 A5 FA F1
$8F98: ED 85 FA C8 A5 FB F1 ED
$8FA0: 85 FB AD 68 91 D0 04 A9
$8FA8: 02 D0 1E AD 66 91 D0 27
$8FB0: A2 00 8E 69 91 A2 0E 20
$8FB8: E0 90 A0 07 B1 ED 10 04
$8FC0: A9 03 D0 05 8A D0 08 A9
$8FC8: 04 8D 65 91 4C 30 91 A2
$8FD0: 0E 8E 66 91 20 D1 90 20
$8FD8: AA 90 AD 67 91 D0 03 4C
$8FE0: 81 8F 20 E6 90 A9 00 8D
$8FE8: 65 91 60 A2 00 20 05 90
$8FF0: A2 1D BD 6A 91 9D 88 91
$8FF8: CA 10 F7 20 B7 00 F0 EA
    
```

```

$9000: 20 BE DE A2 1E 86 F9 20
$9008: 7B DD 20 FD E5 85 EF A6
$9010: F9 A0 FF 20 8D 90 B0 06
$9018: D0 0A E0 01 D0 24 4C C9
$9020: DE 4C 99 E1 20 7D E0 90
$9028: F5 B0 07 20 8D 90 B0 4F
$9030: F0 10 09 80 9D 69 91 C0
$9038: 1D 90 F0 20 8D 90 B0 3F
$9040: D0 F9 84 47 86 46 A6 F9
$9048: F0 02 A2 0E 20 8E 90 C9
$9050: 44 D0 04 A9 03 D0 07 C9
$9058: 53 D0 C3 E8 A9 08 85 45
$9060: 20 8E 90 B0 B9 E9 2F F0
$9068: B8 C5 45 B0 B4 9D 4A 91
$9070: 20 8E 90 B0 04 F0 CF 90
$9078: A5 A4 47 F0 0F A6 46 A9
$9080: A0 C0 1E B0 07 9D 69 91
$9088: E8 C8 D0 F5 60 E8 C8 C4
$9090: EF B0 05 B1 5E C9 2C 18
$9098: 60 A0 06 A5 FA 91 ED C8
$90A0: A5 FB 91 ED A2 08 A9 03
$90A8: D0 19 A6 FE A4 FA D0 05
$90B0: 8A D0 01 60 CA 88 98 A0
$90B8: 06 91 ED C8 8A 91 ED A2
$90C0: 16 A9 04 48 C8 A5 73 91
$90C8: ED C8 A5 74 91 ED 68 D0
$90D0: 19 A0 02 A9 00 91 ED C8
$90D8: C0 06 D0 F9 A9 0A D0 0A
$90E0: A9 01 A0 04 D0 06 A2 16
$90E8: A9 02 A0 0C 48 BD 49 91
$90F0: 91 ED E8 C8 C0 12 D0 F5
$90F8: A0 00 68 91 ED C8 A9 02
$9100: 91 ED AE 69 91 20 D6 03
$9108: A0 0A B1 ED C9 04 F0 1A
$9110: C9 08 F0 18 C9 09 D0 0A
$9118: 20 E6 90 8D 65 91 A9 09
$9120: D0 0A C9 0A D0 02 A9 04
$9128: AA 60 A9 07 BA E8 E8 9A
$9130: AA 24 D8 10 03 4C E9 52
$9138: 20 FB DA A9 A3 20 5C DB
$9140: 8A 09 B0 20 5C DB 4C 2A
$9148: D4 00 A2 0E 20 6A 91 C9
$9150: 44 A6 91 00 92 00 94 00
$9158: 53 D0 C3 88 91 08 85 D3
$9160: 91 00 93 00 95 00 00 00
(ab hier Datenpuffer)
    
```

## DB-MEISTER

### Adreß- und Schemabriefprogramm

*Der DB-Meister ist ein in Assembler geschriebenes, ungewöhnlich schnelles, unkompliziertes und zugleich „narrensicheres“ Adreß-, Datei- und Schemabriefprogramm.*

Technische Daten

- Recordlänge bis zu 230 Zeichen
- 560 bis 1000 Records pro Datendiskette
- Maximal 25 Felder pro Record
- Suche nach 3 Indexfeldern
- Ausdruck der Dateien als Etiketten, Listen und Schemabriefe (mit Felder- und Tastatureinschieben an beliebigen Stellen des Formbriefes)
- normal kopierbare Programmdiskette, unterteilt in Hauptprogramme und diverse Hilfsprogramme
- einsatzfähig auf Apple IIe, IIc oder II Plus mit 2 Drives (1 Drive ebenfalls möglich)

**Gesamtpreis 290,- (2 Disketten + gedrucktes Manual)**

**U. Stiehl**

c/o Dr. A. Hüthig Verlag  
Postfach 10 28 69 · 6900 Heidelberg

## Tabelle der Fehlermeldungen und Hinweise

### Hinweise

- 1- Quelldiskette einlegen:  
& : fortfahren  
&. : Kopie abbrechen
- 2- Zieldiskette einlegen:  
& : fortfahren  
&. : Kopie abbrechen
- 3- Datei gesperrt:  
& : Datei überschreiben  
&NAME: mit neuem Namen weiter  
&. : Kopie abbrechen
- 4- Datei existiert schon:  
& : Datei überschreiben  
&NAME: mit neuem Namen weiter  
&. : Kopie abbrechen

### Fehlermeldungen

- 5- Falscher Aufruf:  
Versuch eines Neustarts bei laufender Kopie  
(Kopie beenden oder mit "&." abbrechen)
- 6- Datei existiert nicht:  
Versuch, nicht existierende Datei zu kopieren  
(mit neuem Namen nochmal versuchen)
- 7- Diskette schreibgeschützt:  
(Schreibschutzmarke entfernen und Kopie neu starten)
- 8- I/O-Fehler:  
Laufwerk nicht geschlossen oder Diskette schadhaf  
(andere Diskette einlegen und Kopie neu starten)
- 9- Diskette voll:  
Datei paßt nicht mehr auf Zieldiskette  
(unvollständige Datei löschen)

## STARTUP

```
101 REM * STARTUP *
110 HOME : PRINT "Ramdisk geladen ..."
120 PRINT CHR$(4)"BLOAD RAMDISKLC"
130 POKE 24902,96: POKE 25084,96: CALL 24895
140 PRINT "... und initialisiert"
150 PRINT CHR$(4)"BRUN AS.FILER"
160 HIMEM: PEEK (109) + PEEK (110) * 256 + 7 * 30: REM
210 Bytes für Strings
170 ONERR GOTO 300
180 READ AS: PRINT "Datei "AS;
190 & AS, "S3": PRINT " übertragen"
200 GOTO 180
299 REM Fehlerbehandlung
300 IF PEEK (222) = 42 THEN PRINT : PRINT "Übertragung
beendet": END
310 IF PEEK (222) = 6 THEN PRINT " existiert nicht": GOTO
180
320 IF PEEK (222) = 8 THEN PRINT CHR$(7): PRINT
"Diskettenfehler": END
330 IF PEEK (222) = 9 THEN PRINT CHR$(7): PRINT "RAM-Disk
voll": PRINT CHR$(4)"DELETE"AS",S3": END
340 POKE 763,166: POKE 764,222: POKE 765,76: POKE 766,25:
POKE 767,212: POKE 117, PEEK (218): POKE 118, PEEK
(219): CALL 763
400 DATA STARTUP, RAMDISKLC, AS.FILER, ENDUP
```

## ENDUP

```
1001 REM * ENDUP *
1010 PRINT CHR$(4)"BRUN AS.FILER"
1020 PRINT CHR$(12): HOME
1030 INPUT "Quellaufwerk (S,D): ";SLOT,DRIVE
1040 IF SLOT < 0 OR SLOT > 6 THEN 1050
1050 IF DRIVE < 0 OR DRIVE > 2 THEN 1030
1060 INPUT "Ziellaufwerk (S,D): ";ZS,ZD
1070 IF ZS < 0 OR ZS > 6 THEN 1060
1080 IF ZD < 0 OR ZD > 2 THEN 1060
1090 PRINT : PRINT "Zieldiskette einlegen..."
1100 PRINT TAB(22)"...Taste drücken": GET AS: PRINT
1110 HIMEM: PEEK (109) + PEEK(110) * 256 + 2000: REM 2000
Bytes für Strings
1120 ONERR GOTO 1300
1200 GOSUB 2000: IF ERR = 1 THEN 1900
1210 IF ERR THEN 1700
1220 PRINT : PRINT "Datei: ";NS;
1230 & NS + ",S" + STR$(SLOT) + ",D" + STR$(DRIVE),",S" +
STR$(ZS) + ",D" + STR$(ZD)
1240 GOTO 1200
1300 ON PEEK (222) GOTO 1400,1400,1500,1500,1900,1900,1600,
1700,1800
1400 & : GOTO 1200: REM kein Diskettenwechsel
1500 NS = NS + ".COPY": REM keine Datei überschreiben
1510 PRINT ".COPY"; & NS: GOTO 1200
1600 PRINT : PRINT CHR$(7)"Zieldiskette schreibgeschützt":
END
1700 PRINT : PRINT CHR$(7)"I/O-Fehler": END
1800 PRINT : PRINT CHR$(7)"Zieldiskette voll": END
1900 END
1997 REM Nächsten Catalog-Eintrag holen
2000 IF TRK = 17 THEN 2100
2010 TRK = 17: REM Catalog-Spur
2020 SEC = 15: REM 1. Catalog-Sektor
2030 IND = 11 - 35
2100 IND = IND + 35: IF IND = 256 THEN SEC = SEC - 1: IND =
11
2110 IF SEC = 0 THEN 2300
2120 IF IND = 11 THEN GOSUB 3000: IF ERR THEN 2400
2130 IF PEEK (512 + IND) = 0 THEN 2300
2140 IF PEEK (512 + IND) = 255 THEN 2100
2150 NS = ""
2160 FOR I = 544 + IND TO 516 + IND STEP - 1
2170 IF PEEK (I) < > 160 THEN 2200
2180 NEXT
2200 FOR J = 515 + IND TO I
2210 NS = NS + CHR$( PEEK (J) - 128): NEXT
2220 RETURN
2300 ERR = 1: RETURN : REM Catalog-Ende
2400 ERR = 2: RETURN : REM I/O-Fehler
2997 REM RWTS-Aufruf
3000 IOB = PEEK ( PEEK (996) + PEEK (997) * 256) * 256 +
PEEK ( PEEK (999) + PEEK (1000) * 256)
3010 RWTS = 768
3020 POKE 768,32: POKE 769,227: POKE 770,3: POKE 771,76:
POKE 772,217: POKE 773,3
3030 POKE IOB + 15, PEEK (IOB + 1): REM letzter Slot
3040 POKE IOB + 16, PEEK (IOB + 2): REM letzter Drive
3050 POKE IOB + 1,SLOT * 16: REM neuer Slot
3060 POKE IOB + 2,DRIVE: REM neuer Drive
3070 POKE IOB + 3,0: REM Volume-Nummer
3080 POKE IOB + 4,TRK: REM Spurnummer
3090 POKE IOB + 5,SEC: REM Sektornummer
3100 POKE IOB + 8,0: REM Low-Byte, Eingabepuffer
3110 POKE IOB + 9,2: REM High-Byte, Eingabepuffer
3120 POKE IOB + 12,1: REM Lesen
3130 CALL RWTS:ERR = PEEK (IOB + 13): RETURN
```

## Computer-unterstütztes Lernen

Maschinenschreiben, Basic, Wortschatztrainer, Computer-Simulator, Schnelllesen, Rechtschreibtrainer, usw.

- Programme für Apple IIe/IIc/teilw. +.
- Demo-Diskette mit 8 Programmen und 7 Denkspielen DM 10,-
- Gesamtkatalog mit über 200 Titeln kostenlos.

INTUS-Lernprogramme, Demo-Disketten und den Gesamtkatalog erhalten Sie bei den meisten Apple-Fachhändlern, in Computer-Fachabteilungen und bei Pandasoft.



INTUS SOFTWARE

Kaiserstraße 21, 7890 Waldshut, Telefon 077 51/79 20



## AS.FILER

```

1
2
3
4
5
6
7      ORG $8EA4
8
9  * Neu speichern mit: BSAVE AS.FILER, A$8EA4, L$02C1
10 * Nach BRUN: HIMEM maximal auf 36312 ($8DD8) setzen
11
12 PAGES    EQU 1           ;minimale Seitenzahl des Puffers
13 ACC      EQU $45        ;Akkumulator hierher retten
14 XREG     EQU $46        ;X-Register hierher retten
15 YREG     EQU $47        ;Y-Register hierher retten
16 INDEX    EQU $5E        ;Stringdeskriptor (ohne Länge)
17 MEMSIZE  EQU $73        ;<-> HIMEM
18 CHRGET   EQU $B1        ;Zeichen aus Quelltext holen
19 CHRROT   EQU $B7        ;letztes Zeichen nochmal
20 ERRFLG   EQU $D8        ;"ONERR GOT0"-Flag
21 IOB      EQU $EB        ;Adresse des IOB
22 FMPL     EQU $ED        ;Adresse der FMPL
23 LEN      EQU $EF        ;Länge bei Namensbestimmung
24 SDINDEX  EQU $F9        ;Index auf Slot oder Drive
25 BUPRNGE  EQU $FA        ;Umfang des Puffers
26
27 FM       EQU $3D6        ;Sprung zum File-Manager
28 GETFMPL  EQU $3DC        ;Adresse der FMPL bestimmen
29 GETIOB   EQU $3E3        ;Adresse des IOB bestimmen
30 AMPER    EQU $3F5        ;&-Vektor
31
32 ILLQERR  EQU $E199       ;"ILLEGAL QUANTITY"
33 MEMERR   EQU $D410       ;"OUT OF MEMORY"
34 ERRENT   EQU $D42A       ;Fehlerausgabe (Einsprung)
35 CRDO     EQU $DAPB       ;CR ausgeben
36 OUTDO    EQU $DB5C       ;Zeichen ausgeben
37 CHKCOM   EQU $DEBE       ;prüfen ob Komma folgt
38 SYNERR   EQU $DEC9       ;"SYNTAX"
39 FRMEVL   EQU $DD7B       ;Ausdruck auswerten
40 ISLETC   EQU $E07D       ;prüfen, ob Buchstabe
41 FRESTR   EQU $E5FD       ;Stringdeskriptor freigeben
42 HANDLERR EQU $F2E9       ;Fehlerbehandlung
43
44 *----- Initialisierung -----*
45
46 INIT     LDA #<COPY      ;Adresse
47          LDY #>COPY      ; der COPY-Routine in
48          STA AMPER+1     ; &-Vektor eintragen
49          STY AMPER+2
50          JSR GETFMPL     ;Adresse der FMPL
51          STY FMPL        ; holen
52          STA FMPL+1
53          JSR GETIOB     ;Adresse des IOB
54          STY IOB         ; holen
55          STA IOB+1
56          LDY #0          ;Status für
57          STY STATUS     ; Neustart
58          INY             ;Index auf Slot
59          LDA (IOB),Y     ;zuletzt
60          LSR             ; benutzt
61          LSR             ; Slot
62          LSR             ; durch 16
63          LSR
64          STA PRIMSLT    ; als Ersatzwert
65          STA SECSLT     ; eintragen
66          INY
67          LDA (IOB),Y     ;letzten Drive als
68          STA PRIMDRVE   ; Ersatzwert
69          STA SECDRVE    ; eintragen
70          RTS
71
72 *----- Kopieren -----*
73
74 COPY     CMP #','        ;Komma
75          BNE NOCOM       ; bricht Kopie ab
76          JSR CHRGET      ;Textzeiger erhöhen
77          JMP ENDCOPY     ; und Kopie beenden
78          NOCOM          LDA STATUS ;Status = 0,
79          BEQ NEWCOPY     ; d.h. neue Kopie
80          CPX #1          ;Quelldiskette eingelegt?
81          BNE NOTRD
82          JSR CHKEOS      ;Endmarker muß folgen
83          JMP RDBUF1
84          NOTRD          CPX #2 ;Zieldiskette eingelegt?
85          BNE NOTWRT

```

```

86          JSR CHKEOS      ;Endmarker muß folgen
87          JMP OPEND1
88          NOTWRT        CPX #3 ;Datei gesperrt?
89          BNE NOTLCKD   ; nein, dann weiter
90          JSR CHRROT    ;Endmarker?
91          BNE NEWNME    ; ja, dann überschreiben
92          LDX #14       ;zweiter Puffer
93          LDA #58       ;"UNLOCK"
94          JSR CALLFM    ;aufschließen
95          JMP DELETE    ; und löschen
96          NOTLCKD      JSR CHRROT ;Endmarker? (Status 4)
97          BNE NEWNME    ; nein, dann neuer Name
98          DELETE       LDX #14 ;zweiter Puffer
99          LDA #5        ;"DELETE"
100         JSR CALLFM    ;existierende Datei löschen
101         JMP OPEND2     ; und nochmal versuchen
102         NEWNME        JSR GETDST ;neue Zieldatei
103         JSR CHKEOS    ;Endmarker muß folgen
104         JMP OPEND2     ; nochmal versuchen
105
106         CHKEOS        JSR CHRROT ;Endmarker?
107         BNE ILLCALL   ; nein, dann Fehler
108         RTS
109         ILLCALL       LDA #5 ;"ILLEGAL CALL"
110         JMP EXIT0     ; Fehlermeldung
111
112 * neue Kopie beginnen
113
114         NEWCOPY       LDA #0
115         STA LSTPASS    ;noch nicht letzter Durchgang
116         STA OPNPLG    ;Zieldatei noch geschlossen
117         SEC
118         LDA #<COPY     ;Pufferumfang
119         SBC MEMSIZE    ; berechnen
120         STA BUPRNGE
121         LDA #>COPY
122         SBC MEMSIZE+1
123         STA BUPRNGE+1
124         BCC MEMERR1    ;negativ, dann Fehler
125         SBC #PAGES     ;Mindest-Seitenzahl?
126         BCS NEWCOPY1   ; ja, dann weiter
127         MEMERR1       JMP MEMERR ;"OUT OF MEMORY"
128
129         NEWCOPY1      JSR GETNME ;Quell- und Zielname einlesen
130         LDA PRIMSLT    ;Quell- und Ziellaufwerk
131         CMP SECSLT     ; gleich?
132         BNE SETFLAG
133         LDA PRIMDRVE   ; (C = 1)
134         SBC SECDRVE
135         SETFLAG       STA SAMESD ;= 0, d.h. gleiches Laufwerk
136
137 * Quelldatei eröffnen
138
139         LDX #1         ;Neuzuweisung
140         STX ALLOCATE   ; verhindern
141         DEX             ;erster Puffer
142         JSR OPEN       ;Datei eröffnen
143         BEQ FLEFND     ;kein Fehler, dann weiter
144         JMP EXIT       ;"FILE NOT FOUND"
145         FLEFND        LDY #57 ;Index auf Dateityp
146         LDA (FMPL),Y   ;Dateityp
147         AND #%0111111 ; (Lockbit löschen)
148         STA FILETYP    ; eintragen
149
150 * Quelldatei positionieren
151
152         LDX #0         ;erster Puffer
153         JSR POSITION    ;auf Null positionieren
154         BEQ RDBUF1    ;weiter mit einlesen
155
156 * Quelldatei schubweise lesen
157
158         RDBUF         LDA SAMESD ;ein Laufwerk?
159         BNE RDBUF1    ; nein, dann weiter
160         LDA #1        ;"INSERT SOURCE DISK"
161         BNE RTNOPN
162
163         RDBUF1        JSR READ
164         BEQ OPEND      ;kein "END OF DATA"
165         STA LSTPASS    ;ansonsten letzten Schub
166         CLC            ; markieren
167         LDY #6         ;Index auf Pufferlänge
168         LDA BUPRNGE    ;Differenz zwischen
169         SBC (FMPL),Y   ; vorgegebener Pufferlänge
170         STA BUPRNGE    ; und dem Restpuffer
171         INY            ; gibt die Länge des letzten
172         LDA BUPRNGE+1 ; Durchgangs an
173         SBC (FMPL),Y

```

```

174      STA  BUFRNGE+1
175
176      * Zieldatei eröffnen
177
178  OPEND  LDA  SAMESD      ;zwei Laufwerke?
179          BNE  OPEND1    ; ja, dann weiter
180          LDA  #2        ; "INSERT DESTINATION DISK"
181          BNE  RTNOPN
182
183  OPEND1 LDA  OPNFLG     ;Zieldatei schon eröffnet?
184          BNE  WRTBUF    ; ja, dann weiter
185  OPEND2 LDX  #0        ;Neuzuweisung
186          STX  ALLOCATE  ; möglich
187          LDX  #14       ;zweiter Puffer
188          JSR  OPEN
189          LDY  #57        ;Index auf Dateityp
190          LDA  (FMPL),Y  ;Datei gesperrt?
191          BPL  OPEND3    ;nein, dann weiter
192          LDA  #3        ; "FILE LOCKED"
193          BNE  RTNOPN    ;unbedingt
194  OPEND3 TXA            ;Fehernummer < 0.
195          BNE  POSD      ; d.h. Datei existierte nicht
196          LDA  #4        ; "FILE ALLREADY EXISTS"
197  RTNOPN STA  STATUS     ;Meldung an Benutzer
198          JMP  EXIT
199
200      * Zieldatei positionieren
201
202  POSD   LDX  #14       ;zweiten Puffer
203          STX  OPNFLG   ; als eröffnet markieren
204          JSR  POSITION  ;auf Null positionieren
205
206      * Zieldatei schreiben
207
208  WRTBUF JSR  WRITE     ;Puffer schreiben
209          LDA  LSTPASS   ;letzter Durchgang?
210          BNE  CLOSED   ; ja, dann fertig
211          JMP  RDBUF    ; ansonsten nächster Schub
212
213      * Zieldatei schließen, fertig
214
215  CLOSED JSR  CLOSE
216  ENDCOPY LDA #0
217          STA  STATUS   ;Neustart möglich
218  DONE   RTS
219
220      *----- allgemeine Unterprogramme -----*
221
222      * Dateiname(n) auswerten
223
224  GETNME LDX  #0        ;ersten Namen überprüfen
225          JSR  GETNME1
226          LDX  #29
227  COPYNME LDA PRIMNME,X ;ersten Namen
228          STA  SECNME,X ; in zweiten Puffer
229          DEX          ; eintragen
230          BPL  COPYNME
231          JSR  CHRGTOT  ;zweiter Name?
232          BEQ  DONE     ; nein, dann fertig
233          JSR  CHKCOM   ;Komma muß folgen
234  GETDST LDX  #30      ;zweiten Namen überprüfen
235
236  GETNME1 STX  SDINDX   ;retten
237          JSR  FRMEVL   ;Ausdruck auswerten
238          JSR  FRESTR   ;String wieder freigeben
239          STA  LEN      ;Stringlänge eintragen
240          LDX  SDINDX   ;Index auf 1. oder 2. Puffer
241          LDY  #-1      ;Index auf String
242          JSR  GETCHR   ;erstes Zeichen holen
243          BCS  SYNERR1  ;leerer String verboten
244          BNE  CHKLET   ;kein Komma, dann weiter
245          CPX  #1       ;X = 1, d.h. erster Name
246          BNE  COMMA   ;Komma bei Zielname erlaubt
247          SYNERR1 JMP  SYNERR ; "SYNTAX"
248  ILLQERR1 JMP  ILLQERR ; "ILLEGAL QUANTITY"
249  CHKLET  JSR  ISLETC  ;prüfen ob Buchstabe
250          BCC  SYNERR1  ;kein Buchstabe -> Fehler
251          BCS  CHRL00P1 ; ansonsten eintragen
252
253  CHRLOOP JSR  GETCHR   ;nächstes Zeichen
254          BCS  EOSTR     ; (Ende erreicht)
255          BEQ  COMMA    ; (Komma gefunden)
256  CHRLOOP1 ORA  #10000000 ;Bit 7 = 1
257          STA  PRIMNME-1,X ; (-1 wegen INX bei GETCHR)
258          CPY  #29      ;maximale Länge erreicht?
259          BCC  CHRLOOP  ; nein, dann weiter
260  COMLOOP JSR  GETCHR   ;weiterrufen bis Komma
261          BCS  EOSTR     ; oder Stringende

```

```

262      BNE  COMLOOP
263
264  COMMA  STY  YREG      ;Indexe
265          STX  XREG     ; retten
266  COMMA1 LDX  SDINDX   ;Index auf Slot oder Drive
267          BEQ  COMMA2   ; 0, dann weiter
268          LDX  #14      ; ansonsten auf Zielparameter
269  COMMA2 JSR  GETCHR1  ; "D" oder "S" muß folgen
270          CMP  #'D'
271          BNE  NOTDRVE
272          LDA  #3        ;Maximalwert + 1
273          BNE  GETPAR
274  NOTDRVE CMP  #'S'
275          BNE  SYNERR1  ;weder "S" noch "D"
276          INX          ;nach Drive kommt Slot
277          LDA  #8        ;Maximalwert + 1
278  GETPAR STA  ACC
279          JSR  GETCHR1  ;Parameter
280          BCS  SYNERR1  ; muß folgen
281          SEC  #0'-1    ;aus ASCII wird Hex
282          BEQ  ILLQERR1 ; 0 ist verboten
283          CMP  ACC      ;mit vorgegebenem Wert
284          BCS  ILLQERR1 ; vergleichen
285          STA  PRIMDRVE,X ;Parameter eintragen
286          JSR  GETCHR1
287          BCS  EOSTR0   ;C-Flag geht vor Z-Flag
288          BEQ  COMMA1  ;noch ein Komma -> weiter
289          BCC  SYNERR1 ;kein Ende -> Fehler
290
291  EOSTR0 LDY  YREG      ;wenn 0, wurde kein 2. Namen
292          BEQ  RTNGN    ; angegeben -> belassen
293          LDX  XREG     ;altes Register (vor COMMA)
294  EOSTR  LDA  #" "      ;Rest mit Blanks
295  EOSTR1 CPY  #30      ; auffüllen
296          BCS  RTNGN
297          STA  PRIMNME-1,X
298          INX
299          INY
300          BNE  EOSTR1
301  RTNGN  RTS
302
303  GETCHR INX          ;Index
304  GETCHR1 INY         ; erhöhen
305          CPY  LEN      ;Ende erreicht?
306          BCS  RTNGC   ; ja, dann zurück
307          LDA  (INDEX),Y ;Zeichen holen
308          CMP  #','     ;Z = 1 bei Komma
309          CLC
310  RTNGC  RTS          ;C = 1 bei Stringende
311
312      *- File-Manager-Aufrufe -*
313
314  READ   LDY  #56      ;Index auf Pufferlänge
315          LDA  BUFRNGE  ;Pufferlänge
316          STA  (FMPL),Y ; eintragen
317          INY
318          LDA  BUFRNGE+1
319          STA  (FMPL),Y
320          LDX  #58      ;erster Puffer
321          LDA  #53      ; "READ"
322          BNE  RDWRT
323
324  WRITE  LDX  BUFRNGE+1 ;beim Schreiben muß
325          LDY  BUFRNGE  ; die Länge des Puffers
326          BNE  WRITE2   ; um 1 erniedrigt werden
327          TXA          ;Länge = 0,
328          BNE  WRITE1   ; d.h.
329          RTS          ; fertig
330  WRITE1 DEX
331  WRITE2 DEY
332          TYA          ;eintragen
333          LDY  #6       ;Index auf Pufferlänge
334          STA  (FMPL),Y
335          INY
336          TXA
337          STA  (FMPL),Y
338          LDX  #14+8    ;zweiter Puffer
339          LDA  #54      ; "WRITE"
340  RDWRT  PHA
341          INY          ;Pufferadresse
342          LDA  MEMSIZE  ; eintragen
343          STA  (FMPL),Y
344          INY
345          LDA  MEMSIZE+1
346          STA  (FMPL),Y
347          PLA          ; Befehlsnummer
348          BNE  CALLFMI ;unbedingt

```

```

349
350 POSITION LDY #2 ;Index auf Record-Werte
351 LDA #0 ;Länge und Offset
352 POSN1 STA (FMPL),Y ; zu Null
353 INY
354 CPY #6 ;insgesamt 4 Bytes
355 BNE POSN1
356 LDA #A ;"POSITION"
357 BNE CALLFM1
358
359 OPEN LDA #1 ;"OPEN"
360
361 * eigentlicher Aufruf des FM
362
363 CALLFM LDY #4 ;ab Volume eintragen
364 BNE CALLFM2 ;unbedingt
365
366 CLOSE LDX #14+8 ;zweiter Puffer
367 LDA #2 ;"CLOSE"
368 CALLFM1 LDY #C ;ab Pufferadressen eintragen
369 CALLFM2 PHA ;Befehlsnummer merken
370 SETFMPL LDA PLTABLE,X ;Tabelle
371 STA (FMPL),Y ; übertragen
372 INX
373 INY
374 CPY #12 ;Ende erreicht?
375 BNE SETFMPL ; nein, dann weiter
376 LDY #0 ;Index auf Befehlsnummer
377 PLA
378 STA (FMPL),Y
379 INY ;Index auf Subcode
380 LDA #2 ;immer (Bereich ohne POS.)
381 STA (FMPL),Y
382 LDX ALLOCATE ;Neuzuweisung?
383 JSR FM ;File-Manager via Page 3
384 LDY #A ;Fehlercode
385 LDA (FMPL),Y ; holen
386 CMP #4 ;schreibgeschützt?
387 BEQ WPERR
388 CMP #8 ;I/O-Fehler?
389 BEQ EXIT0
390 CMP #9 ;Diskette voll?
391 BNE NOTDF ;nein, dann weiter
392 JSR CLOSE ;ansonsten Datei schließen,
393 STA STATUS ; Neustart einstellen
394 LDA #9 ; und Fehlermeldung
395 BNE EXIT0 ; an Benutzer
396 NOTDF CMP #10 ;Datei gesperrt?
397 BNE RTNCFM
398 LDA #4 ;"FILE LOCKED"
399 RTNCFM TAX ;Zero-Flag aktualisieren
400 RTS
401 WPERR LDA #7 ;"WRITE PROTECTED"
402
403 *----- Fehlerbehandlung -----*
404
405 EXIT0 TSX ;Return-Adresse
406 INX ; vom Stack
407 INX
408 TXS
409 EXIT TAX
410 BIT ERRFLG ;ERRorFLaG gesetzt?
411 BPL COPYERR ; nein, dann weiter
412 JMP HANDLERR ;"ONERR GOTO" behandeln
413 COPYERR JSR CRDO
414 LDA #"" ;Fehlernummer
415 JSR OUTDO
416 TXA ; ausgeben
417 ORA #""
418 JSR OUTDO
419 JMP ERRENT ;weiter mit Fehlerbehandlung
420
421 *----- Tabellen, Variablen, Puffer -----*
422
423 PLTABLE ;FMPL-Tabellen
424 DFB $0 ;Pos. $4 Volume-Nummer
425 PRIMDRVE DS 1 ;Pos. $5 Drive-Nummer
426 PRIMSLT DS 2 ;Pos. $6 Slot-Nummer
427 DA PRIMNME ;Pos. $8 1. Namensadresse
428 DS 2 ;Pos. $A Fehlercode
429 DA PRIMFMWA ;Pos. $C Arbeitsbereich
430 DA PRIMTSL ;Pos. $E TSL-Puffer
431 DA PRIMDAT ;Pos. $10 Puffer-Adresse
432
433 DFB $0 ;Pos. $4 Volume-Nummer
434 SECDRVE DS 1 ;Pos. $5 Drive-Nummer
435 SECSLT DS 1 ;Pos. $6 Slot-Nummer

```

```

436 FILETYP DS 1 ;Pos. $7 Datei-Typ
437 DA SECNME ;Pos. $8 2. Namensadresse
438 DS 2 ;Pos. $A Fehlercode
439 DA SECFMWA ;Pos. $C Arbeitsbereich
440 DA SECTSL ;Pos. $E TSL-Puffer
441 DA SECDAT ;Pos. $10 Puffer-Adresse
442
443 STATUS DS 1 ;Aufrufstatus
444 OPNFLG DS 1 ;<> 0, wenn Zieldatei eröffnet
445 LSTPASS DS 1 ;<> 0 bei letztem Schub
446 SAMESD DS 1 ;0, wenn gleiche Laufwerke
447 ALLOCATE DS 1 ;Flag für Dateineuzuweisung
448
449 PRIMNME DS 30 ;Puffer für ersten Namen
450 SECNME DS 30 ;Puffer für zweiten Namen
451 PRIMFMWA DS 45 ;File-Manager-Arbeitsbereich
452 SECFMWA DS 45 ; umfaßt 45 Bytes
453 PRIMTSL DS $100 ;Track/Sector-List-Puffer
454 SECTSL DS $100 ; umfaßt einen Sektor
455 PRIMDAT DS $100 ;Quell-Datenpuffer
456 SECDAT DS $100 ;Ziel-Datenpuffer

```

1884 Bytes



## ProDOS-Begleitdiskette

zu „ProDOS für Aufsteiger“, Band 2  
Soeben erschienen, DM 28,-

Diese Begleitdiskette ist mehr als nur eine Diskette zum Buch, denn sie enthält neben etwa 20 Demoprogrammen eine ganze Reihe selbständiger und sofort einsatzfähiger Programme, die eine wichtige Ergänzung zum Betriebssystem darstellen, z. B.:

- Dateileseprogramme zum Dump von Textfiles und anderen Dateien in ASCII- und Hex-Darstellung
- Dateikopierprogramm zum Kopieren beliebig großer Dateien (PROFID)
- Diskettenkopierprogramme für 1- und 2-Drive-Besitzer
- Formatierungsprogramm
- Diskettenvergleichsprogramm
- Bad-Block-Programm
- DOS-3.3-Konvertierungsprogramm
- Blockeditor-Utility
- Directory-Utilities
- Zahlenarray-BSAVE/BLOAD-Utility

Alle Programme laufen unter allen ProDOS-Versionen und können auch von Nicht-Programmierern problemlos eingesetzt werden.

**Hüthig Software Service**  
Postfach 102869 · 6900 Heidelberg 1

# ProDOS-Fastboot

oder wie man ProDOS in 6 Sekunden bootet

von Arne Schäpers

nach einer Idee von Ulrich Stiehl

Das Booten unter ProDOS dauert etwa 10 Sekunden. Das nachfolgende Programm vereinigt die zwei Systemdateien PRODOS und BASIC.SYSTEM zu einer einzigen bootfähigen Gesamtdatei namens PBASIC, die nach PR#6 in weniger als 6 Sekunden eingeladen wird. Wer unter ProDOS häufig Programme durch Neubooten wechseln muß, wird deshalb diese Fastboot-Routine, die bei allen bisher bekannten PRODOS-(1.0.1, 1.0.2, 1.1.1) und BASIC-SYSTEM-Versionen (1.0, 1.1) funktioniert, zu schätzen wissen.



## 1. Urlader

Über den Boot-Prozeß von ProDOS ist bereits detailliert geschrieben worden. Wer es ganz genau wissen möchte, kann darüber in der „ProDOS-Analyse“ nachlesen. Hier folgen deshalb nur die logischen Stufen des Bootvorgangs bis zum Start des Hello-Programms „STARTUP“:

1. Das Programm auf der Controller-Karte liest den Sektor 0 auf Spur 0 nach \$0800 in den Speicher und springt nach \$0801. Das soeben geladene Programm („Urlader“) benutzt das Programm auf der Controller-Karte, um die zweite Hälfte des Blocks 0 (Spur 0, Sektor 2) zu lesen. Damit ist der Urlader vollständig im Speicher.
2. Der Urlader lädt das Volume-Directory (Blocks 2-5) und sucht es nach dem Eintrag „PRODOS“ ab.
3. Dann lädt er die Datei PRODOS ab \$2000 in den Speicher und springt nach \$2000. Das Programm PRODOS reloziert sich selbst in die zwei Speicherbanks der Language-Card.
4. Das Programm PRODOS liest nun erneut das Volume-Directory ein und sucht es nach der ersten Datei mit der Namensendung „.SYSTEM“ ab.
5. Das „.SYSTEM“-Programm wird ab \$2000 in den Speicher geladen und von dort gestartet.
6. Wenn es sich um das BASIC.SYSTEM handelt, so passiert danach folgendes:  
Das Programm kopiert (*nicht* reloziert) sich selbst in den Speicherbereich \$9A00-\$B9FF und sucht nach einem CLEAR PREFIX und ONLINE die Datei „STARTUP“ über GET FILE INFO, d.h. das Volume-Directory wird abgesucht, nunmehr aber über MLI-Befehle und nicht durch Laden der Volume-Directory-Blocks. Wird STARTUP gefunden, so kopiert das Programm die Befehlszeile „-STARTUP“ in den Input-Puffer. Es folgt ein Kaltstart des BASIC.SYSTEMs über \$BE00.  
Dieser Boot-Prozeß ist so komplex, daß jeder graue Haare bekommt, wenn er ihn modifizieren will.

## 2. Die Idee

Wer häufig Programme wechselt, muß häufig booten. Dies dauert jedoch unter ProDOS (ca. 9-10s) länger als unter DOS 3.3 (ca. 4-5s), weil die ProDOS-Systemdateien größer sind. Deshalb ist es wünschenswert, diesen Prozeß zeitlich abzukürzen.

Zuerst soll geschildert werden, was man aus dem Boot-Prozeß eliminieren kann:

1. Die Gerätekonfiguration, d.h. Speichergröße, Zusatzkarten usw. sind immer gleich. Ein Test ist daher nach einer einmaligen Installation nicht mehr nötig. Damit entfällt auch der ohnehin größtenteils nutzlose Relokator von PRODOS.
2. Das insgesamt dreimalige Lesen des Volume-Directory ist auf den Befehl „-PRODOS“ bzw. „-BASIC.SYSTEM“ zurückzuführen. Da über dieses Kommando die Programme direkt aufgerufen werden können, dürfen sie nicht auf die vom Urlader bereits gelesenen Directory-Blocks zurückgreifen. Bei einem Boot „in einem Rutsch“ muß das Directory nur einmal gelesen werden.
3. Wenn man jetzt noch den Urlader dahingehend modifiziert, zumindest einen Teil des Systems direkt an die endgültige Speicheradresse zu lesen, entfällt deren Verschieberoutine mit der Folge, daß bei einem Reboot ein wesentlich größerer Speicherbereich unversehrt bleibt.

Unter Berücksichtigung dieser Besonderheiten wurde ein Programm mit den folgenden Charakteristika geschrieben:

1. Es ist lauffähig unter allen bekannten Versionen (1.0.1, 1.0.2, 1.1.1).
2. Man hat 5 freie Blocks mehr auf der Diskette.
3. Die Ladezeit von PR#6 bis zur Meldung des BASIC.SYSTEMs wurde fast halbiert. Sie beträgt nun ca. 5.6s anstelle von ca. 10s.
4. Der bei einem Reboot unzerstörte Speicherbereich erstreckt sich von \$1400 bis \$61FF, also ein zusammenhängendes Stück von mehr als \$4800 Bytes. Dies dürfte für die meisten Programme während der Entwicklungsphase ausreichend sein.
5. Alle anderen Eigenschaften des Bootvorgangs sind unverändert, d.h. die Datei STARTUP ist nach wie vor fakultativ: Sie wird ausgeführt, wenn sie auf der Diskette gefunden wird, ansonsten erfolgt nur ein Kaltstart des BASIC.SYSTEMs. Das Präfix und die „zuletzt benutzte Unit“ (= Slot/Drive) in der System Global Page sind korrekt gesetzt.

## 3. Einschränkungen

Die neu angelegte Fastboot-Diskette sollte nur auf demjenigen Gerät eingesetzt werden, auf dem sie erzeugt wurde. Andernfalls können Probleme auftreten. Beispiele:

- Generierung auf einem System mit Uhrenkarte und Booten auf einem System ohne Uhrenkarte oder mit einer Uhrenkarte in einem anderen Slot.
- Generierung auf einem System mit RAM-Disk und Booten auf einem System mit 64K oder umgekehrt: Im ersten Fall führt das Ansprechen der vermeintlichen RAM-Disk zu einem unverhofften BRK, im zweiten Fall wird eine eventuell installierte RAM-Disk-Routine (auf der 64K-Karte) zerstört.

Ein weiterer Nachteil soll nicht verschwiegen werden: Ein „BRUN PBASIC“ oder „-PBASIC“ führt zu einem katastrophalen Absturz, weil das Programm an die falsche Startadresse geladen wird. Ein „BRUN PBASIC,\$6400“ hilft nicht weiter, denn hier bricht ProDOS mit einer Fehlermeldung ab, weil dabei beim Ladevorgang das BASIC.SYSTEM überschrieben werden würde. Man muß also die PBASIC-Diskette immer mit PR#6 oder allgemein PR#s starten. Die Fastboot-Routine ist damit nicht für Festplatten gedacht.

## 4. Bedienungsanleitung

Eine auf das reine Handling beschränkte Schritt-für-Schritt-Kurzanleitung finden Sie vor dem Listing des Programms MKBOOT.BAS. Der Ablauf der Erstellung von PBASIC ist folgender:

Nach Angabe der Nummer des Slots, von dem aus PBASIC in Zukunft gebootet werden soll, wird das gesamte System aus den verschiedenen Banks (LC Bank 1, Bank 2 und 64K-Karte) zu einem Block zusammenkopiert, der dann auf der Zieldiskette unter dem Namen PBASIC gespeichert wird.

Danach wird der Urlader auf dieser Diskette so modifiziert, daß nicht mehr ProDOS, sondern PBASIC im Volume-Directory gesucht wird. Außerdem lädt der Urlader nicht mehr ab \$2000, sondern ab \$6400 in den Speicher. Damit wird das gesamte BASIC.SYSTEM direkt in den späteren Bereich übertragen. Der File PBASIC enthält noch ein kurzes Start-Programm, mit dem das MLI, REBOOT und der RAM-Disk-Treiber in die entsprechenden Banks bzw. in die 64K-Karte AUX kopiert werden. Danach wird von PBASIC aus die Routine im Urlader, die das geladene Volume-Directory vorher nach „PBASIC“ abgesucht hat, so modifiziert, daß bei einem zweiten Durchlauf die Existenz der Datei „STARTUP“ überprüft wird. Wenn sie vorhanden ist, so wird der Befehl „-STARTUP“ simuliert. Ansonsten

erfolgt nur ein Kaltstart des BASIC.SYSTEMS über \$BE00.

## 5. Technische Einzelheiten

**MKBOOT.OBJ** stellt folgende Bereiche für PBASIC zusammen:

\$8F00-\$8FFF: System Global Page

\$8E00-\$8EFF: BASIC Global Page

\$6A00-\$8DFF: BASIC.SYSTEM. Die letzten \$0400 Bytes sind der Pufferbereich des BASIC.SYSTEMS und damit entbehrlich.

\$3A00-\$69FF: MLI, \$0600 Bytes sind Pufferbereiche, die sich aber je nach Version in den Adressen unterscheiden.

\$3700-\$39FF: REBOOT von \$D100-D3FF aus der Bank 2 der LC

\$3500-\$36FF: RAM-Disk-Driver aus \$0200-\$03FF aus der 64K-Karte, bei 64K-Maschinen aus den unteren 64K („Müll“)

\$34E0-\$34FF: Reset, BRK und NMI-Vektoren von \$03E0-\$03FF

**DOBOOT.OBJ** wird an den Anfang von PBASIC gesetzt, d.h. belegt den Bereich \$3400-\$34DF.

Der Urlader lädt PBASIC ab \$6400 in den Speicher. DOBOOT.OBJ ist deshalb nur ab dieser Adresse lauffähig!

Hier werden dann das MLI, die REBOOT-Routine und der RAM-Disk-Driver zurückkopiert. Der RAM-Disk-Driver muß dabei vor den Reset-Vektoren kopiert werden, weil er sonst bei 64K-Maschinen die Vektoren in den unteren 64K überschreiben würde.

Nachdem alle Verschiebevorgänge abgeschlossen sind, wird es kompliziert:

Als nächstes muß Applesoft kaltgestartet werden. Ein „JSR \$E000“ wäre zwecklos, weil der Stack innerhalb der Initialisierungsroutine neu gesetzt wird. Deshalb bleibt nur der Umweg über die COUT-Vektoren: Wenn Applesoft das Prompt Ü bzw. J (= \$DD) drucken will, wird es über GETBAS abgefangen, d.h. COUT springt zu GETBAS als vermeintlicher Ausgaberoutine, nachdem CSWL (= \$0036-\$0037) entsprechend gesetzt worden sind. (Interessanterweise druckt Applesoft das erste Return vor dem Prompt, bevor die Initialisierung vollständig beendet ist. Deshalb muß auf \$DD getestet werden, bevor es innerhalb von DOBOOT.OBJ weitergeht.)

In GETBAS werden jetzt für COUT die Vektoren des Monitors (\$FDF0) ersetzt, HIMEM und FRETOP (\$0070 und \$0074) auf \$9600 (dezimal 38400) eingestellt sowie das Trace-Flag \$00F2 für das BASIC.SYSTEM gesetzt.

Als nächstes wird die Suchroutine innerhalb des Urladers so umgebaut, daß nicht mehr PBASIC, sondern STARTUP gesucht wird. Der Pointer innerhalb der geladenen Directory-Blocks (\$4B) wird wieder auf den Directory-Anfang gesetzt, und der Vergleich darf nicht mehr Storage-Type (Speichertyp) und Namenslänge einschließen, weil STARTUP auch einen Storage-Type gleich 1 haben darf. Die Vergleichsschleife endet deshalb nicht mehr mit „BPL“, sondern mit „BNE“. Dabei muß aber geprüft werden, ob der gefundene File-Eintrag nicht gelöscht ist: Das erste Byte (Storage Type + Namenslänge) wird separat geladen und das nachfolgende „BIT \$D4“ wird zu „BEQ nicht gleich“. Der Exit aus der Suchroutine wird ebenfalls modifiziert: Bei „gefunden“ folgt RTS mit Carry clear, bei „nicht gefunden“ folgt RTS mit Carry set, also *nicht* „\*\*\* UNABLE TO LOAD PRODOS \*\*\*“.

Als letztes wird der Dateiname „STARTUP“ sowohl in die Suchroutine als auch nach \$0200 (Input-Puffer für BASIC) kopiert. Danach folgt der Aufruf der Suchroutine.

Wenn Ihnen das noch nicht kompliziert genug war, beschäftigen Sie sich mit dem nächsten Schritt:

\$0038-\$0039 sind mit SETIOV auf die Routine FAKECR gesetzt worden, d.h. wenn das BASIC.SYSTEM nach dem Prompt zu ersten Mal einen Input erwartet, wird über GETLN nicht GETKEY, sondern FAKECR als vermeintliche Input-Routine aufgerufen.

Innerhalb von FAKECR werden zuerst die Monitor-Vektoren in \$0038-\$0039 gesetzt, d.h. FAKECR hängt sich selbst beim ersten Aufruf ab.

Wir sind aber momentan noch innerhalb der Installationsroutine DOBOOT.OBJ: War die Suche nach STARTUP erfolgreich, wird ein Kaltstart des BASIC.SYSTEMS durchgeführt. Beim ersten Aufruf von GETKEY produziert FAKECR ein Return und kehrt, nachdem es sich selbst abgehängt hat, mit X=8 und A=\$8D=Return zurück. Das BASIC.SYSTEM „glaubt“ jetzt, daß hier jemand nach 8 Zeichen Eingabe auf die Return-Taste gedrückt hat. Im Input-Puffer steht noch „-STARTUP“ vor dem Return, und genau dieses Kommando führt das BASIC.SYSTEM dann auch aus.

Wenn STARTUP nicht gefunden wurde, wird FAKECR vor dem Kaltstart des BASIC.SYSTEMS *einmal* aufgerufen, hängt sich dabei selbst ab, und die Werte von A und X werden ignoriert, d.h. wenn das BASIC.SYSTEM danach GETLN aufruft,

wird bereits beim ersten Mal ein Zeichen von der Tastatur gelesen. Der Index (X-Register) ist 0, und die Zeichenfolge „-STARTUP“ im Input-Puffer wird mit den eingegebenen Zeichen überschrieben.



## MKBOOT.OBJ Hex-Dump

BSAVE MKBOOT.OBJ, A\$3000, L205

```
$3000: 4C 06 30 4C 81 30 20 56
$3008: 30 A9 BF 85 01 A9 8F 85
$3010: 03 A2 26 20 00 01 A9 FF
$3018: 85 01 A2 30 2C 8B C0 20
$3020: 00 01 A9 D3 85 01 A2 03
$3028: 2C 83 C0 20 00 01 2C 82
$3030: C0 A9 03 85 01 A2 02 EE
$3038: 01 01 20 00 01 CE 01 01
$3040: A9 03 85 01 A2 01 A0 E0
$3048: 20 00 01 A2 03 A9 00 9D
$3050: 38 8E CA 10 FA 60 A0 1B
$3058: B9 65 30 99 FF 00 88 D0
$3060: F7 84 00 84 02 60 8D 02
$3068: C0 8D 04 C0 B1 00 91 02
$3070: C8 D0 F9 C6 01 C6 03 CA
$3078: D0 F2 8D 02 C0 8D 04 C0
$3080: 60 AD FE 02 8D AA 30 20
$3088: 00 BF 80 A9 30 D0 16 A2
$3090: 00 BC AF 30 BD BE 30 99
$3098: BE 90 EB E0 0F 90 F2 20
$30A0: 00 BF 81 A9 30 8D FF 00
$30A8: 60 03 00 00 90 00 00 02
$30B0: 01 02 03 04 05 0B 1D 40
$30B8: 45 46 47 48 49 4A B1 4A
$30C0: 24 D4 18 EA 06 62 64 50
$30C8: 42 41 53 49 43 00 00 00
```

## DOBOOT.OBJ Hex-Dump

BSAVE DOBOOT.OBJ, A\$6400, L222

```
$6400: 20 AF 64 A9 99 85 01 A2
$6408: 30 2C 8B C0 2C 8B C0 A9
$6410: FF 20 00 01 A2 03 2C 83
$6418: C0 2C 83 C0 A9 D3 20 00
$6420: 01 A9 EE 8D 00 D0 2C 82
$6428: C0 A2 02 EE 01 01 A9 03
$6430: 20 00 01 CE 01 01 A2 01
$6438: A0 E0 A9 03 20 00 01 A2
$6440: 03 BD 4C 64 95 36 CA 10
$6448: F8 4C 00 E0 5D 64 50 64
$6450: A9 1B 85 38 A9 FD 85 39
$6458: A2 08 A9 8D 60 C9 DD D0
$6460: FB A9 F0 85 36 A9 FD 85
$6468: 37 A9 96 85 70 85 74 A9
$6470: A5 85 F2 20 2F FB 20 58
$6478: FC A2 07 BD D6 64 9D 00
$6480: 02 29 7F 9D 02 02 CA 10
$6488: F2 A9 AD 8D 00 02 A9 0C
$6490: 85 4B A9 D0 8D BC 08 A9
$6498: F0 8D C0 08 A9 60 8D C3
$64A0: 08 8D FF 08 20 92 08 90
$64A8: 03 20 50 64 4C 00 BE A0
$64B0: 17 B9 BE 64 99 FF 00 88
$64B8: D0 F7 84 00 84 02 60 8D
$64C0: 04 C0 85 03 B1 00 91 02
$64C8: C8 D0 F9 C6 01 C6 03 CA
$64D0: D0 F2 8D 04 C0 60 07 D3
$64D8: D4 C1 D2 D4 D5 D0 00 00
```

# Jetzt neu von SYBEX

**Schnell und effizient programmieren mit Apple II/II+/Ile**

**Was ist wo im Apple**

496 Seiten / 84 Abb.  
Best.-Nr. 3098  
ISBN 3-88745-098-1 (1985)  
DM 58,- / sFr 53,40 / S 452,-

Besitzer eines Apple II, II+ oder Ile erhalten eine systematische Arbeitshilfe, um schnelle und elegante Programme auf ihrem Rechner zu schreiben. Der Autor erläutert die Bedeutung von 2000 wichtigen Speicheradressen; dazu alle wichtigen ROM-Routinen. Für den schnellen Zugriff sind Routinen und Speicheradressen in alphabetischer und numerischer Reihenfolge sortiert.

Viele Apple-Routinen, die Sie in den eigenen Programmen leicht verwenden können, helfen Ihnen, Ihre BASIC-Programme zu beschleunigen und bei der Assemblerprogrammierung viel Arbeit und Zeit zu sparen.



**überall, wo es gute Computerbücher gibt**

## ccp datentechnik

### 640 KByte-Drives für den Apple //c!!

- 5/4- od. 3 1/2-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 35/40 Track
- Anschluß an die externe Laufwerkbusche
- Durch Einbauplatine (kein Löten) 640 KByte im Direktzugriff
- Einfache Anpassung für DOS 3.3, UCSD-Pascal und PRODOS durch menügeführten Patch
- Anpassung von CP/M in Verbindung mit einer Z 80-Zusatzplatine in Vorbereitung
- anschlussfertig im Gehäuse . . . . . **DM 1090,-**

### Festplatten für Apple II (//e)

- 5/4 Zoll-Format (Slimline)
- Booten direkt von der Festplatte in DOS 3.3, UCSD-Pascal, PRODOS und CP/M 2.2 / 3.0
- Gemischtbetr. mit 35/40/80/160 Track-Drives
- Copy- und Install-Programme im Lieferumfang
- Umfangreiches Manual
- z. B. 12 MB form, incl. Netzteil u. Contr., anschlussfertig an Ihren Apple . . . . . **DM 3698,-**

### 640 KByte-Drives für Apple II (//e)

- 5/4- od. 3 1/2-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 40 Track (Apple kompatibel)
- Installationssoftware für DOS 3.3, UCSD-Pascal, CP/M 2.2, CP/M 2.23 (60K), PRODOS, AP22, ALS CP/M+
- Umfangreiches Handbuch
- Anschlussfertige Auslieferung incl. Contr. und 2 Drives
- Diskstation 55II (2 Teac FD55-F, 1,2 MB) . . . . . **DM 1598,-**
- Diskstation 35II (2 Teac FD35-F, 1,2 MB) . . . . . **DM 1580,-**

### 80 Zeichen + 64 K für Apple //e

- und jetzt einsetzen . . . . . **DM 158,-**

## Alles für Ihren Apple

Info bei:

**ccp-datentechnik**

Herderstraße 12 - 2000 Hamburg 76  
Telefon 040/225676

**APPLE -- DISKETTEN LAUFWERKE**

Original Disk II m. Controller + DOS 3.3 + Hdbuch Original Disk II (2 Laufwerk)

**Duo-Disk Station** Slimline, 2 x 143KB Chinon Siemens Disk-Laufw., 143KB, m. Kabel + Gehäuse

**Ahor-Disk-Laufw.**, Slimline, 143KB, m. Kab. + Geh. Chinon Slimline, 143KB, Superleise, m. Kab. + Geh. Teac 55F, 1 MB unv. Kapazität, Shugartbus

**Teac 55F**, kpl.-m. Gehäuse, 40/80 Track, 1MBYTE anschlussfertig, m. Umschaltung 40/80 + Kabel

**Zweitlaufwerk für Apple II C**, 143 KB, mit Kabel

**APPLE -- INTERFACES + MAINBOARDS**

Disk Controller II Original o. kompat. Drives

**Super-Controller I** 2x Teac 55F, mit Software

**80 Zeichen-Karte II**, m. Softswitch, 2 Zeichensätze

**16K RAM Erweiterung** für II und kompatible

**2 80A Interface Karte für CP/M 2.2**

**Z 80B Interface Karte für CP/M 3.0**, m. 64K RAM

**Printer-Grafik-Interface**, Epson kompatibel

**Centronic-Parallell Text-Interface Karte**

**Printer-Grafik-Interface**, NEC/ITOH kompatibel

Anschluß Kabel I, Parallell + Grafik-Interface

**Epson Grafik-Interface**, benutzbar für Drucker

**Apple Nec/Itch/Okidata u. andere**, m. 32K Buffer

**Buffer-Interface** wie vor, aber mit 64K Buffer

Anschluß Kabel I, Buffer Interface Karte

**232C Serielle Interface Karte**

**Super Serielle Interface Karte**, Full Duplex

**Sprach (Speech) Karte**, Sprachwiedergabe

**6522 Parallell Interface Karte**

**Clock Karte (Datum/Uhrzeit)** Ein/Ausgabe

**Epson Writer Karte** (2716 - 2764)

**IEEE-488 Interface Karte**

**Logo-Karte** mit Diskette und Handbuch

**Musik Karte**, m. Diskette und Handbuch

**PAL Color Interface Karte** (UIIF + Video)

**RGB Interface Karte (f. Apple II + II+)**

**Wid. Karte (kopiert über RAM-Bereich)**

**128K RAM Erweiterung** Karte m. Patchsoftware

**256K RAM Erweiterung** Karte m. Patchsoftware

**8809 Prozessor Exell-9** Interfacekarte

**IC-Tester Interface Karte** (RAMS/TTL 54/74)

**Epson Writer Karte** (2716 - 2764)

**Hauptplatine 49K**, o. Firmware-Eprome, 8 Slots

**Testplatine 64K**, wie vor

**APPLE -- LEEERPLATINEN**

Leerplatine der obigen Interface Karten sind alle in vergoldeter, m. Bestuck-Audruck + Best-Plan Lieferbar

Leerplatinen mit der Kennzeichnung

Leerplatinen mit der Kennzeichnung

Leerplatinen mit der Kennzeichnung

Experimentier-Platine, für Apple Slot EX300

**Leerplatine Motherboard 48K**, mit Best-Audruck

**Leerplatine Motherboard 64K**, mit 6502 + Z80

**100% Apple-kompatibel bei Verwendung des Apple-Betriebssystems.**

**6 Mon. Garantiereparaturservice**



**APPLE - TASTATUREN + LEERGEHÄUSE + NETZTEILE - APPLE**

Standard Einbau Tastatur, ASCII, freibel. II Tasten, Doppelbelag, aller Tasten, Groß-/Kleinschr. Nr. N26

Tastatur wie vor, jedoch mit 15er Block Nr. N67

Tastatur Nr. N67, mit sep. Gehäuse kpl. m. Kabel

Separate Tastatur IBM-Look, anschlussfertig mit Kabel Dopp. belegt Tasten, frei prog. Funkt. Tasten

Separate Tastatur, Eprom progr. bar. Cursorblock

Tastenfeld mit 24 Funktionen, Deutsch o. ASCII

Leergehäuse Standard, passend f. Tastatur Nr. N26

dto. wie vor, jedoch passend für 15er Tast. N67

Leergehäuse wie Mewa 9000, für Einbau von zwei

Slimline-Laufwerken + Tastaturanschluß Plastic

Leergehäuse wie vor, jedoch in Metall-Ausführung

Schaltteile für APPLE u. kompatible Rechner

+5V/5 A -5V/0.5 A +12V/2.5 A -12V/0.5 A N74

+5V/7.5 A -5V/0.5 A +12V/2 A -12V/0.5 A N75

**IBM -- KOMPATIBEL**

Alle Teile unterliegen einer sorgfältigen Endkontrolle und wir übernehmen daher volle Garantie für die Funktionsfähigkeit, sowohl für die gelieferten Leerplatinen, als auch für die fertig bestückten Boards und Interface Karten, die fast ohne Ausnahme mit geschalteten IC's geliefert werden

**IBM - Kompatible Interface Karten und Mainboards - IBM**

Disk Controller Karte für 2 Disk-Drives 'KL-2020 DM 199,-

Color Video Board 'KL-2050 DM ab 599,-

Monochrome Video boards 'KL-2060 DM ab 299,-

RS 232 Interfac Karte 'KL-2074 DM 199,-

Centronics Parallell Interface Karte 'KL-2072 DM 148,-

Multi-IO Interface Karte, RAM-Bereich RS232, Parallell Clock, mit OK bestückt 'KL-2040 DM 398,-

dto. wie vor, jedoch mit 128K bestückt 'KL-2041 DM 489,-

dto. wie vor, jedoch mit 256K bestückt 'KL-2042 DM 578,-

152K RAM Karte, mit OK bestückt 'KL-2056 DM 248,-

dto. wie vor, mit 128K bestückt 'KL-2056 DM 238,50

Multi I/O Interface Board, RS232, Disk-Controller, Centronics, Clock, Joystick und Lightpen Port 'KL-2071 DM 568,-

Epson Writer Karte (bis 128K benutzbar) 'KL-2022 DM ab 998,-

Hardscr/Winchester Host-Adapter Karte 'KL-1008 DM 829,-

Mainboard XT Version, 8 Slots, 0 bestückt mit I Boot-Eprome, 6 Eprom-Plätze frei 'KL-1004 DM 649,-

dto. wie vor, jedoch mit 128K bestückt 'KL-1005 DM 739,-

dto. wie vor, jedoch mit 256K bestückt 'KL-1006 DM 829,-

Bei den mit \* gekennzeichneten Artikeln geht zum Lieferumfang ein Manual und 7 ein Schaltbild

**Apple II\* Kompatibel mit 80-Zeichen-Karte!**

**128 K., DM 898,-**

Zusatzfunktion zur Apple II

80-Zeichen-Karte + 64 16K RAM Erweiterung + Keyboard

Motherboard II, bestückt + gedruckt DM 509,-

IBM-Look Gehäuse für MEWA 9000 Serie Aufpreis DM 3998,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

IBM-Loock Gehäuse f. IBM 9000 Serie Aufpreis DM 3698,-

**COMPUTER CENTER CONEX**

5650 SOLINGEN 11 Postfach 11 02 06-9 P

Telefon (02 12) 7 54 49

**ERICH-WILLI MEYER**

6343 FRÖHNHAUSEN Postfach 70 11

Telefon (0 27 71) 3 50 71

## Kurzanleitung

1. (Kopie der) ProDOS-Systemdiskette (beliebige Version) mit PR#6 booten und dann CONVERT mit -CONVERT starten.  
Nun MKBOOT.BAS + MKBOOT.OBJ + DOBOOT.OBJ von Peeker-Sammeldisk (DOS-3.3-Format) auf (Kopie der) ProDOS-Systemdiskette übertragen.
2. Jetzt von ProDOS-Systemdiskette FILER mit -FILER starten und Leerdiskette formatieren.
3. Nun erneut ProDOS-Diskette booten und dann MKBOOT.BAS mit -MKBOOT.BAS starten, das seinerseits MKBOOT.OBJ und DOBOOT.OBJ automatisch einlädt.
4. Wenn Menü erscheint, formatierte Leerdiskette einlegen, Slot-Nummer 6 eingeben und Return drücken. Damit wird die formatierte Leerdiskette zu einer Fastboot-Diskette mit der Systemdatei "PBASIC" (PRODOS + BASIC.SYSTEM).
5. Nun zu Testzwecken PBASIC-Diskette mit PR#6 starten.

## MKBOOT.BAS

```

100 HIMEM: 8192: REM $2000
110 GSYS = 12288: REM $3000
120 MODB00 = 12291: REM $3003
130 TEXT : HOME
140 HTAB 8: PRINT "ProDOS Fastboot-Erstellung"
150 FOR X = 1 TO 40: PRINT "=": NEXT
160 PRINT CHR$(4)"PREFIX /"
170 IF PEEK(GSYS) + PEEK(GSYS + 1) = 82 THEN 200
180 PRINT CHR$(4)"BLOAD MKBOOT.OBJ": REM A$3000
190 PRINT CHR$(4)"BLOAD DOBOOT.OBJ, A$3400": REM (!)
200 VTAB 5: HTAB 7: SL = 6: REM Default
210 PRINT "Slotnummer des Ziel-Drives: ", SL
220 VTAB 5: HTAB 35: GET X$
230 IF X$ = CHR$(13) THEN 260
240 PRINT X$: IF X$ < "1" OR X$ > "7" THEN 220
250 SL = VAL(X$)
260 VTAB 8: HTAB 5
270 PRINT "Zieldiskette in Drive 1 einlegen."
280 PRINT : PRINT : HTAB 6
290 PRINT "<RETURN>=Start <ESC>=Ende"
300 VTAB 11: HTAB 22: GET X$
310 IF X$ = CHR$(27) THEN 450
320 IF X$ < > CHR$(13) THEN 300
330 POKE 34,13: HOME : POKE 34,0
340 PRINT CHR$(4)"PREFIX, D1, S": SL:
    REM Setzt "Last Unit" in SGP
350 PRINT CHR$(4)"PREFIX /"
360 CALL GSYS: REM Kopiert aus allen Banks nach $3400..
370 PRINT "Das System wird geschrieben...": PRINT
380 PRINT CHR$(4)"BSAVE PBASIC, A$3400, E$8FFF"
390 PRINT "Der Umlader wird modifiziert...": PRINT
400 UNIT = 16 * SL: POKE 766, UNIT:
    REM Für BLOCKREAD/WRITE
410 CALL MODB00: REM Read Block 00, Mod & Rewrite
420 IF PEEK(767) = 0 THEN PRINT "Ok.": GOTO 260
430 PRINT CHR$(7)"I/O ERROR!": GOTO 450
440 REM Programmende
450 CLEAR : HIMEM: 38400: END
460 REM Arne Schäpers 5/85

```

## MKBOOT.OBJ

BSAVE MKBOOT.OBJ, A\$30000, L\$00CD

```

1 * MKBOOT.OBJ
2 * Fastloader-Erstellung
3 *
4 SPTR EQU $00
5 TPTR EQU $02 ;Ziel-Pointer
6 *
7 MOVER EQU $100 ;Kopier-Routine
8 BUFFER EQU $9000 ;Puffer für MODB00
9 *
10 ORG $3000
11 *
12 JMP MOVESYS ;dezimal 12288
13 JMP MODB00 ;dezimal 12291
14 *
15 MOVESYS JSR COPYMV ;kopiert MOVER nach $100
16 LDA #$BF ;zuerst BASIC.SYSTEM und
17 STA SPTR+1 ;beide Global Pages
18 LDA #$8F ;nach $6A00..8FFF
19 STA TPTR+1
20 LDX #$26
21 JSR MOVER
22 LDA #$FF ;darunter kommt das MLI
23 STA SPTR+1

```

```

24 LDX #$30 ;$3000 Bytes
25 BIT $C08B ;READ von LC Bank 1
26 JSR MOVER ;nach $3A00...
27 LDA #$D3 ;REBOOT von $D100..D3FF
28 STA SPTR+1
29 LDX #$03 ;$300 Bytes
30 BIT $C083 ;READ von LC Bank 2
31 JSR MOVER ;nach $3800...
32 BIT $C082 ;ROMON
33 LDA #$03
34 STA SPTR+1 ;RAM-Disk von $200..$3FF AUX
35 LDX #$02 ;$200 Bytes
36 INC MOVER+1 ;READ von AUX
37 JSR MOVER
38 DEC MOVER+1 ;zurück auf READ von MAIN
39 LDA #$03 ;Page-3-Vektoren
40 STA SPTR+1
41 LDX #$01 ;nur eine Speicherseite
42 LDY #$E0 ;von $3E0..3FF
43 JSR MOVER ;nach $34E0..34FF
44 *
45 LDX #3 ;die Redirection-Vektoren
46 LDA #0 ;in der Global Page des
47 ZVSYIS0 STA $8E38,X ;BASIC.SYSTEM werden
48 DEX ;auf Null gesetzt
49 BPL ZVSYIS0
50 RTS ;zum BASIC-Programm
51 *
52 COPYMV LDY #$1B ;kopiert MOVER nach $100
53 CPML LDA MVLOC-1,Y
54 STA MOVER-1,Y
55 DEY
56 BNE CPML
57 STY SPTR ;SPTR= $xx00
58 STY TPTR ;TPTR= $xx00
59 RTS
60 *
61 MVLOC STA $C002 ;RDMAIN
62 STA $C004 ;WRMAIN
63 MVL LDA (SPTR),Y
64 STA (TPTR),Y
65 INY
66 BNE MVL
67 DEC SPTR+1
68 DEC TPTR+1
69 DEX
70 BNE MVL
71 STA $C002 ;RDMAIN
72 STA $C004 ;WRMAIN
73 RTS
74 *
75 *
76 MODB00 LDA $2FE ;von BASIC: Unitnummer
77 STA RWUNIT ;=> Pblock für Read/Write
78 *
79 JSR $BF00
80 DFB $80 ;READ BLOCK
81 DA PBLOCK
82 BNE IOERR
83 *
84 LDX #0
85 DOMOD LDY MODOFF,X ;Offset zur ersten Patch-Adresse
86 LDA NEWOP,X ;neues Byte
87 STA BUFFER+$BE,Y
88 INX
89 CPX #LASTOP-NEWOP ;Tabellenende?
90 BCC DOMOD
91 *
92 JSR $BF00
93 DFB $81 ;WRITE BLOCK
94 DA PBLOCK
95 IOERR STA $2FF ;Flag für BASIC
96 RTS
97 *
98 PBLOCK DFB 03 ;3 Parameter
99 RWUNIT DS 1 ;Unitnummer für READ/WRITE
100 DA BUFFER ;Ziel/Quellpuffer
101 DA 0000 ;Blocknummer: $00 00
102 *
103 MODOFF DFB 0,1,2,3,4,5
104 DFB $C9-$BE
105 DFB $DB-$BE
106 DFB $FE-$BE
107 DFB $45,$46,$47 ;=> $903: neuer Filename
108 DFB $48,$49,$4A
109 *
110 NEWOP LDA ($4A),Y ;für "AND #$F0
111 BIT $D4 ;wird später zu "BEQ $896"

```



```

112 CLC ;späterer Exit: STARTUP gefunden
113 NOP ;wird dann zu RTS
114 DFB 06 ;Filetype: BIN, nicht SYS ($FF)
115 DFB $62 ;Startadresse für Indexblock
116 * ;(nicht $1E)
117 DFB $64 ;Startadresse für Programm
118 * ;(nicht $2000)
119 ASC 'PBASIC' ;anstelle von 'PRODOS'

120 LASTOP EQU * ;Tabellenende

205 Bytes

```

### DOBOOT.OBJ

BSAVE DOBOOT.OBJ, A\$6400, L\$00DE

```

1 * DOBOOT.OBJ
2 * Prodos-Fastloader
3 *
4 SPTR EQU $00 ;Quell-Pointer
5 TPTR EQU $02 ;Ziel-Pointer
6 *
7 MOVER EQU $100 ;Kopier-Routine
8 *
9 ORG $6400 ;wird hierher vom Boot geladen
10 *
11 JSR COPYMV ;kopiert MOVER nach $100
12 LDA #$99 ;oberste Quell-Seite ist $9900
13 STA SPTR+1
14 LDX #$30 ;$30 Speicherseiten
15 BIT $C08B
16 BIT $C08B ;WRITE ENABLE Bank 1 LC
17 LDA #$FF ;oberste Ziel-Seite: $FF00
18 JSR MOVER ;kopiert das MLI in Bank 1
19 LDX #$03
20 BIT $C083
21 BIT $C083 ;WRITE ENABLE Bank 2 LC
22 LDA #$D3
23 JSR MOVER ;kopiert REBOOT in Bank 2
24 LDA #$EE
25 STA $D000 ;BANKID für Bank 2
26 BIT $C082 ;ROMON
27 LDX #$02 ;RAM-Disk: 2 Seiten, obere $300
28 INC MOVER+1 ;"STA WRMAIN" => "STA WRAUX"
29 LDA #03
30 JSR MOVER ;kopiert RAM-Disk nach AUX
31 DEC MOVER+1 ;zurück auf WRMAIN
32 LDX #$01 ;Page-3-Vektoren
33 LDY #$E0 ;von $3E0..$3FF
34 LDA #03
35 JSR MOVER
36 *
37 LDX #03
38 SETIOV LDA IOVECS,X ;COUT zeigt danach auf GETBAS,
39 STA $36,X ;KEYIN auf "FAKECR"!
40 DEX
41 BPL SETIOV
42 JMP $E000 ;Kaltstart Applesoft
43 IOVECS DA GETBAS
44 DA FAKECR
45 *
46 FAKECR LDA #$1B ;KEYIN zeigt hierher!
47 STA $38 ;$38/39 auf Monitor
48 LDA #$FD
49 STA $39
50 LDX #08 ;"8 Zeichen empfangen"
51 LDA #$8D ;"jetzt <CR>"
52 DDRTS RTS
53 *
54 GETBAS CMP #DDD ;Applesoft Prompt?
55 BNE DDRTS ;noch nicht (ist <CR>)
56 LDA #$F0 ;Applesoft versucht nach dem
57 STA $36 ;Kaltstart, ein Prompt zu druck-
58 LDA #$FD ;ken und wird hier abgefangen
59 STA $37 ;COUT jetzt auf FDF0 (Monitor)
60 *
61 LDA #$96
62 STA $70 ;HIMEM für Applesoft:
63 STA $74 ;$9600 (dezimal 38400)
64 LDA #$A5
65 STA $F2 ;Trace-Flag
66 *
67 JSR $FB2F ;TEXT (40*24)
68 JSR $FC58 ;HOME
69 *
70 LDX #$07

```

```

71 SETNAME LDA FNAME,X
72 STA $200,X ;"STARTUP" nach $201
73 AND #$7F
74 STA $902,X ;und für die File-Suche im Boot
75 DEX
76 BPL SETNAME
77 LDA #"-."
78 STA $200 ;=> "-STARTUP"
79 *
80 LDA #$0C ;das Volume-Directory wurde
81 STA $4B ;vom Urlader ab $C00 geladen,
82 LDA #$D0 ;BPL im Vergleich wird zu
83 STA $8BC ;"BNE" - kein Vergleich STYPE!
84 LDA #$F0 ;BIT $D4
85 STA $8C0 ;=> BEQ $896: File gelöscht und
86 LDA #$60 ;"RTS" am Ende der Suchroutine
87 STA $8C3 ;RTS für "STARTUP gefunden"
88 STA $8FF ;RTS für "nicht gefunden"
89 JSR $892 ;sucht VolDIR nach STARTUP ab
90 BCC GOTSTUP ;"STARTUP" gefunden
91 JSR FAKECR ;KEYIN-Vektoren auf Monitor
92 GOTSTUP JMP $BE00 ;Kaltstart BASIC.SYSTEM
93 *
94 *
95 COPYMV LDY #FNAME-MVLOC
96 MVC1 LDA MVLOC-1,Y ;kopiert MOVER nach $100
97 STA MOVER-1,Y
98 DEY
99 BNE MVC1
100 STY SPTR ;= 00
101 STY TPTR ;= 00
102 RTS
103 *
104 MVLOC STA $C004 ;WRITE nach MAIN
105 STA TPTR+1 ;Zieladresse (high)
106 MV1 LDA (SPTR),Y
107 STA (TPTR),Y
108 INY
109 BNE MV1
110 DEC SPTR+1 ;Quelle - $100
111 DEC TPTR+1 ;Ziel - $100
112 DEX ;Seitenzähler
113 BNE MV1
114 STA $C004 ;WRITE nach MAIN
115 RTS
116 *
117 FNAME DFB 07 ;Namenslänge
118 ASC "STARTUP"

```

222 Bytes



Apple II+/e

**256K RAM-KARTE**

NEU

\* abwärtskompatibel zur Saturn 128K  
 \* für alle Slots verwendbar  
 \* incl. Software RAMDISK (2\*496 Sekt.) DOS 3.3  
 \* incl. Software RAMDISK Pascal, Prodos, CP/M  
 \* erweiterbar auf 512K

**495.-DM**

**512K RAM-KARTE incl. 4 Disketten 795.-DM**  
**1M RAM-KARTE incl. 4 Disketten 1395.-DM**

**16K AKKU-RAM-KARTE**

\* Kompatibel zur 16K Language Card  
 \* Datensicherung nach Abschalten  
 \* PSEUDO-ROM (z.B. Rom-Edit etc.)  
 \* leichter Einbau  
 \* für alle Slots verwendbar  
 \* incl. Software DOS 3.3 Utilities

**175.-DM**

**128K AKKU-RAM-KARTE incl. 4 Disketten 795.-DM**

Alle Preise incl. 14% MwSt; Info Gratis; Erhältlich über den Fachhandel oder  
 Direktversand per Nachnahme zuzüglich Porto und Verpackung bei  
 Ing. Büro M. Fricke Neue Str. 13 1000 Berlin 37 Tel: 030/601 56 52

## Verkauf Software

\*\*\*\*\* WARGAMES \*\*\*\*\*

Die Besten von SSI für APPLE C 64 ATARI. Info gegen 80 Pfg. RP Computer-Service Th. Müller Postfach 2526 7600 Offenburg

**Vereins-Manager**, Bibliotheks-Chef u. a. Profi-Software von E. Heinz Waldgürtel 7, 5060 Berg. Gld. 1

**CAD-Programm ROBO 500** + PLOTTER-SOFTWARE 500/1000 kompl. m. HARDWARE. Test s. APPLE'S 7/8 NP DM 3300 VHB. DM 1900 Tel. 07 11/52 31 54

**PASCAL-Textdump** f. All 79,- LISP-Interpreter f. Mac 249,- Thomas Maier, Mühlweg 24, 6360 Friedberg 1, 0 60 31/9 16 50

**ASTRO-Programme** für APPLE: RADIX Solar/Composite/Partnervergleich mit Plotter-Routinen für: HP, EPSON Hi-80, WATANABE, u. andere  
Liste anfordern von ROJASOFT Postfach 4461 CH-8022 Zürich

**MACINTOSH-SOFTWARE.**  
US-Import Preisliste + Gratisinfo SC-INFO-1, PF 2013, 5100 Aachen

**Schreibmaschinenübungsprogramm.** Verbessern der Fingerfertigkeit! Durch Zeit- u. Fehlerkontrolle für 2e m. 80Z-Karte; 50 DM VS, NN Patz, 6082 Walldorf/2, Nordendstr. 50, Tel. 061 05/7 45 61 ab 19 Uhr

\*\*\*\*\*  
\* **LAND.** \*  
\* **SPITZENPROGRAMME** \*  
\* FÜR APPLE II \*  
\* AB 190 DM INFOMAPPE \*  
\* 3.50 DM \*  
\* A. Wachendorf, \*  
\* 2814 Engeln 30 \*  
\*\*\*\*\*

## Ankauf Software

**Suche MERLIN-Assembler** (DOS 3.3) nachmittags, Tel. 071 59/36 02

## Verkauf Hardware

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

**ERPHI-SUBSYSTEM**  
**2x640KB** 1900,00 ERPHI-Controller 280,00 DM Phillips 2x80 Track 440,00 DM 35/40 TRACK LAUFWERK 430,00 DM M. Killermann 09 11/39 67 72 + 33 30 31

**\*APPLE II\*I/O-Probleme?\***  
& zusätzliche Slots, per Software schaltbar, ohne komplexe Steuer-softw. 2Platinen, 1a Ind. qualität unbest. DM 180,- Info: E. Tausendpfund Gonsenheimer Spieß 18, 65 Mainz

**Distar-Laufwerk** incl. Kabel 439 DM Contr. 97 DM; Touch Tabl. + SW 198 DM APPLE IIe Komat. + 80 Zei. + 128K 1398 DM. Robert Hartmann \*06 81/6 63 93\* EDV + Zubehör Mainzer Str. 102 6600 Saarbrücken 3

**Z80A-Appli-Card** (PCPI, 4MHz, 64KB, CPIM 2.2), 10 Diskseiten für 600 DM, Tel. 05 31/84 46 49

**APPLE-II-Kompatibler**, 48K, 16K-Karte, 80Z/Z, Tastatur/ 10er Block Gehäuse, Monitor, Controller VB 1100,- Tel. 07 11/42 58 90

**apple IIc mit CPM/2.** LW/ Mouse/2M. alt/Software/Literatur/ DM 4200,- Tel. 09 31/8 63 72

**IBS Z-80B Karte** 590 DM, 68000 Ka. 799 DM, 256 KRAM 599 DM 05 21/87 04 24

**APPLE IIe 2 Floppys** Monitor LITERA Div. Karten Software günstig. 0 41 06/7 19 70

**Erscheinungstermin für Ausgabe 10/85 ist am 23. 9. 1985**

## 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. 0761/27 68 64, Tx. 772642aaad

**Suche Apple IIc Reference**  
Manual Manfred Rost Tel. 02 11/33 58 42

012345678912012345678901234  
**256-K-Karte** für Apple als RAM-Floppy günstiger wie Laufwerk. Betriebsbereit mit Software nur **349,- DM** FUCOM GmbH, Postfach 483, 4600 Dortmund 1.

## Tausch

**Suche f. Apple IIc** Dbase Compiler, ROBO 1500, C Helper u. Microcap, Software z. Tausch vorh. Tel. 02 28/37 44 74

## Einkaufsführer



Keithstr. 26 · 1 Berlin 30 · ☎ 0 30-26 111 26



Bachstr. 104 · 2 HH 76 · ☎ 0 40-220 11 55

Für weitere Informationen zu einem der in dieser Ausgabe vorgestellten Produkte stehen Ihnen die Produktkarten zur Verfügung

Bitte verwenden Sie für Kleinanzeigen die vorgedruckten Antwortkarten in diesem Heft.

## 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, Postfach 10 28 69, 6900 Heidelberg 1 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 1

## 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



## Abo-Karte

Ja, ich möchte **peeker** abonnieren.

Liefere Sie mir **peeker** ab Ausgabe ..... (1985 erscheinen 11 Ausgaben – 1 Doppelnummer) zum Jahresbezugspreis von DM 72,- (Inland) incl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt DM 72,- incl. MwSt., zzgl. DM 16,80 Versandkosten.

Ich wünsche jährliche Berechnung durch:

- Verlagsrechnung       Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank / PostA \_\_\_\_\_

Bankleitzahl \_\_\_\_\_ Kto.-Nr. \_\_\_\_\_

Datum \_\_\_\_\_ Unterschrift \_\_\_\_\_



## Buch-Shop

Bitte senden Sie mir gegen Rechnung folgende Bücher:

Menge	Autor, Titel	à DM	gesamt DM

Datum \_\_\_\_\_ Unterschrift \_\_\_\_\_



## Software-Karte

Bitte senden Sie mir gegen Rechnung folgende Apple-Programme:

- |  |   |
|--|---|
| <input type="checkbox"/> Peeker-Sammeldiskette, einzeln<br>Disk# _____, Disk# _____<br>Disk# _____, Disk# _____<br>Preis je Disk DM 28,- (einzeln) | <input type="checkbox"/> Apple DOS 3.3, Begleitdiskette, DM 28,-        |
| <input type="checkbox"/> Peeker Sammeldiskette, im Fortsetzungsbezug<br>ab Disk # _____<br>(Mindestbezug 6 Disketten)<br>Preis je Disk DM 20,-     | <input type="checkbox"/> Apple ProDOS, Band 1, Begleitdiskette, DM 28,- |
| <input type="checkbox"/> CP/M ja <input type="checkbox"/> CP/M nein  | <input type="checkbox"/> Apple ProDOS, Band 2, Begleitdiskette, DM 28,- |
| <input type="checkbox"/> Pascal ja <input type="checkbox"/> Pascal nein  | <input type="checkbox"/> Apple Assembler, Begleitdiskette, DM 28,-      |
|  | <input type="checkbox"/> ProDOS-Editor 1.0, Programm, DM 98,-           |
|  | <input type="checkbox"/> MMU 2.0, Programm, DM 98,-                     |
|  | <input type="checkbox"/> INPUT 2.0, Programm, DM 98,-                   |
|  | <input type="checkbox"/> Softbreaker 1.0, Programm, DM 48,-             |
|  | <input type="checkbox"/> DB-Meister, Programm, DM 290,-                 |
|  | <input type="checkbox"/> Superplot, Programm, DM 48,-                   |
|  | <input type="checkbox"/> Superquick, Programm, DM 48,-                  |

Datum \_\_\_\_\_ Unterschrift \_\_\_\_\_





# Abo-Karte

Name \_\_\_\_\_

Firma \_\_\_\_\_

Abteilung \_\_\_\_\_

Straße \_\_\_\_\_

PLZ/Ort \_\_\_\_\_

**Vertrauensgarantie:**  
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1 innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

Datum \_\_\_\_\_

Unterschrift \_\_\_\_\_

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



# Buch-Shop

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

Postfach 10 28 69

6900 Heidelberg 1

## POSTKARTE

**peeker**  
Versandbuchhandlung

Postfach 10 28 69

6900 Heidelberg 1

## POSTKARTE

**peeker**  
Softwareabteilung

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) – Ctrflflag – 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

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

# MICROMINT



## MICROMINT VOLLTREFFER

**LASAR 16**  
IBM 256 K, 2 x TEAC B FDD, Contr. color Graphik, Multifunktionscard; Tastatur, Monitor  
Netzteil 15 A **4.678,-**

**LASAR ZE**  
- Apple comp. 64 K + 12 K ROM + 6502 + Z 80 A 80 Z sw Tastatur **1.290,-**

**Außerdem volles Rückgaberecht innerhalb 14 Tagen ohne Begründung.**

	Apple	IBM
● Mehrzweckklappgehäuse lt. Abb.	147,-	147,-
● Schaltnetzteile Apple 5 A/IBM 15 A	115,-	238,-
● Profitastatur dtsh. LASAR 2000	291,-	291,-
● Interface ab	75,-	148,-
● Monitor 22 Mhz incl. Fuß, bernstein	289,-	289,-

Kaufgarantie/Tiefstpreisgarantie/1A Qualität: 100 % kompatibel inkl. Systemsoftware  
Made by Micromint: Apple II 495,-, Apple IIe 695,-, IBM 895,- Fertigungslinien. Tragbare Gehäuse für 7-Zoll/9-Zoll-Monitore 595,- DM inkl. Tastaturen. Winchester 27 MB auf Anfrage 1A First Class Controller bis 140 MB 935,- DM.

**Generalimporteur MICROMINT Computer GmbH**  
Hochdahlr Straße 151, 4006 Erkrath 2  
Telex 8589305 mcm

☎ **0 2104/3 30 24**



**UNIVERSAL**

**KEYBOARDS**

Modell AN95FTE... DM 448,- ohne MwSt. (DM 510,72 incl. MwSt.)  
Die KEYBOARDS SPEZIELL angepaßt für den APPLE IIe  
Händleranfragen erwünscht



- FLEXIBEL — Jede Taste frei im EPROM programmierbar in bis zu 8 Ebenen im mitgelieferten EPROM
- PROFESSIONELL — Für Anwender mit gehobenen Ansprüchen
- ERGONOMISCH — Nach DIN ULTRAFLACH gestaltetes stabiles Gehäuse
- KOMPLETT — Tastatur, Gehäuse und Kabel fertig montiert und getestet. Durch Spezial-Kabel und Spezial-EPROM sofort einsteckfertig.
- KOMPAKT + FLACH — Durch Einsatz von „SIEMENS“ Flachstastmodulen



D-4930 Detmold ★ Alter Mühlenweg 5  
Telefon 0 52 31/41 76 ★ Telex 9 35 660 acs d

### Apple und IBM kompatible Computer

16K, Z80, Diskcontroller je	85,-
80 Zeichenkarte mit Softswitch	
2 Zeichensätze	160,-
Motherboard 48K ohne	
Firmware	550,-
Erphi-controller mit Autopatch	300,-
Siemenslaufwerk F 122	515,-
TEAC FD-55B 2 x 40 Track	448,-
TEAC FD-55F 2 x 80 Track	475,-

<b>Drucker Star SG 10</b>	<b>940,-</b>
Monochrome Monitore	ab 375,-
Farbmonitore	ab 998,-
Tastaturen für IBM und Apple	ab 330,-

Versand nur per Nachnahme oder Vorkasse  
Weiteres Zubehör für Apple und IBM gegen frankierten Rückumschlag.

**Preissenkung:**  
**128K Karte** (Saturn kompatibel) **375,-**  
**Preissenkung 4164-200 ns**  
Mindestabnahme 20 Stck. **3,90**

**Ulf Mohwinkel Electronic**  
Berliner Straße 73 Pf: 250 166  
5090 Leverkusen Fettehenne  
Telefon 02 14/937 61



### Preisliste kostenlos! Katalog DM 2,-

Unser Angebot im August

**Profimax III**, APPLE II kompatibel, mit Z80 und 6502, IBM-look nur 1248,-

**Profimax IIe**, APPLE IIe kompatibel, IBM-look nur 1598,-

**Chinon-Laufwerk F051A**, mit Kabel und Gehäuse, anschlussfertig nur 398,-

**STAR sg-10**, schönschreibfähig nur 898,-  
Druckerinterface, grafikfähig nur 118,-

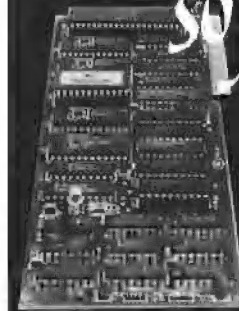
### D.O.S. Computersysteme

Am Kühnbach 42, 7170 Schwäbisch Hall 11  
Telefon (0791) 5 17 36

## Apple zu langsam?

Die TempoHexe ("Speedemon") macht den Apple II, II+ und IIe so schnell wie die populären 16bit Rechner. Die TempoHexe beschleunigt alle Programme bis zu 3 1/2 mal - absolut ohne jede Software-Änderung! Einfach einstecken, einschalten... los.

Mehr Informationen und Händleranfragen bei:



*softline*

R. Alverdes  
Schwarzwaldstr. 8a  
7602 Oberkirch  
Tel.: (078 02) 37 07  
Telex: 752637 sfe d

Katalog gegen DM 1,- in Briefmarken.

Wir haben die neuste Software und Peripherie für den Apple II, II+, IIe und Mac.

## APPLE-II kompatibles

Info geg. DM 1,40 in Briefmarken  
**SPRINGMANN COMPUTER GmbH**  
Stöckener Str. 199  
3800 Hannover 21  
Tel: 0511-791111 Tlx: 921466 comps d

PC-48 (europus) DM **799**

**Traum-Preise**

PC-48 Kombi-Preis nur **999**  
+DISTAR-Diskdrive +Controller DM

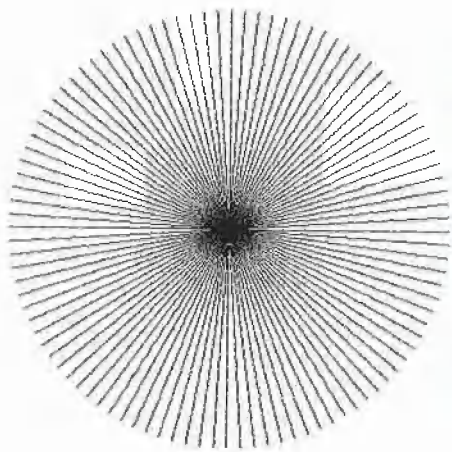
PC-64 (europus+16K) DM **899**

**Traum-Preise**

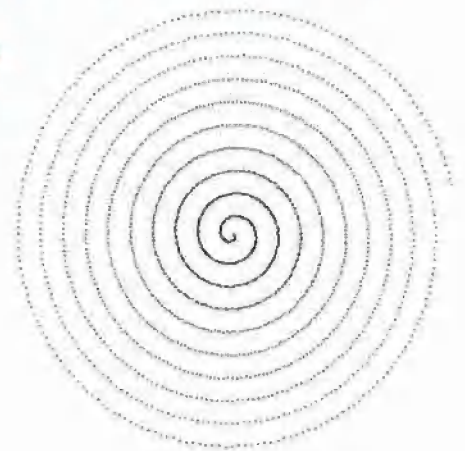
PC-64 Kombi-Preis nur **1099**  
+DISTAR-Diskdrive +Controller DM

**DISTAR-Drive** für alle II-Typen

IIc... DM **439**  
IIc... DM **457**  
f. IBM-PC (360K) DM **469**



# Super-HGR für NEC- und ITOH- Drucker



von Reiner Hammerschmidt

Angeregt durch die Grafik-Möglichkeiten einiger größerer Computer entstand der Wunsch, höhere Grafik-Auflösungen mit dem Apple zu realisieren. Dieser hat zwar die Möglichkeit, hochauflösende Grafiken darzustellen, dies aber nur mit maximal  $280 * 192$  Punkten bzw.  $560 * 192$  Punkten beim IIc oder IIe mit entsprechender Karte. Ein Bild mit noch höherer Auflösung ist mit dem nachfolgenden Programm **SUPER.HGR** und einem Matrixdrucker mit Einzelnadelansteuerung möglich.

Als Hardware reichen ein Apple II und ein grafikfähiger Matrixdrucker mit passendem Interface aus. Eine noch weitaus größere Auflösung als hier ( $640 * 320$ ) ist erzielbar, wenn man mit dem Programm einen sogenannten DOS-Mover verwendet.

Zur einfacheren Ausgabe liegt die Grafik im Speicher zeilenweise druckergerecht vor. Die Auflösung wird in Zeile 150 durch die Variablen LE (horizontal) und ZE (vertikale Punktzahl =  $ZE * 8$ ) festgelegt. Die hier angegebenen Werte gelten für einen **NEC 8023**. Die Variablen LE und LE\$ müssen bei anderen Druckern eventuell angepaßt werden (z.B. bei weniger als 640 Punkten pro Zeile). LE\$ enthält die Initialisierungsfolge für den Drucker, um eine Grafikzeile byteweise ausgeben zu können. Der hier verwendete String besagt, daß der Drucker auf Grafik umschalten und 640 Bytes pro Zeile als Grafik ausdrucken soll.

Um bei Druckeranpassungen zu bleiben: Der NEC druckt im Grafikmodus Bit 0 oben. Da es aber auch Drucker gibt, die Bit 0 unten drucken, muß hierfür die Zuweisung an die Variable Z2% in Zeile 480 folgendermaßen lauten:

$$Z2\% = 2 \uparrow (7 - (ZZ-Z1\%) * 8)$$

Ebenso muß die Einstellung des Zeilenabstandes in Zeile 530 (16/144 Inch) sowie die Grafikinitialisierung in Zeile 550 bei Druckern außer NEC 8023 bzw. ITOH 8510 angepaßt werden.

## Zur Funktion des Programms

Die Kernroutine basiert auf den Zeilen 480-510. Um auch bestehende Grafikprogramme weiterverwenden zu können, brauchen nur X- und Y-Koordinaten des zu setzenden Punktes bestimmt und übergeben werden. Diese Werte müssen innerhalb der vorgegebenen Grenzen liegen, da keine Abfrage eingebaut ist, welche Fehler abfangen könnte. Der Koordinatenursprung liegt, wie gewohnt, links oben. Die ermittelten X-Y-Werte werden mittels der Variablen I (Abszisse) und Y% (Ordinate) an das Unterprogramm 480ff. übergeben, welches die absolute Adresse (N1% und N2%), sowie die Pixelwertigkeit des Punktes im Byte (Z2%) bestimmt. Das hierdurch eindeutig bestimmte Bit wird dann durch eine Maschinenroutine in den Speicher „hineingeODERT“ (CALL 788). Das Programm stellt in der abgedruckten Form eine Demonstration dar und soll zu

weiteren Experimenten anregen. Nach Start mit RUN löscht es den Grafik-Speicher und fragt nach der gewünschten Grafik, die durch Modifikation einzelner Parameter im Programm variiert werden kann. Nach Eingabe der entsprechenden Ziffer entsteht die Grafik, zunächst unsichtbar, bis der Ausdruck erfolgt. Weitere Ausdrücke sind möglich durch Eingabe von GOTO 530. Die Zeilen 170-450 stellen somit Anwendungsbeispiele der Super-HGR dar: Zunächst wird der Maschinensprachteil gepokt (630ff.; auf Peeker-Sammeldisk unter dem Namen **SUPER.HGR.ASM**), und der Grafik-Speicher oberhalb von HIMEM wird gelöscht. Anschließend wird die gewünschte Grafik berechnet, wobei noch lange nicht alle Möglichkeiten im Programm ausgeschöpft sind (siehe Beispiele). Man denke z.B. an hochauflösende Darstellungen von 3D-Funktionen. Abschließend erfolgt der Ausdruck der erstellten Grafik.

Da Applesoft-BASIC bei Ausgaben grundsätzlich Bit 7 auf Null setzt, mußte auch hierfür ein kleines Maschinenprogramm verwendet werden, daß über USR() aufgerufen wird. Dieses erwartet ein Druckerinterface in Slot 1 und muß für andere Interfaces gegebenenfalls angepaßt werden.

Das Programm wurde bewußt soweit wie möglich in Applesoft geschrieben, um sehr leicht Änderungen vornehmen zu können.

## SUPER.HGR

```

100 REM Super-HGR für APPLE II
110 REM *****
120 REM *      Reiner Hamerschmidt      *
130 REM *      Oktober/1984             *
140 REM *****
150 LET PI = 3.1415926:LE = 640:LES = "S0640":ZE = 40
160 HM = 8200: HIMEM: HM
170 GOSUB 630: GOSUB 610
180 HOME : PRINT "Super HGR"
190 VTAB 9: PRINT "1 Sinus"
200 PRINT : PRINT "2 Spirale"
210 PRINT : PRINT "3 Stern"
220 VTAB 22: PRINT "Ihr Wunsch : ": GET AS: PRINT AS
230 IF AS < "1" OR AS > "3" THEN 220
240 ON VAL (AS) GOTO 270,350,410
250 END
260 REM ----- Sinus (X*X) -----
270 FOR I = HM TO HM + LE - 1
280 LET B = (2 * PI) / LE
290 LET Y% = ( SIN (B * (I - HM) * (I - HM) * B) + 1)
    * ((ZE * B) - 1) / 2
300 LET ZZ = Y% / 8: LET Z1% = INT (ZZ):
    LET Z2% = (ZZ - Z1%) * 8
310 LET PT% = PEEK (I + Z1% * LE)
320 POKE I + Z1% * LE, ((PT% OR 2 ↑ Z2%) * 2 ↑ Z2%) + PT%
330 NEXT I: GOTO 530
340 REM ----- Spirale -----
350 FOR WW = 0 TO 20 * PI STEP PI / 100
360 LET G = 159 * WW / (20 * PI)
370 LET I = HM + 1.1 * COS (WW) * G + 320
380 LET Y% = SIN (WW) * G + 160
390 GOSUB 480: NEXT WW: GOTO 530
400 REM ----- Stern -----
410 FOR WW = .00001 TO 2 * PI STEP PI / 10
420 FOR G = 1 TO 160
430 LET I = HM + 1.1 * COS (WW) * G + 320
440 LET Y% = SIN (WW) * G + 160
450 GOSUB 480: NEXT G: NEXT WW: GOTO 530
460 REM -----UP: setze Punkt -----
470 REM I= X-Koordinate , Y%= Y-Koordinate
480 LET ZZ = Y% / 8: LET Z1% = INT (ZZ):
    LET Z2% = 2 ↑ ((ZZ - Z1%) * 8)
490 N1% = (I + Z1% * LE) / 256:N2% = (I + Z1% * LE) -
    N1% * 256
500 POKE 790,N1%: POKE 795,N1%: POKE 789,N2%: POKE 794,N2%

```

```

510 POKE 792,Z2%: CALL 788: RETURN
520 REM -----Ausdruck-Routine für NEC 8023-----
530 PRINT CHR$ (4)"PR#1": PRINT CHR$ (27)"T16":
    REM Zeilenabstand 16/144 inch
540 FOR J = 0 TO ZE - 1
550 PRINT CHR$ (27):LE$: REM Vorbereiten Grafik (NEC)
560 FOR I = HM + J * (LE) TO HM + (J + 1) * LE - 1:
    WY = USR ( PEEK (I))
570 NEXT : PRINT : NEXT
580 PRINT CHR$ (4)"PR#0"
590 END
600 REM ----- Bereich löschen -----
* 610 FOR I = HM TO HM + ZE * LE: POKE I,0: NEXT : RETURN
620 REM --- Assembler Unterprogramm poken ---
630 FOR I = 768 TO 796: READ A: POKE I,A: NEXT
640 POKE 10,76: POKE 11,0: POKE 12,3: REM USR-Vektor
650 RETURN
660 REM UP für 8 Bit Daten
670 DATA 32,12,225,165,161,44,145,192,48,251,141,
    145,192,141,146,192,141,148,192,96
680 DATA 173,8,32,9,0,141,8,32,96

```

\* Zeile 610 nimmt einige Sekunden in Anspruch!

## SUPER.HGR.ASM

```

1      * Maschinensprachteil aus SUPER HGR
2      *
3      ORG      $300
4      *
5      AYINT   EQU   $E10C      ;FAC->INT
6      BYTE   EQU   $2008      ;Byteadresse
7      *
8      JSR     AYINT
9      LDA     $A1
10     INTERF  BIT     $C091      ;Interface
11     BMI     INTERF
12     STA     $C091      ;in Slot 1
13     STA     $C092      ;Typabhängig!
14     STA     $C094
15     RTS
16     *
17     LDA     BYTE
18     ORA     #$80
19     STA     BYTE
20     RTS
29 Bytes

```

## Umwandlung in Großbuchstaben

von H.Grumer

Die meisten Apple-Besitzer verfügen über Groß/Kleinschreibung als serienmäßige Ausstattung des IIe und IIc oder als Nachrüstung des alten II Plus mit Kleinschreibumrüstungs- oder 80-Zeichenkarte. Von dieser Möglichkeit, die die Lesbarkeit von Programmen und Menütexen erheblich verbessert, wird auch in allen im Peeker veröffentlichten Applesoft-Programmen Gebrauch gemacht.

Um den Besitzern eines „ungetunten“ Apple II Plus gerecht zu werden, entstand das Programm **VERSAL**, das alle Kleinbuchstaben eines im Speicher befindlichen **Applesoft**-Programms in Großbuchstaben umwandelt. Darüber hinaus wird das Eszett (ß) in ein einfaches S konvertiert und alle Umlaute (ä, ö, ü, Ä, Ö, Ü) in

die entsprechenden einfachen Selbstlaute (a, o, u, A, O, U). Diese einfache Umsetzung gewährleistet die weitere Funktionsfähigkeit des Programms.

Für die Umwandlung ist folgendermaßen vorzugehen:

```

RUN VERSAL
LOAD PPROGRAMM.ALT
CALL 768
SAVE PPROGRAMM.NEU

```

Das Maschinenprogramm ist relokativ und bearbeitet auch verschobene Programme (z.B. hinter die HGR geladene Programme).

Der Quellcode zu dieser Maschinenroutine befindet sich auf der Peeker-Sammler-Disk unter dem Namen T.VERSAL.

## VERSAL

```

1 REM PROGRAMM ZUR UMWAND-
  LUNG VON KLEIN- IN
  GROSSBUCHSTABEN
2 FOR I = 768 TO 838: READ A:
  POKE I,A: NEXT
3 DATA 165,103,133,250,165,104,
  133,251,160,1,177,250,240,56
4 DATA 200,200,200,177,250,48,
  251,208,12,56,152,101,250,133
5 DATA 250,144,233,230,251,176,
  229,201,126,208,2,169,83,201
6 DATA 97,144,2,41,223,201,91,
  208,2,169,65,201,92,208,2,169
7 DATA 79,201,93,208,2,169,85,
  145,250,200,208,203,96

```

# Flag Monitor

## Eine Taktzähler-Utility

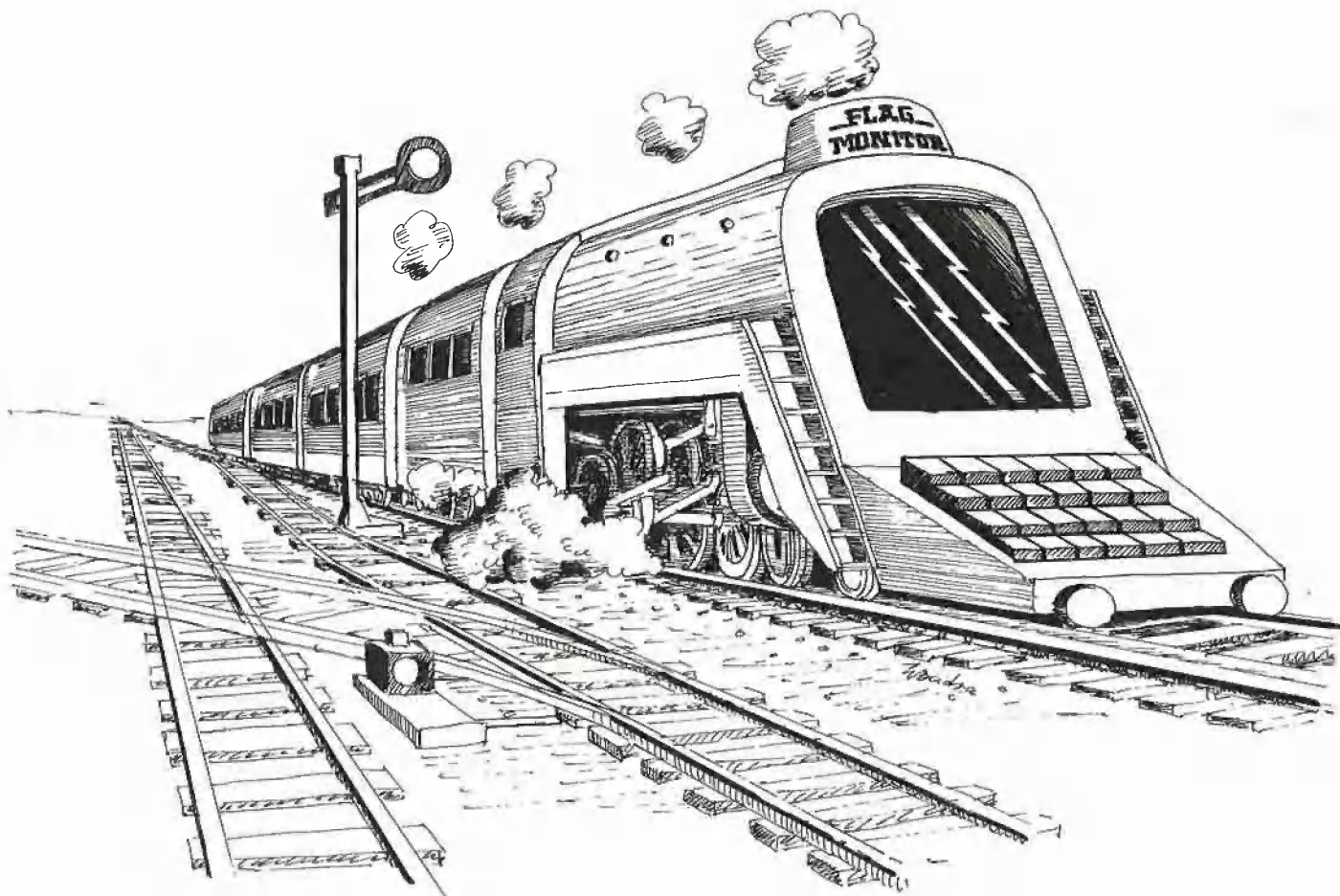
von Michael G. Schneider

Manchmal steht man vor der Aufgabe, die Laufzeit eines fertigen Programms zu reduzieren. Ein nicht ganz einfaches Problem, denn da der statische Quelltext nicht immer genügend Information über das dynamische Verhalten des Codes liefert, kommt man durch bloßes Lesen des Listings nicht weiter. Vielmehr benötigt man exakte Angaben darüber, wo das Programm den größten Teil der Zeit verbringt, um dann an diesen Stellen die Optimierungen durchführen zu können.

Bei der Ausarbeitung meines Beitrags zu dem Primzahlen-Wettbewerb (Pecker, 1/84, Seite 58) entwickelte ich zur Lösung dieses Problems eine Utility, welche ich im folgenden vorstellen möchte.

### Das Verfahren

Der Ablauf eines Maschinenprogramms hat prinzipiell sequentiellen Charakter, so daß man diesem weitestgehend auch mit „Papier und Bleistift“ folgen kann. Einzige Problempunkte sind bedingte Verzweigungen (Branch-Befehle wie etwa BCC oder BMI), da man im voraus nicht immer ganz einfach feststellen kann, in welchem Zustand sich bestimmte Flags befinden.





Man weiß daher auch nicht, wie oft eine Schleife tatsächlich durchlaufen wird und welchen Anteil sie an der Gesamtzeit bildet.

Es sei folgende Sequenz gegeben:

```
START Anweisung 1
.
.
.
Anweisung n
ENDE BCC START
```

Gehen wir einmal davon aus, daß die Schleife nur über das Label START betreten und auch nur über ENDE verlassen werden kann und die Anweisungen 1–n (also ohne das BCC) eine Laufzeit von 111 Zyklen haben. Wenn man nun z.B. wüßte, daß das Label ENDE insgesamt 123mal mit C = 0 und 45mal mit C = 1 erreicht wird, so könnte man die Verweildauer in der Schleife zu  $123 * (111 + 3) + 45 * (111 + 2) = 19107$  Zyklen bestimmen. Hierbei wurde berücksichtigt, daß ein Branch-Befehl 3 Zyklen bei Verzweigung und ansonsten 2 Zyklen benötigt.

Mit meiner Utility kann man nun überprüfen, wie oft eine bestimmte Anweisung passiert wurde und wie die Flag-Verhältnisse dabei aussahen. Dazu muß vor die betreffende Anweisung ein JSR-Sprung zur Utility gelegt werden.

**FLAG.MONITOR** besitzt die drei von außen zugänglichen Routinen INITIAL, MONITOR und DRUCKEN, wobei sich Sprungbefehle zum Anfang dieser Routinen bei \$8000, \$8003 und \$8006 befinden.

**INITIAL** führt eine Initialisierung durch und muß vom Anwender vor MONITOR einmal aufgerufen werden, da ansonsten wichtige Speicherplätze zerstört werden können!

**MONITOR** ist der Kern der Utility und wird von den relevanten Stellen des untersuchten Programms aktiviert. Im allgemeinen sind dies diejenigen Adressen, welche unmittelbar vor Branch-Befehlen liegen. Denn z.B. von einem BCC-Befehl will man ja gerade wissen, wie oft er mit C = 0 bzw. C = 1 erreicht wird. MONITOR legt für jede dieser aufrufenden Stellen einen 19 Bytes langen Block an, in dem alle interessanten Daten gespeichert werden.

Aus der **Tabelle 1** kann man entnehmen, daß in den ersten beiden Bytes des Blocks die Adresse der aufrufenden JSR-Anweisung abgelegt wird. Im Byte 3 befindet sich der Opcode der dem JSR \$8003 fol-

0,1	: JSR-Adresse
2	: Überlauf
3	: Opcode
4,5,6	: Gesamtzähler
7,8,9	: Zähler für PL
10,11,12	: Zähler für VC
13,14,15	: Zähler für CC
16,17,18	: Zähler für NE

genden Anweisung (also z.B. ein BCC-Code). In den Bytes 4 bis 18 werden 5 verschiedene 3-Byte-Zähler geführt. Die Bytes 4 bis 6 registrieren, wie oft MONITOR von der betreffenden Adresse aus aufgerufen wurde. Die übrigen Zähler geben an, wie oft davon der jeweilige Status vorgelegen hat. Um z.B. zu ermitteln, wie oft das Carry-Flag gesetzt war, muß der Zähler in 13 bis 15 von dem in 4 bis 6 subtrahiert werden. Sollte einer dieser Zähler einmal überlaufen (größer als 16 Millionen!), so wird das Byte 2 auf 1 gesetzt; ansonsten bleibt es 0. Die Routine **DRUCKEN** gibt ein formatiertes Listing der durch MONITOR aufgebauten Blöcke aus.

INITIAL und MONITOR benötigen keine externen Routinen, so daß diese jederzeit (also beispielsweise auch trotz zerstörter Zero-Page) aufgerufen werden können. DRUCKEN hingegen benutzt zur Ausgabe von Zeichen die üblichen Routinen aus dem ROM. Dies hat den Vorteil, daß man die Listings auch in einen File oder zum Drucker schicken kann.

### Ein Beispiel

Vor einer Beschreibung der Implementation soll hier ein Anwendungsbeispiel gegeben werden:

Angenommen man benötigt die folgende Funktion: Abhängig von dem Parameter W (zwischen 2 und 127) soll die kleinste Zahl N als Ergebnis geliefert werden, so daß  $256 + N$  ganzzahlig durch W teilbar ist. Einige (W, N)-Paare sind etwa (2, 0), (3, 2), (4, 0), (5, 4), denn z.B.  $256 + 4 = 260$  ist ganzzahlig durch 5 teilbar.

In **FLAG.MONITOR.TEST** sind drei Realisierungen der obigen Funktion zusam-

mengefaßt. Die erste addiert so lange den übergebenen Wert, bis dieser 256 erreicht (oder überschreitet). Die zweite multipliziert den Wert mit 2, bis dieser 128 erreicht, um dann ebenfalls fortgesetzt zu addieren. Die dritte Routine multipliziert zwar auch mit 2, hört jedoch erst dann auf, wenn 256 erreicht wurde. Dann wird der Wert subtrahiert, bis 256 gerade eben unterschritten wird. Um das Endergebnis zu erhalten, muß dann noch einmal der Wert addiert werden.

JSR-Adr : \$0308			
Su =	1074	MI =	564
PL =	510	VS =	126
VC =	948	CS =	126 <<<
CC =	948	EQ =	6
NE =	1068		
JSR-Adr : \$0311			
Su =	240	MI =	126 <<<
PL =	114	VS =	0
VC =	240	CS =	0
CC =	240	EQ =	0
NE =	240		
JSR-Adr : \$0318			
Su =	432	MI =	306
PL =	126	VS =	0
VC =	432	CS =	126 <<<
CC =	306	EQ =	6
NE =	426		
JSR-Adr : \$0321			
Su =	366	MI =	189
PL =	177	VS =	0
VC =	366	CS =	126 <<<
CC =	240	EQ =	6
NE =	360		
JSR-Adr : \$0328			
Su =	462	MI =	168
PL =	294	VS =	63
VC =	399	CS =	336 <<<
CC =	126	EQ =	0
NE =	462		
JSR-Adr : \$0342			
Su =	126	MI =	1 <<<
PL =	125	VS =	0
VC =	126	CS =	126
CC =	0	EQ =	0
NE =	126		

Es ist offensichtlich, daß die zweite Implementation der ersten überlegen ist. In welcher Relation jedoch IMPL2 und IMPL3 stehen, kann nicht ohne weiteres angegeben werden. Es ist nämlich so, daß für bestimmte Werte die Routine IMPL2 besser abschneidet, diese jedoch bei anderen Werten unterlegen ist.

Um eine Entscheidung treffen zu können, habe ich dann an den wichtigsten Punkten meine Utility auferufen (am Rand durch „---“ gekennzeichnet). Im Hauptprogramm werden dann nach der Initialisierung die drei Routinen mit allen möglichen Parametern auferufen. Die durch DRUKKEN erzeugte Zusammenstellung findet sich in der **Tabelle 2**.

Dieser kann man z.B. entnehmen (erster Block), daß MONITOR von \$0308 aus insgesamt 1074mal auferufen wurde (Su als Abkürzung für Summe). Davon galt 948mal CC (C = 0) und 126mal CS (C = 1). Man beachte, daß die Tabelle nach den JSR-Adressen sortiert wurde und die relevanten Zeilen in den Blöcken durch „<<<“ gekennzeichnet wurden. Beispielsweise folgt dem JSR MONITOR auf \$0308 ein BCC, so daß die CC/CS-Zeile hervorgehoben wurde.

Man kann nun die Laufzeit der verschiedenen Routinen einfach bestimmen. Zur Erleichterung habe ich im Quelltext am rechten Rand die jeweilige Zyklenzahl vermerkt. Es ergeben sich dann die in der **Tabelle 3** zusammengefaßten Zahlen. Erwartungsgemäß wird IMPL1 von den beiden anderen Routinen geschlagen. Nicht unmittelbar vorherzusehen war jedoch, daß IMPL2, für welches sich eine mittlere Laufzeit von  $3918 / 126 = 31.1$  Zyklen ergibt, deutlich schneller als IMPL3 ist.

\$0303-\$0305:	126 * (2 + 3)	
\$0306-\$030C:	948 * (3 + 3)	
	126 * (3 + 2)	6948
\$030E :	126 * 3	
\$0310-\$0315:	114 * (2 + 3)	
	126 * (2 + 2)	
\$0316-\$031C:	306 * (3 + 3)	
	126 * (3 + 2)	3918
\$031E :	126 * 3	
\$0320-\$0325:	240 * (2 + 3)	
	126 * (2 + 2)	
\$0326-\$032C:	126 * (3 + 2)	
	336 * (3 + 3)	
\$032D :	126 * 3	5106

## Die Implementation

Die FLAG.MONITOR-Utility belegt den Speicherplatz von \$8000 bis \$8336. Beginnend bei \$8337 werden die 19 Bytes langen Blöcke aufgebaut, deren Anzahl (und damit auch die höchste von FLAG-

.MONITOR benutzte Adresse) vom Anwender bestimmt wird. Die Maximalzahl der verwalteten Blöcke habe ich auf 253 beschränkt, so daß in keinem Fall das DOS, welches üblicherweise bei \$9600 startet, zerstört wird. Die Routinen benötigen ferner noch 6 Speicherplätze in der Zero-Page, die jedoch immer gesichert werden.

ANZAHL gibt die Anzahl der schon angelegten Blöcke an, während sich in ENDPTR die Adresse des ersten freien Speicherplatzes hinter den Blöcken befindet. Alle übrigen Variablen dienen nur zur kurzfristigen Speicherung von Werten und werden im weiteren erklärt.

ZEHNPOT enthält eine Tabelle von Zehnerpotenzen, die bei der Hex-Dez-Umwandlung benutzt werden. FLAGNAME und TEXT definieren einige Texte, auf die durch DRUCKEN zugegriffen wird.

SICHZUST bzw. RESTZUST sichert bzw. restauriert den Zustand, in welchem die Utility auferufen wurde. Schließlich dürfen durch FLAG.MONITOR keine Registerinhalte oder Flags verändert werden. Diese beiden Routinen werden von INITIAL, MONITOR und DRUCKEN auferufen.

INITIAL versetzt den Datenbereich in einen wohldefinierten Zustand, indem ANZAHL auf 0 und BLOCKPTR auf die erste freie Adresse hinter dem Code gesetzt wird.

MONITOR ist das Hauptprogramm und besteht im wesentlichen aus Aufrufen von Unterprogrammen. Zunächst wird die Adresse der auferufenden Stelle ermittelt und in JSRADR gespeichert. Man beachte, daß sich auf dem Stack die um 1 verminderte Rückkehradresse befindet. In SUCHEN wird getestet, ob für diese Adresse schon ein Block angelegt wurde. In diesem Fall ist C = 0 und BLOCKPTR zeigt auf eben diesen Block. Andernfalls weist BLOCKPTR auf den ersten Block, dessen JSR-Adresse größer als die vorgegebene ist, und ZAEHLER gibt die Anzahl der Blöcke jenseits von BLOCKPTR an. ANLEGEN nutzt diese Information dann aus, indem es erst die folgenden Blöcke nach oben schiebt und in dem so entstandenen freien Raum einen neuen Block (mit leeren Zählern) anlegt. Durch dieses Verschieben wird erreicht, daß sich die Blöcke immer in aufsteigender Reihenfolge befinden. Die Routine EINTRAGEN inkrementiert dann abhängig vom Prozessor-Status die Zähler.

Die Routine DRUCKEN gibt die Information aus den zuvor angelegten Blöcken

formatiert aus und benutzt unter anderem die Routinen IDENCODE und HEXDEZ.

Zum besseren Verständnis von IDENCODE habe ich in der **Tabelle 4** die Op-codes der Branch-Anweisungen zusammengefaßt. Aus ihr ersieht man (in binärer Darstellung), daß ein Branch-Code immer die Form hh01 0000 oder hh11 0000 hat. Durch schrittweises Schieben nach rechts wird, falls ein Branch-Code vorliegt, hh + 1 in IDENT abgelegt.

HEXDEZ führt die Umwandlung einer Hex-Zahl in einen String aus dezimalen Ziffern durch.

BPL	-	\$10	-	0001	0000
BMI	-	\$30	-	0011	0000
BVC	-	\$50	-	0101	0000
BVS	-	\$70	-	0111	0000
BCC	-	\$90	-	1001	0000
BCS	-	\$B0	-	1011	0000
BNE	-	\$D0	-	1101	0000
BEQ	-	\$F0	-	1111	0000

## Schlußbemerkungen

Wenn zur Analyse eines Programms viele MONITOR-Aufrufe eingestreut werden, so vergrößern diese natürlich die Ausführungszeit. Obwohl das exakte Ausmaß der Verzögerung von der konkreten Situation (Anzahl der Blöcke, Zustand der Flags) abhängt, läßt es sich mit etwa  $500 + 40 * S$  Zyklen abschätzen. Darin sei S die mittlere Anzahl der Blöcke, die durchsucht werden müssen, bevor der richtige gefunden wird.

Falls z.B. im Verlauf des Programms 20 verschiedene Blöcke aufgebaut wurden, müssen im Durchschnitt 10 Blöcke durchsucht werden. Der Aufwand für jeden Aufruf von MONITOR beträgt dann also ungefähr  $500 + 40 * 10 = 900$  Zyklen. Für jeweils 1111 Aufrufe muß man daher mit einer Verzögerung von 1 Sekunde rechnen.

Bei Abschätzung der obigen Formel bin ich davon ausgegangen, daß die Anzahl der Blöcke sehr viel geringer ist als die Anzahl der MONITOR-Aufrufe. Aus diesem Grund kann der durch die Routine ANLEGEN verursachte Aufwand vernachlässigt werden. Ferner habe ich angenommen, daß beim Inkrementieren 2 der 4 Flag-Zähler behandelt werden müssen. Wenn natürlich immer nur PL, VC, CC und NE vorliegen, muß die Zyklenzahl noch etwas höher angesetzt werden.





**AKUSTIK-KOPPLER - Dataphon s21d**  
 300 Baud Modem, nach CCITT V.21 Standard,  
 m. FTZ-Nr. 18.13.1917.00, Gebühren- und  
 anmeldungsfrei, V24/RS-232 Standard-Schnittst. nur DM 298,00

**TELEKOMMUNIKATIONS - KOMPLETT - PAKET**  
 geeignet für Apple //+ und Apple //e:  
 1 Dataphon s21d,  
 1 Anschlußkabel: V.24 zum Apple II-Game-I/O,  
 1 Terminalprogramm: "HIB Modem-Transfer" nur DM 398,00

**Chinon-Laufwerk (Testbericht in Peeker 5/85)**  
 für Apple //+ und Apple //e anschl. im Gehäuse DM 498,00  
 w.o. jedoch für Apple //c DM 569,00

**TOSHIBA Spitzenlaufwerke zum Superpreis!**  
 ND 06-D, 2 x 80 Track, 640 K-Byte formatiert DM 549,00

**DISK-DOPPEL-STATION (APPLE //+, APPLE //e)**  
 2 x ND 06-D im Geh. + Auto-Patchcontr., 1,2 MB DM 1698,00

**AUTO-PATCH-CONTROLLER** DM 298,00

**IC-Test-Karte (Testet ca. 500 verschiedene IC's)** DM 398,00

**BROTHER-Matrixdrucker, die Super-Drucker!**  
 M-1009 (Matrixdrucker, RS-232 + Centronics) DM 698,00  
 M-1009 anschl. fertig an:  
 Apple //c (mit Drucker-Kabel) DM 798,00  
 Apple //e (mit Graphik-Interface und Kabel) DM 898,00

Alle Preise inklusive der gesetzlichen Mehrwertsteuer.  
 Berechnung der Versandkosten erfolgt nach Entfernung und Gewicht.  
 Fordern Sie noch heute unsere Gratispreisliste an! Wiederverkäufer bitte nur  
 schriftlich anfragen (Kopie der Gewerbeanmeldung beilegen!).

**hib** HIB Computerladen  
 Auß. Bayreuther Str. 72 - Telefon: 0911 / 515 939  
 Postfach 21 01 25 - Telex: 17 - 911 8253  
 8500 Nürnberg 21 - Teletex: 2526 - 911 82 53

## Die Ergebnisse vieler Stunden vor dem Apple hier zu Papier gebracht.

Es geht hier weniger um das elementare Programmieren des Rechners, sondern um Assemblerprogramme, die extensiv Monitor-ROM-Subroutinen benutzen. Diese hat der Autor nach Sachgebieten geordnet, z. B. Mathematik, Graphik, String-Bearbeitung + Disassembler-Listings und diese wiederum mit Erklärungen und Applikationen komplettiert. Eine ausreichende Dokumentation ist dabei immer gewährleistet. Sie geht schrittweise vor, von der Aufgabenstellung über die Programmentwicklung bis zum lauffähigen Maschinenprogramm, die angebotenen Beispiele sind ausbaufähig.

### Das Buch zum Apple II

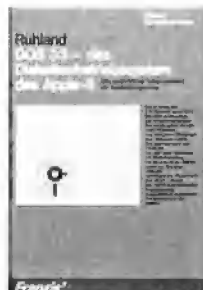
Theorie und Praxis des Apple-II-Systems. Von E. Esders, 210 S., 119 Abb., geb. DM 54.- ISBN 3-7723-7641-X



Das Buch erklärt die Verfahren, mit denen die Informationen auf die Disketten geschrieben werden, mit denen die Steuerung der Laufwerke erfolgt, mit denen Betriebssystem und Programmiersystem aneinandergelinkt werden, mit denen die Routinen des Betriebssystems realisiert werden. Einen großen Teil des Buches nimmt die Einzelschrittdokumentation ein, außerdem sind schwierige Programmstellen durch Flußdiagramme dargestellt. Nun kann der Apple-II-Fan Disketteninformationen auch mal lesen, sogar manipulieren und eventuell das Betriebssystem ändern.

### DOS 3.3 - das Disketten-Betriebssystem des Apple-II

Eine ausführliche Dokumentation der Systemprogramme. Von B. Ruhland, 253 S., 12 Abb., geb. DM 48.- ISBN 3-7723-7691-6



**Franzis**

der große Fachverlag für angewandte Elektronik und Informatik  
 Franzis-Verlag, München

## ZUSATZ-KARTEN:

V-24-Schnittstelle ..... 199,- Z-80-Karte ..... 139,-  
 80-Zeichen-Karte m. Softswitch 236,- 16 K-Sprache-Karte ..... 138,-  
 Centronics-Karte von Epson für Graphik ..... 210,- für Text ..... 145,-  
 Centronics-Schnittstelle für 2 Drucker gleichzeitig ..... 129,-  
 Eprommer incl. Software ..... 198,-

**Super-Eprommer** ..... **239,-**  
 belegt keinen Slot, incl. Software für 2508-27128

## Floppy-Controller

FDC 4 für alle Laufwerke ..... 170,- Bausatz wie links ..... 159,-  
 Leerplatine wie oben incl. Prom u. Eprom ..... 98,-  
 Druck Spooler mit 16, 32 oder 64 KB ..... Preis auf Anfrage

**Erphi-Controller** ..... **298,-**

**Drucker-Spooler 64 kB,** ..... **340,-**  
 fertig aufgebaut incl. Netzteil u. Gehäuse

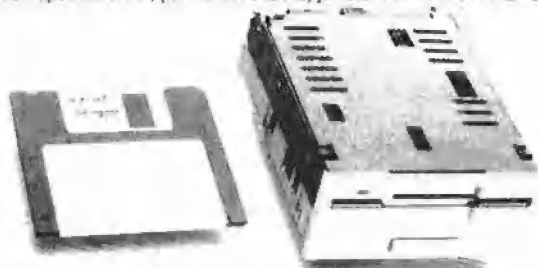
Joy Stick De Luxe ..... 59,- Netzteil 5A ..... 149,-  
 Gehäuse für 1 5/8" Slimline Laufwerk ..... 39,-  
 Gehäuse für 2 5/8" Slimline Laufwerke mit Platz für ein Netzteil ..... 159,-  
 Gehäuse für 2 3/4" Slimline Laufwerke mit Platz für ein Netzteil ..... 79,-  
 IBM"-Gehäuse ..... 229,-  
 Floppy-Kabel 34pol. für 2 Laufwerke mit Shugart-Bus ..... 35,-

## 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,-**



**TEAC 3 1/2" Laufwerk FD 35 F** ..... **535,-**  
 Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur



TEAC FD 55 A 1 x 40 Track ..... 445,- TEAC FD 55 B 2 x 40 Track ..... 515,-  
 TEAC FD 55 E 1 x 80 Track ..... 490,- TEAC FD 55 F 2 x 80 Track ..... 560,-

**SONY 3 1/2" Laufwerk** ..... nur **799,-**  
 Apple"-kompatibles Laufwerk incl. Gehäuse + Kabel ..... **599,-**

**320 KB Laufwerk für IIc** ..... **948,-**  
**640 KB Laufwerk für IIc** ..... auf Anfrage

## EPSON DRUCKER

EPSON FX 80 ..... 1670,- EPSON FX 100 ..... 2159,-  
 EPSON RX 80 ..... 1079,- EPSON RX 80 FT ..... 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 ..... **1640,-**  
 Low Power Version ..... **1740,-**



**10 MB Winchester** ..... **4490,-**  
 mit Software für DOS 3.3, CP/M 2.20, Pascal, Pro-DOS, incl. Controller und Gehäuse

**Sonderangebot**  
**Distar Laufwerk** für II+IIc, ..... jetzt nur **448,-**  
 incl. Kabel u. Gehäuse

Gesamt-Preisliste anfordern! Preise inklusive gesetzlicher Mehrwertsteuer.

**UEDING electronics**

Holtwiese 2  
 5750 Menden 1

DFÜ 02373/66877  
 Tel. 02373/63159

FLAG. MONITOR

```

1 *****
2 *
3 *          FLAG_MONITOR          *
4 *
5 *          Michael G. Schneider   *
6 *
7 *****
8          ORG $8000
9
8000: 4C AC 80      JMP INITIAL
8003: 4C C2 80      JMP MONITOR
8006: 4C C8 81      JMP DRUCKEN
-----
13
14
15 BLOCKLEN EQU 19
16
17 ZEROPAGE EQU $00
18 BLOCKPTR EQU ZEROPAGE
19 QUELLPTR EQU ZEROPAGE+2
20 ZIELPTR EQU ZEROPAGE+4
21 OPCPTR EQU ZEROPAGE+4
22
23 COUT EQU $FDED
24 CROUT EQU $FD8E
25 PRHEX EQU $FDDA
26 PRBL2 EQU $F94A
-----
27
28
29 ANZAHL DS 1
30 ENDPTR DS 2
31
32 SICHERA DS 1
33 SICHERX DS 1
34 SICHERY DS 1
35 SICHERP DS 1
36 SICHERZP DS 6
37
38 FLAG DS 1
39 STATUS DS 1
40 IDENT DS 1
41 ZAEHLER DS 1
42 JSRADR DS 2
43
44 HEXWERT DS 3
45 DEZSTRNG DS 8
-----
46
47
8027: 80 96 98      ZEHNROT DFB $80,$96,$98 ;10000000
802A: 40 42 0F 49      DFB $40,$42,$0F ;10000000
802D: A0 86 01 50      DFB $A0,$86,$01 ;10000000
8030: 10 27 00 51      DFB $10,$27,$00 ;10000000
8033: E8 03 00 52      DFB $E8,$03,$00 ;10000000
8036: 64 00 00 53      DFB $64,$00,$00 ;10000000
8039: 0A 00 00 54      DFB $0A,$00,$00 ;10000000
803C: 01 00 00 55      DFB $01,$00,$00 ;10000000
-----
56
57
803F: D0 CC CD 58      FLAGNAME ASC "PLMIVCVSCCCSNEEQ"
8042: C9 D6 C3 D6 D3 C3 C3 C3
804A: D3 CE C5 C5 D1
-----
59
60
61 TEXT
804F: CA D3 D2 62      TXTJSR ASC "JSR-Adr : $"
8052: AD C1 E4 F2 A0 BA A0 A4
805A: 00 63          HEX 00
805B: A0 A0 A0 64      TXTUBRLF ASC " !!! Überlauf !!!"
805E: A1 A1 A1 A0 DD E2 E5 F2
8066: EC E1 F5 E6 A0 A1 A1 A1
806E: 00 65          HEX 00
806F: D3 F5 66      TXTSU ASC "Su"
8071: A0 BD A0 67      TXTGLHZ ASC " = "
8074: 00 68          HEX 00
8075: A0 BC BC 69      TXTB RNCH ASC " <<<"
8078: BC
8079: 00 70          HEX 00
-----
71
72
73 * Sichere den momentanen Zustand (alle
74 * Register, Prozessor-Status sowie den
75 * Speicher von $00 bis $05).
76
807A: 8D 0C 80 77      SICHZUST STA SICHERA
807D: 8E 0D 80 78      STX SICHERX
8080: 8C 0E 80 79      STY SICHERY

```

```

8083: 08 80          PHP
8084: 68 81          PLA
8085: 8D 0F 80 82      STA SICHERP
8088: A2 05 83          LDX #5
808A: B5 00 84          LDA ZEROPAGE,X
808C: 9D 10 80 85      STA SICHERZP,X
808F: CA 86          DEX
8090: 10 F8 87          BPL L1
8092: 60 88          RTS
-----
89
90
91 * Restauriere den alten Zustand.
92
8093: A2 05 93      RESTZUST LDX #5
8095: BD 10 80 94      L2 LDA SICHERZP,X
8098: 95 00 95          STA ZEROPAGE,X
809A: CA 96          DEX
809B: 10 F8 97          BPL L2
809D: AD 0F 80 98      LDA SICHERP
80A0: 48 99          PHA
80A1: AC 0E 80 100      LDY SICHERY
80A4: AE 0D 80 101      LDX SICHERX
80A7: AD 0C 80 102      LDA SICHERA
80AA: 28 103          PLP
80AB: 60 104          RTS
-----
105
106
107 * Initialisiere die Anzahl der Blöcke
108 * und den Pointer auf deren Ende.
109
80AC: 20 7A 80 110     INITIAL JSR SICHZUST
80AF: A9 00 111          LDA #0
80B1: 8D 09 80 112      STA ANZAHL
80B4: A9 37 113          LDA #<BLOECKE
80B6: 8D 0A 80 114      STA ENDPTR
80B9: A9 83 115          LDA #>BLOECKE
80BB: 8D 0B 80 116      STA ENDPTR+1
80BE: 20 93 80 117     JSR RESTZUST
80C1: 60 118          RTS
-----
119
120
121 * Hauptprogramm: Trage den Flag-Zustand
122 * in einem der Blöcke ein. Lege diesen
123 * notfalls noch an.
124
80C2: 20 7A 80 125     MONITOR JSR SICHZUST
-----
126
127 * Hole die Adresse der rufenden Stelle
128 * vom Stack und lege sie in JSRADR ab.
129
80C5: 68 130          PLA
80C6: AA 131          TAX
80C7: 68 132          PLA
80C8: A8 133          TAY
80C9: 48 134          PHA
80CA: 8A 135          TXA
80CB: 48 136          PHA
80CC: 38 137          SEC
80CD: E9 02 138          SBC #2
80CF: B0 01 139          BCS L3
80D1: 88 140          DEY
80D2: 8D 1A 80 141     L3 STA JSRADR
80D5: 8C 1B 80 142     STY JSRADR+1
80D8: 20 EE 80 143     JSR SUCHEN
80DB: 90 0A 144          BCC L4
80DD: AD 09 80 145     LDA ANZAHL
80E0: C9 FD 146          CMP #253
80E2: F0 06 147          BEQ L5
80E4: 20 26 81 148     JSR ANLEGEN
80E7: 20 85 81 149     L4 JSR EINTRAG
80EA: 20 93 80 150     L5 JSR RESTZUST
80ED: 60 151          RTS
-----
152
153
154 * Suche die schon vorhandenen Blöcke
155 * nach der aktuellen Adresse ab.
156
80EE: A9 37 157     SUCHEN LDA #<BLOECKE
80F0: 85 00 158     STA BLOCKPTR
80F2: A9 83 159     LDA #>BLOECKE
80F4: 85 01 160     STA BLOCKPTR+1
80F6: AD 09 80 161     LDA ANZAHL
80F9: 8D 19 80 162     STA ZAEHLER
80FC: D0 11 163     BNE L9
80FE: 38 164     L6 SEC
80FF: 60 165     RTS

```

```

166
167 * Berechne Pointer auf nächsten Block.
168
8100: A5 00 169 L7 LDA BLOCKPTR
8102: 69 13 170 ADC #BLOCKLEN
8104: 90 02 171 BCC L8
8106: E6 01 172 INC BLOCKPTR+1
8108: 85 00 173 L8 STA BLOCKPTR
810A: CE 19 80 174 DEC ZAEHLER
810D: F0 EF 175 BEQ L6
176
177 * Vergleiche die beiden Adressen.
178
810F: A0 01 179 L9 LDY #1
8111: B1 00 180 LDA (BLOCKPTR),Y
8113: CD 1B 80 181 CMP JSRADR+1
8116: 90 E8 182 BCC L7
8118: D0 0B 183 BNE L10
811A: 88 184 DEY
811B: B1 00 185 LDA (BLOCKPTR),Y
811D: CD 1A 80 186 CMP JSRADR
8120: 90 DE 187 BCC L7
8122: D0 01 188 BNE L10
8124: 18 189 CLC
8125: 60 190 L10 RTS
191
192
193 * Lege einen neuen Block bei (BLOCKPTR)
194 * an. Schiebe zuvor ZAEHLER Blöcke nach
195 * oben.
196
8126: AE 19 80 197 ANLEGEN LDX ZAEHLER
8129: F0 24 198 BEQ L14
199
812B: AD 0A 80 200 LDA ENDPTR
812E: AC 0B 80 201 LDY ENDPTR+1
8131: 85 04 202 L11 STA ZIELPTR
8133: 84 05 203 STY ZIELPTR+1
8135: 38 204 SEC
8136: E9 13 205 SBC #BLOCKLEN
8138: B0 01 206 BCS L12
813A: 88 207 DEY
813B: 85 02 208 L12 STA QUELLPTR
813D: 84 03 209 STY QUELLPTR+1
210
211 * Kopiere von (QUELLE) nach (ZIEL).
212
813F: A0 12 213 LDY #BLOCKLEN-1
8141: B1 02 214 L13 LDA (QUELLPTR),Y
8143: 91 04 215 STA (ZIELPTR),Y
8145: 88 216 DEY
8146: 10 F9 217 BPL L13
8148: A5 02 218 LDA QUELLPTR
814A: A4 03 219 LDY QUELLPTR+1
814C: CA 220 DEX
814D: D0 E2 221 BNE L11
222
223 * Erhöhe ANZAHL und ENDPTR um einen Block
224
814F: EE 09 80 225 L14 INC ANZAHL
8152: 18 226 CLC
8153: AD 0A 80 227 LDA ENDPTR
8156: 69 13 228 ADC #BLOCKLEN
8158: 8D 0A 80 229 STA ENDPTR
815B: 90 03 230 BCC L15
815D: EE 0B 80 231 INC ENDPTR+1
232
233 * Lege die JSR-Adresse im Block ab, ...
234
8160: A0 00 235 L15 LDY #0
8162: AD 1A 80 236 LDA JSRADR
8165: 85 04 237 STA OPCPTR
8167: 91 00 238 STA (BLOCKPTR),Y
8169: C8 239 INY
816A: AD 1B 80 240 LDA JSRADR+1
816D: 85 05 241 STA OPCPTR+1
816F: 91 00 242 STA (BLOCKPTR),Y
243
244 * ... setze das Überlauf-Byte auf 0,
245
8171: C8 246 INY
8172: A9 00 247 LDA #0
8174: 91 00 248 STA (BLOCKPTR),Y
249
250 * ... speichere den Opcode.
251

```

```

8176: C8 252 INY
8177: B1 04 253 LDA (OPCPTR),Y
8179: 91 00 254 STA (BLOCKPTR),Y
255
256 * ... und lösche alle Zähler.
257
817B: A9 00 258 LDA #0
817D: C8 259 L16 INY
817E: 91 00 260 STA (BLOCKPTR),Y
8180: C0 12 261 CPY #BLOCKLEN-1
8182: D0 F9 262 BNE L16
8184: 60 263 RTS
264
265
266 * Trage den Prozessor-Status in die
267 * Zähler ein.
268
8185: AD 0F 80 269 EINTRAG LDA SICHERP
8188: 48 270 PHA
8189: 48 271 PHA
818A: 48 272 PHA
818B: 48 273 PHA
274
275 * Inkrementiere die Gesamtsumme ...
276
818C: A0 04 277 LDY #4
818E: 20 B2 81 278 JSR INKREMENT
279
280 * ... und auch die PL-VC-CC-NE-Zähler.
281
8191: 28 282 PLP
8192: 30 05 283 BMI L17
8194: A0 07 284 LDY #7
8196: 20 B2 81 285 JSR INKREMENT
286
8199: 28 287 L17 PLP
819A: 70 05 288 BVS L18
819C: A0 0A 289 LDY #10
819E: 20 B2 81 290 JSR INKREMENT
291
81A1: 28 292 L18 PLP
81A2: B0 05 293 BCS L19
81A4: A0 0D 294 LDY #13
81A6: 20 B2 81 295 JSR INKREMENT
296
81A9: 28 297 L19 PLP
81AA: F0 05 298 BEQ L20
81AC: A0 10 299 LDY #16
81AE: 20 B2 81 300 JSR INKREMENT
301
81B1: 60 302 L20 RTS
303
304
305 * Inkrementiere den durch (BLOCKPTR),Y
306 * bestimmten 3-Byte-Zähler.
307
81B2: 38 308 INKREMENT SEC
81B3: A2 03 309 LDX #3
81B5: B1 00 310 L21 LDA (BLOCKPTR),Y
81B7: 69 00 311 ADC #0
81B9: 91 00 312 STA (BLOCKPTR),Y
81BB: 90 0A 313 BCC L22
81BD: C8 314 INY
81BE: CA 315 DEX
81BF: D0 F4 316 BNE L21
317
318 * Achtung: Überlauf!
319
81C1: A9 01 320 LDY #1
81C3: A0 02 321 LDY #2
81C5: 91 00 322 STA (BLOCKPTR),Y
81C7: 60 323 L22 RTS
324
325
326 * Drucke die zuvor angelegten Blöcke
327 * formatiert aus.
328
81C8: 20 7A 80 329 DRUCKEN JSR SICHZUST
81CB: AD 09 80 330 LDA ANZAHL
81CE: F0 78 331 BEQ L28
81D0: 8D 19 80 332 STA ZAEHLER
81D3: A9 37 333 LDA #<BLOECKE
81D5: 85 00 334 STA BLOCKPTR
81D7: A9 83 335 LDA #>BLOECKE
81D9: 85 01 336 STA BLOCKPTR+1
81DB: 20 8E FD 337 L23 JSR CROUT

```

```

338
339 * Identifiziere den Opcode.
340
81DE: 20 4C 82 341 JSR IDENCODE
342
343 * Gib die JSR-Adresse aus ...
344
81E1: A0 00 345 LDY #TXTJSR-TEXT
81E3: 20 9F 82 346 JSR PRNTTXX
81E6: A0 01 347 LDY #1
81E8: B1 00 348 LDA (BLOCKPTR),Y
81EA: 20 DA FD 349 JSR PRHEX
81ED: A0 00 350 LDY #0
81EF: B1 00 351 LDA (BLOCKPTR),Y
81F1: 20 DA FD 352 JSR PRHEX
353
354 * ... und eventuell die Überlauf-Info.
355
81F4: A0 02 356 LDY #2
81F6: B1 00 357 LDA (BLOCKPTR),Y
81F8: F0 05 358 BEQ L24
81FA: A0 0C 359 LDY #TXTUBRLF-TEXT
81FC: 20 9F 82 360 JSR PRNTTXX
81FF: 20 8E FD 361 L24 JSR CROUT
8202: 20 8E FD 362 JSR CROUT
8205: A0 20 363 LDY #XTSU-TEXT
8207: 20 9F 82 364 JSR PRNTTXX
365
366 * Kopiere die Summe nach HEXWERT, ...
367
820A: A2 02 368 LDX #2
820C: A0 06 369 LDY #6
820E: B1 00 370 L25 LDA (BLOCKPTR),Y
8210: 9D 1C 80 371 STA HEXWERT,X
8213: 88 372 DEY
8214: CA 373 DEX
8215: 10 F7 374 BPL L25
375
376 * ... wandle sie in das dezimale System
377 * um und drucke sie dann aus.
378
8217: 20 64 82 379 JSR HEXDEZ
821A: 20 AB 82 380 JSR PRNTDEZ
821D: 20 8E FD 381 JSR CROUT
382
383 * Gib die Summen der 4 Flags aus.
384
8220: A9 00 385 LDA #0
8222: 8D 16 80 386 STA FLAG
8225: 20 CA 82 387 L26 JSR PRNTFLG
8228: 20 8E FD 388 JSR CROUT
822B: EE 16 80 389 INC FLAG
822E: AD 16 80 390 LDA FLAG
8231: C9 04 391 CMP #4
8233: D0 F0 392 BNE L26
393
394 * Setze BLOCKPTR auf den nächsten Block
395
8235: 18 396 CLC
8236: A5 00 397 LDA BLOCKPTR
8238: 69 13 398 ADC #BLOCKLEN
823A: 85 0A 399 STA BLOCKPTR
823C: 90 02 400 BCC L27
823E: E6 01 401 INC BLOCKPTR+1
8240: 20 8E FD 402 L27 JSR CROUT
403
404 * Nächster Block.
405
8243: CE 19 80 406 DEC ZAEHLER
8246: D0 93 407 BNE L25
8248: 20 93 80 408 L28 JSR RESTZUST
824B: 60 409 RTS
410
411
412 * Identifiziere den Opcode:
413
414 * 0 ... kein Branch-Befehl
415 * 1 ... BPL oder BMI
416 * 2 ... BVC oder BVS
417 * 3 ... BCC oder BCS
418 * 4 ... BNE oder BEQ
419
824C: A0 03 420 IDENCODE LDY #3
824E: B1 00 421 LDA (BLOCKPTR),Y
8250: A2 00 422 LDX #0
8252: A0 04 423 LDY #4

```

```

8254: 4A 424 L29 LSR
8255: B0 09 425 BCS L30
8257: 88 426 DEY
8258: D0 FA 427 BNE L29
825A: 4A 428 LSR
825B: 90 03 429 BCC L30
825D: 4A 430 LSR
825E: AA 431 TAX
825F: E8 432 INX
8260: 8E 18 80 433 L30 STX IDENT
8263: 60 434 RTS
435
436 * -----
437 * Wandle HEXWERT in DEZSTRNG um.
438
8264: A2 00 439 HEXDEZ LDX #0
8266: A0 00 440 LDY #0
8268: A9 B0 441 L31 LDA #"0"
826A: 9D 1F 80 442 STA DEZSTRNG,X
826D: D0 0E 443 BNE L33 ;immer!
444
445 * Sie ließ sich subtrahieren.
446
826F: 8D 1E 80 447 L32 STA HEXWERT+2
8272: 68 448 PLA
8273: 8D 1D 80 449 STA HEXWERT+1
8276: 68 450 PLA
8277: 8D 1C 80 451 STA HEXWERT
827A: FE 1F 80 452 INC DEZSTRNG,X
453
454 * Versuche, ob sich die Zehnerpotenz vom
455 * HEXWERT subtrahieren läßt.
456
827D: 38 457 L33 SEC
827E: AD 1C 80 458 LDA HEXWERT
8281: F9 27 80 459 SBC ZEHNPOT,Y
8284: 48 460 PHA
8285: AD 1D 80 461 LDA HEXWERT+1
8288: F9 28 80 462 SBC ZEHNPOT+1,Y
828B: 48 463 PHA
828C: AD 1E 80 464 LDA HEXWERT+2
828F: F9 29 80 465 SBC ZEHNPOT+2,Y
8292: B0 DB 466 BCS L32
467
468 * Die Differenz wäre negativ. Also gehe
469 * zur nächsten Ziffer/Potenz über.
470
8294: 68 471 PLA
8295: 68 472 PLA
8296: C8 473 INY
8297: C8 474 INY
8298: C8 475 INY
8299: E8 476 INX
829A: E0 08 477 CPX #8
829C: D0 CA 478 BNE L31
829E: 60 479 RTS
480
481 * -----
482 * Drucke einen der Texte aus.
483
829F: B9 4F 80 484 PRNTTXX LDA TEXT,Y
82A2: F0 06 485 BEQ L34
82A4: 20 ED FD 486 JSR COUT
82A7: C8 487 INY
82A8: D0 F5 488 BNE PRNTTXX ;immer!
82AA: 60 489 L34 RTS
490
491 * -----
492 * Gib die Dezimalzahl aus, wandle aber
493 * führende Nullen in Leerzeichen um.
494
82AB: A0 00 495 PRNTDEZ LDY #0
82AD: B9 1F 80 496 L35 LDA DEZSTRNG,Y
82B0: C9 B0 497 CMP #"0"
82B2: D0 0A 498 BNE L36
82B4: A9 A0 499 LDA #" "
82B6: 20 ED FD 500 JSR COUT
82B9: C8 501 INY
82BA: C0 07 502 CPY #7
82BC: D0 EF 503 BNE L35
504
82BE: B9 1F 80 505 L36 LDA DEZSTRNG,Y
82C1: 20 ED FD 506 JSR COUT
82C4: C8 507 INY
82C5: C0 08 508 CPY #8
82C7: D0 F5 509 BNE L36

```

```

82C9: 60      510      RTS
511      *-----
512
513      * Drucke die zwei Summen des Flags (für
514      * Status = 0 bzw. 1) aus
515
82CA: A9 00    516      PRNTFLG LDA #0
82CC: 8D 17 80 517      STA STATUS
82CF: F0 08    518      BEQ L38      ;immer!
82D1: EE 17 80 519      L37      INC STATUS
82D4: A2 08    520      LDX #8
82D6: 20 4A F9 521      JSR PRBL2
522
523      * Berechne den Index des Namens ...
524
82D9: AD 16 80 525      L38      LDA FLAG
82DC: 0A      526      ASL
82DD: 6D 17 80 527      ADC STATUS
82E0: 0A      528      ASL
82E1: A8      529      TAY
82E2: B9 3F 80 530      LDA FLAGNAME,Y
82E5: 20 ED FD 531      JSR COUT
82E8: B9 40 80 532      LDA FLAGNAME+1,Y
82EB: 20 ED FD 533      JSR COUT
82EE: A0 22    534      LDY #TXTGLHZ-TEXT
82F0: 20 9F 82 535      JSR PRNTTXT
536
537      * ... und den Index des Zählers.
538
82F3: AD 16 80 539      LDA FLAG
82F6: 0A      540      ASL
82F7: 6D 16 80 541      ADC FLAG
82FA: 69 09    542      ADC #9
82FC: A8      543      TAY
544
545      * Kopiere die Zahl nach HEXWERT ...
546
82FD: A2 02    547      LDX #2
82FF: B1 00    548      L39      LDA (BLOCKPTR),Y
8301: 9D 1C 80 549      STA HEXWERT,X
8304: 88      550      DEY
8305: CA      551      DEX
8306: 10 F7    552      BPL L39
553
554      * ... und subtrahiere sie gegebenenfalls
555      * (Status = 1) von der Gesamtsumme.
556
8308: AD 17 80 557      LDA STATUS
830B: F0 14    558      BEQ L41
559
830D: 38      560      SEC
830E: A2 00    561      LDX #0
8310: A0 04    562      LDY #4
8312: B1 00    563      L40      LDA (BLOCKPTR),Y
8314: FD 1C 80 564      SBC HEXWERT,X
8317: 9D 1C 80 565      STA HEXWERT,X
831A: E8      566      INX
831B: C8      567      INY
831C: 8A      568      TXA
831D: 49 03    569      EOR #3
831F: D0 F1    570      BNE L40
571
572      * Gib sie in Dezimaldarstellung aus.
573
8321: 20 64 82 574      L41      JSR HEXDEZ
8324: 20 AB 82 575      JSR PRNTDEZ
8327: AD 17 80 576      LDA STATUS
832A: F0 A5    577      BEQ L37
578
579      * Liegt das interessante Flag schon vor?
580
832C: CE 18 80 581      DEC IDENT
832F: D0 05    582      BNE L42
8331: A0 26    583      LDY #TXTIBRNCH-TEXT
8333: 20 9F 82 584      JSR PRNTTXT
8336: 60      585      L42      RTS
586
587
588      BLOECKE

```

823 Bytes

**Hinweis:** Dieses Programm wurde mit dem S-C Macro Assembler erstellt. Bei der Konvertierung in das Big-Mac-Format wurden die lokalen Labels in die Labels L1 bis L42 umgewandelt.

## FLAG.MONITOR.TEST

```

1      *****
2      *
3      *          FLAG_MONITOR_TEST
4      *
5      *          Michael G. Schneider
6      *
7      *-----
8      *
9      *          ORG $300
10     *
11     *          JMP START
12     *-----
13     WERT      EQU $E0
14
15     INITIAL   EQU $8000
16     MONITOR   EQU $8003
17     DRUCKEN   EQU $8006
18     *-----
19
20     0303: 18      20     IMPL1   CLC          ;2
21     0304: A5 E0   21     LDA      WERT      ;3
22     0306: 65 E0   22     IMPL11  ADC      WERT      ;3
23     0308: 20 03 80 23     JSR      MONITOR   ;---
24     030B: 90 F9   24     BCC     IMPL11    ;2/3
25     030D: 60      25     RTS
26     *-----
27
28     030E: A5 E0   28     IMPL2   LDA      WERT      ;3
29     0310: 0A      29     IMPL21  ASL          ;2
30     0311: 20 03 80 30     JSR      MONITOR   ;---
31     0314: 10 FA   31     BPL     IMPL21    ;2/3
32     0316: 65 E0   32     IMPL22  ADC      WERT      ;3
33     0318: 20 03 80 33     JSR      MONITOR   ;---
34     031B: 90 F9   34     BCC     IMPL22    ;2/3
35     031D: 60      35     RTS
36     *-----
37
38     031E: A5 E0   38     IMPL3   LDA      WERT      ;3
39     0320: 0A      39     IMPL31  ASL          ;2
40     0321: 20 03 80 40     JSR      MONITOR   ;---
41     0324: 90 FA   41     BCC     IMPL31    ;2/3
42     0326: E5 E0   42     IMPL32  SBC      WERT      ;3
43     0328: 20 03 80 43     JSR      MONITOR   ;---
44     032B: B0 F9   44     BCS     IMPL32    ;2/3
45     032D: 65 E0   45     ADC      WERT      ;3
46     032F: 60      46     RTS
47     *-----
48
49     0330: 20 00 80 49     START   JSR      INITIAL   ;---
50     0333: A9 02   50     LDA      #2
51     0335: 85 E0   51     STA      WERT
52     0337: 20 1E 03 52     LOOP    JSR      IMPL3
53     033A: 20 0E 03 53     JSR      IMPL2
54     033D: 20 03 03 54     JSR      IMPL1
55     0340: E6 E0   55     INC      WERT
56     0342: 20 03 80 56     JSR      MONITOR   ;---
57     0345: 10 F0   57     BPL     LOOP
58     0347: 20 06 80 58     JSR      DRUCKEN   ;---
59     034A: 60      59     RTS

```

75 Bytes



Zum Ausprobieren dieser Utility laden Sie mit BLOAD FLAG.MONITOR und BLOAD FLAG.MONITOR.TEST die beiden Programme, schalten den Drucker mit PR#1 ein und starten das Demo mit CALL 768.

# Tips und Tricks in Pascal

## Teil 2: Kann man den P-Code optimieren?

von Dieter Geiß

Um die Antwort gleich vorwegzunehmen: Sie lautet „ja“. Die Methoden, um dies zu erreichen, sind mannigfaltig und bedürfen keiner Spezialkenntnisse. Jeder Pascal-Programmierer, ob Anfänger oder Fortgeschrittener, kann sich leicht ein paar „goldene Regeln“ merken, die jedes Pascal-Programm kürzer und schneller machen. Um diese Regeln aufzustellen, muß man sich sehr gut im sog. P-Code auskennen. Genaue Untersuchungen zeigen nämlich, daß man den Code eines Programms durchaus verkürzen kann, ohne dessen Funktion zu ändern.

Ein einfaches Beispiel aus der Maschinensprache wäre etwa die Benutzung eines 1-Byte-Zählers, der innerhalb statt außerhalb der Zeropage liegt. Ein DEC-Befehl vermindert einen solchen Zähler, gleich wo er sich befindet – nur der Zeropage-DEC-Befehl ist sowohl kürzer als auch schneller.

### Was ist P-Code?

Bevor ein Programm ausgeführt werden kann, muß es vom Compiler übersetzt werden. Der Code, der bei diesem Übersetzungsablauf entsteht, ist kein 6502-Maschinencode, sondern ein Pseudo-Code (P = Pseudo) für eine fiktive bzw. hypothetische Maschine, die in dieser Form gar nicht existiert. Diese Maschine wird softwaremäßig emuliert. Dabei wird jeder P-Code-Befehl vom P-Code-Interpreter abgearbeitet. Diese Methode ist quasi eine Zwischenlösung zwischen reiner Übersetzung (z.B. vom BASIC- oder Pascal-Quelltext in die Maschinensprache) und reiner Interpretation (z.B. des Applesoft-Quelltextes durch den Applesoft-Interpreter).

Der P-Code selbst ist ein sehr kompakter Code, der äußerst viele 1-Byte-Befehle kennt, so z.B. für alle arithmetischen Ganzzahl- oder Fließkomma-Operationen. Dies erklärt auch, warum der Code von sehr langen Programmen relativ kurz ist und auch in Maschinen mit wenig Speicher (64K) ablaufen kann.

### Regel 1 (Kurze-Variablen-Regel)

Es gibt P-Code-Befehle, um auf die ersten Variablen, die in einem Programm oder einer Prozedur definiert wurden, zuzugreifen. Von diesen speziellen Befehlen gibt es 16 für globale Variablen und 16 für lokale Variablen. Diese 1-Byte-Befehle beinhalten dann nicht nur die Operation, sondern auch den Operanden.

Jeder dieser Befehle kann ein Wort (16 Bits) laden, also Variablen vom Typ Integer, Char, Boolean, Pointer und Adressen. Das bedeutet, daß die ersten 16 Variablen eines Programms oder einer Prozedur 1-Wort-Variablen sein sollten, also keine großen Arrays oder Records, Files oder Strings. Es kommt also lediglich auf die Reihenfolge der Definition der einzelnen Variablen an.

Wenn man dies beachtet, kann man bis zu zwei Bytes pro Ladebefehl einer solchen Variablen sparen. Ein Ladebefehl tritt z.B. bei der Zuweisung

```
A := B
```

auf. B wird geladen und dann in A gespeichert. **Beispiel 1** macht deutlich, daß hier der Code kürzer ist und trotzdem dasselbe leistet. Regel 1 möchte ich deswegen die **Kurze-Variablen-Regel** nennen. Sie be-

sagt, daß man kurze Variablen, d.h. ein Wort umfassende Variablen, für die ersten 16 Variablen wählen sollte, und zwar sowohl im Programm (globale Variablen) als auch in jeder Prozedur (lokale Variablen). Man hat also lediglich auf die Reihenfolge der Definition der Variablen zu achten.

Sind die ersten 16 Variablen deklariert, sollte man trotzdem nicht gleich die großen Variablen dahinter definieren, sondern weiterhin die kleinen zuerst. Denn bei den Variablen, deren relative Adresse im Speicher zwar größer als 16, aber noch kleiner als 127 ist, kann man pro Ladebefehl wenigstens ein Byte sparen.

Es wird zwar kaum ein Programm geben, das 127 globale Variablen benutzt, aber ein Array [1..100] of Integer kostet ja auch 100 Worte, ein String 1–128 Worte, so daß man eventuell schnell an die Zahl 127 herankommt. Für die Regel gilt nämlich immer die Anfangsadresse der Variablen, die sich im Bereich 1 bis 16 bzw. 17 bis 127 befinden sollte. Die Anfangsadresse jeder Variablen erhält man durch ein Compiler-Listing mit der (\*\$L\*)-Option (s.a. Apple Pascal Language Reference Manual, S. 65).

Man beachte, daß auch die Parameter einer Prozedur zu deren lokalen Variablen gerechnet werden. Dasselbe gilt für die Programm-Parameter Input und Output. Diese belegen die beiden ersten Adressen der globalen Variablen eines Programms. Sie sind jeweils nur ein Wort lang, da es Zeiger-Variablen sind, die auf die File-Deskriptoren von Input und Output zeigen sollten, aber in der UCSD-Version nicht benutzt werden. Um Input und Output anzusprechen, werden intern vielmehr direkt die Input- und Output-De-



skriptoren des Pascal-Systems über einen sog. „Intermediate“-Aufruf angesprochen. Dies soll hier aber nicht weiter ausgeführt werden.

Für Units gilt die Regel analog. Bei den Intrinsic-Units gibt es jedoch keine globalen Variablen, sondern die „Extended“-Variablen. Das sind diejenigen, die im Interface- oder im Implementation-Teil erklärt wurden und dann ein Data-Segment brauchen. Für diese Variablen gibt es keine ganz kurzen Ladebefehle, aber bei den ersten 127 (kurzen) Variablen kann man analog zu oben jeweils ein Byte pro Ladebefehl sparen.

Bei normalen Units gilt diese Regel nicht. Die dort erklärten Variablen gehören zum Hauptprogramm, sind also globale Variablen, deren Adresse erst durch den Linker festgelegt wird. Da der Compiler während des Übersetzens nicht weiß, wieviel Platz er reservieren muß, werden für jeden Ladebefehl 3 Bytes, also das Maximum, reserviert.

## Regel 2 (Lokale-Variablen-Regel)

Nun zur zweiten Regel, die ich **Lokale-Variablen-Regel** nennen möchte. Man sollte darauf achten, lieber viele lokale Variablen in einer Prozedur zu definieren, als Variablen zu benutzen, die um ein oder mehrere Glieder in der statischen Verweiskette zurückliegen. Das Zurückgreifen auf diese Intermediate-Variablen kostet nämlich drei bis vier Bytes gegenüber einem Byte bei einer kurzen lokalen Variablen.

Benutzt man also in mehreren Prozeduren, die den gleichen statischen Vorgänger haben, eine Indexvariable I, so empfiehlt es sich, diese nicht in der Vorgängerprozedur ein einziges Mal zu erklären, sondern jeweils einmal lokal in jeder der Prozeduren. Das schützt auch vor ungewollten Seiteneffekten. Zu dieser Regel siehe auch **Beispiel 2**.

## Regel 3 (Record-Regel)

Die **Record-Regel** besagt etwas Ähnliches wie die Kurze-Variablen-Regel. In einem Record werden die ersten 8 (kurzen) Variablen bevorzugt behandelt und mit kurzen Ladebefehlen geholt. Dies gilt aber nur dann, wenn dessen Adresse auf dem Stack liegt, also wenn der Zugriff auf ein Element des Records über eine Zeigervariable (Pointer) oder über einen Var-Parameter einer Prozedur erfolgt. Im übrigen sollte man in einem solchen Fall zusätzlich, sooft es geht, einen With-Befehl benutzen.

meter einer Prozedur erfolgt. Im übrigen sollte man in einem solchen Fall zusätzlich, sooft es geht, einen With-Befehl benutzen.

**Beispiel 3** zeigt die Ausnutzung der Record-Regel. Auch hier werden die Variablen mit Adresse 8 bis 127 mit kürzeren Befehlen behandelt als die restlichen.

## Regel 4 (Leer-String-Regel)

Wenden wir uns jetzt den Strings zu. Dort gibt es ebenfalls eine Reihe von Regeln, die man beachten sollte. Die **Leer-String-Regel** gilt für den Test, ob ein String leer ist oder nicht. Grundsätzlich gibt es dazu zwei Möglichkeiten:

- If S = '' then...
  - If Length (S) = 0 then...
- wobei S vom Typ String ist. Obwohl der zweite Ausdruck komplizierter aussieht, erzeugt er weniger Code.

## Regel 5 (Concat-Regel)

Die **Concat-Regel** für Strings besagt, daß der Concat-Befehl möglichst vermieden werden sollte. Will man einen String an den Anfang oder das Ende eines anderen Strings setzen, dann sollte man das mit dem Insert-Befehl erledigen, d.h. im Klartext: Statt

```
S := Concat (S, T)
sollte man
Insert (T, S, Length (S) + 1)
```

und statt

```
S := Concat (T, S)
sollte man
Insert (T, S, 1)
```

benutzen. Der Grund für diese Regel liegt darin, daß es in UCSD-Pascal (im Gegensatz zu Modula-2 unter dem UCSD-System) nicht erlaubt ist, Funktionen zu definieren, deren Funktionswert mehr als zwei Worte (für Reals) beträgt. Nun ist aber Concat eine solche Funktion. Intern wird der Aufruf von Concat durch einen Prozeduraufruf ersetzt. Dazu muß der Compiler eine Hilfsvariable (Funktionswert) anlegen, die zudem erst einmal initialisiert werden muß. Ein Concat-Aufruf mit zwei Argumenten resultiert dann in zwei internen Aufrufen einer Concat-Prozedur (nicht Funktion), die eigentlich „Sconcat“ heißt und im Pascal-System selbst definiert ist (Segment #0, Prozedur #23). Bei Insert dagegen erfolgt der Aufruf ohne Hilfsvariable.

## Regel 6 (Konstanten-Regel)

Die **Konstanten-Regel** gilt für String-, Long-Integer- und Set-Konstanten. Bei den String-Konstanten (const KString = 'Dies ist ein konstanter String') wird jedesmal, wenn eine solche Konstante einer String-Variablen zugewiesen wird, der konstante String im Code eingebettet. Bei zehn Zuweisungen des gleichen konstanten Strings an zehn Variablen wird also der zehnfache Speicherplatz des *einen* konstanten Strings „verbraten“. Weist man jedoch diesen konstanten String zuerst einer neuen Stringvariablen zu

```
KStrVar := KString
und benutzt diese zur Zuweisung an die zehn anderen Variablen, kann man den Code kompakter machen.
```

Bei den Long-Integer-Konstanten verfährt der Compiler denkbar ungünstig. Hat man eine konstante Long-Integer-Zahl (const MaxLong = 999999...) mit z.B. 36 Stellen definiert und weist diese dann einer Long-Integer-Variablen zu (z.B. Long := MaxLong), so ergibt sich etwa derselbe Code, als wenn man ähnlich dem Horner-Schema den Ausdruck

```
Long := ((9999 * 10000 + 9999) * 10000 + 9999) * 10000...
```

berechnen würde. Dabei werden die Integer-Konstanten 9999 und 10000 vorher in Long-Integer-Konstanten umgewandelt. Das ganze kostet dann über 190 Bytes für die eine einzige Zuweisung! Hat man z.B. zehn solcher Zuweisungen, so werden fast 2K benötigt. Eine Verkürzung ist möglich, wenn man wie bei den konstanten Strings verfährt und nur einmal einer Variablen die konstante Long-Integer-Zahl zuweist

```
MaxLongVar := MaxLong
und dann immer diese Variable statt der Konstanten benutzt, d.h.
```

```
Long := MaxLongVar
statt
```

```
Long := MaxLong.
Ähnliches gilt auch für konstante Mengen (Set). Statt zwanzigmal eine Abfrage wie
If C in ['a'..'z'] then...
```

zu benutzen, sollte man eine neue Variable einführen

```
Var Klein: Set of Char
und diese dann initialisieren
```

```
Klein := ['a'..'z'],
um die Abfrage in
```

```
If C in Klein then...
umwandeln zu können.
```

## Regel 7

(Compiler-Regel)

Die **Compiler-Regel** bezieht sich auf die beiden Optionen (\*\$R-\*) und (\*\$I-\*), die auch im Apple Pascal Language Reference Manual auf den Seiten 67 und 63 beschrieben werden. Mit R- wird das sogenannte Range-Checking unterbunden, d.h. Subrange-, Array- und String-Zugriffe werden nicht mehr dahingehend überprüft, also ob sie in den jeweils gültigen Grenzen liegen oder nicht.

Die (\*\$R-\*)-Option bringt immerhin drei bis fünf Bytes pro Array- oder Subrange-Zugriff. Dies spart bei sehr langen Programmen, die ständig auf Arrays zugreifen, möglicherweise mehrere tausend Bytes. Programme mit der R-Option laufen dann natürlich auch etwas schneller. Anwenden sollte man ein R-Option aber nur, wenn man sicher ist, daß Index und Subrange-Grenzen nicht über- oder unterschritten werden. Normalerweise kann man sich nur sicher sein, wenn man sein Programm verifiziert hat. Aber wer macht das schon?

Durch die (\*\$I-\*)-Option spart man zwei Bytes pro I/O-Befehl, d.h. bei jeder Read-, Write-, Get-, Put-, Reset-, Rewrite-Anweisung usw. Auch hier muß man mit größter Vorsicht arbeiten, da bei nicht stattfindender I/O-Prüfung ein Programm nicht abbricht, wenn ein entsprechender Fehler auftritt.

Hat man jedoch viele Bildschirmausgaben in seinem Programm, kann man dort getrost die I-Option einschalten, denn der Bildschirm wird als Ausgabegerät nie „offline“ sein, es sei denn, die Firmware der 80-Zeichenkarte wäre defekt.

Ähnliches gilt für die Eingabe von der Tastatur, die ja eigentlich immer da sein sollte. So kann man bei jeder Eingabe von der

Tastatur, also bei jedem Read (Input, ...), und jeder Ausgabe auf den Bildschirm, also bei jedem Write (Output, ...), die o.g. zwei Bytes einsparen.

## Regel 8

(Standard-Prozedur-Regel)

Es gibt noch weitere, jedoch mehr programmiertechnische Kniffe der Code-Komprimierung, die wir unter der **Standard-Prozedur-Regel** zusammenfassen wollen. Diese Kniffe verkürzen auch den Code, haben aber überdies den Vorteil, daß sie die gestellte Aufgabe zügiger erledigen. Es handelt sich dabei um die Benutzung der schnellen UCSD-Standard-Prozeduren Fillchar, Moveleft und Moveright. Als Aufgabe sei die Initialisierung eines zweidimensionalen Arrays gestellt, der die Grenzen 0..Xmax und 0..Ymax habe. Ein Algorithmus, um eine solche Matrix auf 0 zu setzen, ist einfach und im **Beispiel 4** nachzulesen. Die zweite Version ist aber wesentlich effektiver, spart etwas Code und noch mehr Zeit.

Um die Prozedur Moveright zu demonstrieren, sei die Aufgabe gestellt, ein Element in einen Array an der x-ten Stelle einzufügen. Statt nun mit einer For-Schleife alle Elemente hinter der x-ten Stelle nach hinten zu schieben, geht dies auch mit einem einzigen Befehl (**Beispiel 5**).

Das gleiche gilt für das Entfernen eines Elements aus einem Array. Auch hier genügt ein Befehl für das Verschieben (s. **Beispiel 6**).

Für die letzten drei Beispiele spielt die Sizeof-Funktion eine große Rolle. Benutzt man diese anstelle einer konstanten Zahl, so kann man sichergehen, daß nicht aus Versehen zuviel gelöscht oder verschoben wird.

## Zusammenfassung

Zum besseren Überblick hier noch einmal alle Regeln kurz zusammengefaßt:

**Kurze-Variablen-Regel:** Durch eine geschickte Reihenfolge definiere man die kurzen (1-Wort-Variablen) jedes Programms und jeder Prozedur zuerst.

**Lokale-Variablen-Regel:** Man benutze möglichst lokale Variablen.

**Record-Regel:** In einem Record definiere man ebenfalls die kurzen Variablen zuerst.

**Leer-String Regel:** Man benutze zum Testen leerer Strings die Length-Funktion.

**Concat-Regel:** Man ziehe die Insert-Prozedur der Concat-Funktion vor.

**Konstanten-Regel:** Man benutze nicht zu viele Zuweisungen von String-, Long-Integer- und Set-Konstanten.

**Compiler-Regel:** Die beiden Optionen R- und I- sollten sooft wie möglich benutzt werden, aber nur wenn Indexüberschreitungen unmöglich sind bzw. kein I/O-Fehler in einem (Teil-)Programm auftreten kann.

**Die Standard-Prozedur-Regel:** Man programmiere mit Hilfe der Standard-Prozeduren Fillchar, Moveleft und Moveright sowie der Standard-Funktion Sizeof statt mit For-Schleifen.

Für Pascal 1.2 (128K-System) gibt es noch eine Möglichkeit, eventuell viel Speicher zu sparen, da es dort im Speicher Bereiche geben kann, die miteinander identisch sind. Von diesen Bereichen kann dann die Kopie gelöscht werden (Crunching). Doch dazu mehr im nächsten Beitrag.



Für IIC und IIE mit 64K-Karte

## SUPERPLOT

**Double-Hires-Utility**

von Karl-Walter Bott, 1984, Programmdiskette und Manual, DM 48,-

SUPERPLOT ist eine neue, ungewöhnlich kompakte und schnelle Ampersand-Utility für Double Hires, die einschließlich eines vollständigen ASCII-Shape-Zeichensatzes wahlweise in Bank 1 oder Bank 2 der Language Card liegt und damit sowohl unter ProDOS als auch unter DOS 3.3, falls letzteres in die LC-Bank geschoben wurde, benutzt und in eigene Applesoftprogramme integriert werden kann. SUPERPLOT unterstützt die üblichen HGR-Befehle, denen lediglich ein & vorangestellt werden muß, also z. B. & HPLLOT 500, 100 TO 500, 150 usw. SUPERPLOT ist speziell für das Plotten von beschrifteten wissenschaftlichen Funktionskurven mit hoher Auflösung gedacht und weniger für HGR-Spiele.

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

## Beispiele zur P-Code-Optimierung

### Beispiel 1

```

1  (*$L BSP1ALST.TEXT*)
1  program Bsp1a (input, output);
3
3  var A : array [3..127] of integer;
128 I : integer; (*beginnt bei Adresse 128*)
129 J : integer;
130 K : integer;
131
0  begin
0  I := J;
8  J := K;
14 K := I;
20 A [3] := 0
30 end.

1  (*$L BSP1BLST.TEXT*)
1  program Bsp1b (input, output);
3
3  (*Das Programm leistet dasselbe wie Bsp1a,
3  ist aber 8 Byte kuerzer*)
3
3  var I : integer; (*beginnt bei Adresse 3*)
4  J : integer;
5  K : integer;
6  A : array [3..127] of integer;
131
0  begin
0  I := J;
5  J := K;
8  K := I;
11 A [3] := 0
21 end.

```

### Beispiel 2

```

1  (*$L BSP2ALST.TEXT*)
1  program Bsp2a (input, output);
3
1  procedure Aussen;
1
1  var I : integer; (*Laufvariable fuer 3 Prozeduren*)
2
1  procedure Innen1;
1
0  begin
0  for I := 0 to 9 do (*irgendwas*)
14 end;
38
1  procedure Innen2;
1
0  begin
0  for I := 10 to 19 do (*irgendwas*)
14 end;
38
1  procedure Innen3;
1
0  begin
0  for I := 20 to 29 do (*irgendwas*)
14 end;
38
0  begin
0  end;
12
0  begin
0  end.

1  (*$L BSP2BLST.TEXT*)
1  program Bsp2b (input, output);
3
3  (*Das Programm leistet dasselbe wie Bsp2a,
3  ist aber 18 Byte kuerzer*)
3
1  procedure Aussen;
1
1  procedure Innen1;
1
1  var I : integer; (*lokale Laufvariable*)
2
0  begin
0  for I := 0 to 9 do (*irgendwas*)
11 end;
32
1  procedure Innen2;
1
1  var I : integer; (*lokale Laufvariable*)
2

```

```

0  begin
0  for I := 10 to 19 do (*irgendwas*)
11 end;
32
1  procedure Innen3;
1
1  var I : integer; (*lokale Laufvariable*)
2
0  begin
0  for I := 20 to 29 do (*irgendwas*)
11 end;
32
0  begin
0  end;
12
0  begin
0  end.

```

### Beispiel 3

```

1  (*$L BSP3ALST.TEXT*)
1  program Bsp3a (input, output);
3
3  type Rec = record
3  A : array [0..127] of integer;
3  I : integer; (*beginnt bei Adresse 128*)
3  J : integer;
3  K : integer;
3  end;
3
3  var Recl : Rec;
134
1  procedure Doit (var R : Rec);
2
0  begin
0  if R.I = R.J then
11 if R.J = R.K then R.A [0] := 1
29 end;
44
0  begin
0  Doit (Recl)
4 end.

1  (*$L BSP3BLST.TEXT*)
1  program Bsp3b (input, output);
3
3  type Rec = record
3  I : integer; (*beginnt bei Adresse 0*)
3  J : integer;
3  K : integer;
3  A : array [0..127] of integer;
3  end;
3
3  var Recl : Rec;
134
1  procedure Doit (var R : Rec);
2
0  begin
0  with R do
3 if I = J then
10 if J = K then A [0] := 1
26 end;
40
0  begin
0  Doit (Recl)
4 end.

```

### Beispiel 4

```

1  (*$L BSP4LST.TEXT*)
1  program Bsp4 (input, output);
3
3  var A : packed array [0..100, 0..100] of boolean;
710
1  procedure Bsp4a;
1
1  var I : integer;
2  J : integer;
3
0  begin
0  for I := 0 to 100 do
11 for J := 0 to 100 do A [I, J] := false
37 end;
70
1  procedure Bsp4b;
1
1  (*kostet bei gleicher Leistung 48 Bytes
1  weniger als Bsp4a*)

```

```

1
0 begin
0 fillchar (A, sizeof (A), false)
9 end;
22
0 begin
0 end.

```

#### Beispiel 5

```

1 (*$L BSP5LST.TEXT*)
1 program Bsp5und6 (input, output);
3
3 type Element = integer; (*oder ein record...*)
3
3 var A : array [0..100] of Element;
104
1 procedure Bsp5; (*Einfuegen*)
1
1 procedure Bsp5a (X : integer; E : Element);
3
3 var I : integer;
4
4 begin
0 for I := 99 downto X do A [I + 1] := A [I];
38 A [X] := E
46 end;
62
1 procedure Bsp5b (X : integer; E : Element);
3
3 (*spart gegenueber 5a 14 Bytes*)
3
3 begin
0 moveright (A [X], A [X + 1], (100 - X)
0 * sizeof (Element));
27 A [X] := E
35 end;
50

```

```

0 begin
0 end;
12
1 procedure Bsp6; (*Ausfuegen*)
1
1 procedure Bsp6a (X : integer);
2
2 var I : integer;
3
3 begin
0 for I := X + 1 to 100 do A [I - 1] := A [I];
40 end;
54
1 procedure Bsp6b (X : integer);
2
2 (*spart gegenueber 6a 12 Bytes*)
2
2 begin
0 moveleft (A [X + 1], A [X], (100 - X)
0 * sizeof (Element));
27 end;
40
0 begin
0 end;
12
0 begin
0 end.

```

**Hinweis:** Die Zahlen an den Zeilenanfängen sind die Word-Offsets für den Speicher bzw. Byte-Offsets für den Code.

# SUPERQUICK

## Ein superschnelles Disketten-Kopierprogramm

von Arne Schäpers, 1985, Programmdiskette mit Anleitung, DM 48,-

Mit SUPERQUICK ist es möglich, Disketten jeden Formats (DOS 3.3, ProDOS, UCSD-Pascal und CP/M) in einer unglaublich kurzen Zeit von nur 29 Sekunden (mit Formatierung) zu kopieren. Bei entsprechender Speichererweiterung kann der gesamte Disketteninhalt eingelesen werden, um mehrere Kopien anzufertigen. Die Zeit für eine Einzelkopie reduziert sich dann auf sage und schreibe 19 Sekunden.

SUPERQUICK erkennt die 64K-Karte (in Slot 3) des Apple IIe und IIc sowie eine 16K-Language-Card in Slot 0 und bezieht diese selbständig als Datenpuffer ein. Darüber hinaus werden die IBS-Karten AP17 in den Ausbaustufen 64K bis 256K automatisch unterstützt und gegebenenfalls als weitere Puffer eingesetzt.

Eine Anpassung an Laufwerke mit höherer Spurenzahl (bis 80 Spuren, 160-Spurlaufwerke können nach manuellem Side-Select ebenfalls betrieben werden) ist ohne Schwierigkeiten aus dem Programm-Menü heraus möglich.

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

**NEU**



Vorname, Name  
Beruf  
Straße  
Wohnort  
PLZ

Bitte veröffentlichen Sie den umstehenden Text von \_\_\_\_\_ Zeilen à \_\_\_\_\_ DM in der nächsterreichbaren Ausgabe von »peeker«

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

Datum Unterschrift

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

Karte bitte vollständig ausfüllen

Vorname, Name  
Firma  
Straße  
PLZ/Ort  
Telefon mit Vorwahl

## ANTWORTKARTE

**peeker-Börse**  
Anzeigen-Service  
Dr. Alfred Hüthig Verlag  
Postfach 10 28 69  
6900 Heidelberg 1

## POSTKARTE

Firma  
Straße  
PLZ/Ort

## POSTKARTE

**peeker**  
Redaktion  
Postfach 10 28 69  
6900 Heidelberg 1



# Produkt-Karte

Wünschen Sie weitere Informationen zu einem der im Heft vorgestellten Produkte ?

Nichts einfacher als das.  
Produkt-Karte ausfüllen, mit 60-Pfennig frankieren und absenden.

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/Herstellers und Ihre vollständige Firmenanschrift ein.

---

# *Fotosatz ohne Umwege!*

*Durch Verknüpfung von EDV und Fotosatzanlage  
Vorteile über Vorteile – für Autoren und Verlage*

---

Sind Ihre Texte schon erfaßt oder möchten Sie diese selbst auf einem Computer erfassen?

Dann sprechen Sie mit uns!

Wir lesen und konvertieren Ihre Daten zu Fotosatz.

Bei uns finden Sie Beratung und Service zur Buchherstellung vom Satz über den Druck bis zum Einband.



Zechnersche Buchdruckerei

6720 Speyer · Daimlerstraße 9 · Postfach 2080

Telefon (06232) 33076-79 · Telex 465167

Das Microsoft BASIC-80, auch MBASIC genannt, besitzt nicht nur eine große Verbreitung – MBASIC-Interpreter sind für fast alle Rechner verfügbar, die ein Betriebssystem wie CP/M oder MS-DOS benutzen – sondern auch etliche Befehle, die im Applesoft „durch Abwesenheit glänzen“ und nachträglich gepatcht werden müssen. Dazu gehören die formatierte Zahlengabe (PRINT USING), die INSTR-Funktion, die Möglichkeit zum Löschen von Feldern und vieles mehr. Diese Erweiterungen vorzustellen und den erfahrenen Programmierer auf weitere syntaktische Unterschiede zum Applesoft hinzuweisen ist Ziel dieses Artikels.

# MBASIC für den Applesoft-Profi

von Jörg Lange

## 1. Laden des MBASIC-Interpreters

Wie oben schon erwähnt, arbeitet MBASIC unter dem Betriebssystem CP/M, so daß zur Arbeit mit dieser Sprache neben der CP/M-2.2-Masterdisk, auf der sich der Interpreter befindet, auch eine im Apple installierte „Z80-Softcard“ benötigt wird.

Hinweise hierzu sowie eine Einführung in die CP/M-Befehle, deren Kenntnisse aber für den Anfang noch nicht notwendig sind, finden Sie z.B. in dem Artikel „CP/M für Einsteiger“ im Peeker 4/1985.

Nach dem Booten der CP/M-2.2-Masterdisk meldet sich der Rechner mit dem CP/M-Prompt „A>“. Der Befehl „MBASIC <Return>“ startet den BASIC-Interpreter.

Auf dem Bildschirm erscheint daraufhin eine Titelmeldung („BASIC-80 Rev. 5.2 ...“) sowie das Bereitschaftszeichen „Ok“, das nach jeder erfolgreichen Programm- oder Befehlsausführung ausgegeben wird.

## 2. Verlassen des MBASIC

Bevor es in das Detail geht, soll noch verraten werden, wie man MBASIC wieder verläßt:

**SYSTEM:** Dieser Befehl bewirkt – nach der Schließung noch offener Dateien – die Rückkehr in das CP/M-Betriebssystem. *Programm und Variablen gehen dabei verloren!*

## 3. Variablentypen und Typvereinbarungen

Neben den bekannten Variablentypen String (A\$) und Integer (A%) stellt MBASIC für Gleitkommazahlen (Real) zwei AbSpeicherungsmöglichkeiten zur Verfügung: die einfach genaue Real-Variable (A oder A!) mit 7 signifikanten Stellen, von denen im Regelfall aber höchstens 6 ausgegeben werden, sowie den „Doubleprecision“-Typ (A#), der intern mit bis zu 17 Stellen verarbeitet wird. Ausgegeben werden aber hier auch nur maximal 16 Stellen.

Neben der Typenvereinbarung durch ein an den Variablennamen angehängtes Zeichen (Suffix) wie z.B. „\$“ oder „#“ gibt es die Möglichkeit, mit den DEF-Befehlen global Variablennamen mit bestimmten Anfangsbuchstaben einem Typ zuzuordnen.

```
10 REM Beispiel 1
20 DEFDBL I-M
30 I = 2456.23
40 J% = 54.46
50 K = 145.35
```

Durch den DEFDBL-Befehl in Zeile 20 werden alle Variablen, deren erster Buchstabe im Bereich „I“ bis „M“ liegt, im ganzen Programm zu Realvariablen doppelter Genauigkeit erklärt. Eine Suffix-Vereinbarung hat jedoch Vorrang, so daß in unserem Beispiel I und K dem doppelgenauen Typ zugeordnet werden, J% aber eine Integervariable bleibt. Eine Übersicht über Variablentypen und





## 7. Arrays

Die Dimensionierung von Feldern mit dem DIM-Befehl geschieht genauso wie im Applesoft. Neu ist jedoch dieser Befehl:

**ERASE FELD1, FELD2,...**: Die Arrays mit den angegebenen Namen werden gelöscht und können neu dimensioniert werden, ohne daß man einen „Duplicate Definition“-Fehler erhält.

## 8. Befehle zur Programmablaufsteuerung

Wie im Applesoft stehen unter MBASIC die Befehle GOTO, GOSUB, ON... GOTO, ON...GOSUB, RETURN und POP (letzterer ist nur in der Appleversion des MBASIC implementiert!) zur Verfügung. Bei allen diesen Anweisungen ist jedoch auf das notwendige Einfügen von Leerzeichen (s.o.) zu achten.

### IF...THEN mit Alternative

Neben dem üblichen IF-Befehl gibt es eine weitere Form:

**IF logischer Ausdruck THEN Befehlsliste1 ELSE Befehlsliste2**: Ist der logische Ausdruck wahr, wird die Befehlsliste 1 ausgeführt. Im anderen Fall verzweigt das Programm zur Befehlsliste2. Jede dieser Listen darf mehrere Befehle – wie im Applesoft durch „;“ getrennt – enthalten; die gesamte IF-THEN-ELSE-Sequenz muß aber innerhalb einer Zeile stehen. Beispiel:

```
20 IF X < 10 THEN 200 ELSE PRINT "ERROR!": GOTO 5
```

Erwähnt sei in diesem Zusammenhang, daß logisch „wahr“ im MBASIC dem Wert -1 entspricht (Applesoft: +1): PRINT 3 = 3 ergibt -1!

### Schleifen mit WHILE...WEND

Für Programmschleifen existiert im MBASIC neben der „klassischen“ „FOR-NEXT“-Möglichkeit die WHILE-WEND-Anweisung, die vorzugsweise dann angewandt wird, wenn die Zahl der Durchläufe vor Schleifenbeginn nicht feststeht:

**WHILE logischer Ausdruck /Programmzeilen/ WEND**: Nach Prüfung des logischen Ausdrucks wird zur nächsten Programmzeile nach dem WEND-Befehl verzweigt, wenn dieser nicht wahr ist. Andernfalls werden die von „WHILE“ und „WEND“ eingeschlossenen Anweisungen ausgeführt. Danach wird erneut der logische Ausdruck ausgewertet usw.

```
10 REM Beispiel 2 / WHILE/WEND
20 PRINT "WEITER ? J/N ?"; E$ = ""
```

```
30 WHILE E$ <> "J" AND E$ <> "N"
40 GET E$
50 WEND
```

Zu FOR...NEXT ist noch ein wichtiger Nachtrag zu machen: MBASIC prüft vor dem Schleifendurchlauf die Zählvariable, d.h. eine Anweisung des Typs „FOR L = 3 TO 1“ wird nicht ausgeführt. Im Applesoft dagegen erfolgt immer mindestens ein Durchlauf.

## 9. Anweisungen zur Fehlerbehandlung

Vom Applesoft her bekannte Anweisungen sind im MBASIC z.T. in erweiterter Form vorhanden:

**ON ERROR GOTO n**: Beim Auftreten eines Fehlers wird in eine Fehlerbehandlungsroutine in Zeile *n* verzweigt. ON ERROR GOTO 0 entspricht dem Applesoft-POKE 216,0: Die normale Fehlerbehandlung (Ausgabe von Fehlermeldungen) wird wieder aktiviert.

*Eine Fehlerbehandlungsroutine muß mit einer RESUME-Anweisung enden:*

**RESUME** bzw. **RESUME 0**: Die Programmausführung wird in der Zeile wieder aufgenommen, in der der Fehler auftrat. Anm.: Die applesofttypischen Schwierigkeiten in Verbindung mit diesem Befehl, z.B. mit FOR-NEXT-Schleifen, gibt es im MBASIC nach eigener Erfahrung nicht.

**RESUME NEXT**: Wie RESUME, nur wird hier die nachfolgende Zeile angesprungen.

**RESUME n**: Das Programm wird mit der Zeile *n* fortgesetzt.

**X = ERR**: Der Variablen X wird die Nummer des aufgetretenen Fehlers zugewiesen (Applesoft: X = PEEK (222)). MBASIC kennt ungefähr vierzig verschiedene Fehler, auf die hier aus Platzgründen nicht näher eingegangen werden kann.

**X = ERL**: Die Variable ERL enthält die Nummer der Zeile, in der der letzte Fehler auftrat. Das Applesoft-Äquivalent ist X = PEEK (218) + PEEK (219) \* 256. Ein Tip: Um bei einer Neunummerierung des Programmes mit der RENUM-Anweisung nicht alle ERL-Abfragen von Hand ändern zu müssen, sollte stets die Form „IF ERL = Zeile THEN“ benutzt werden, da derartige Ausdrücke von der RENUM-Funktion korrigiert werden, nicht aber z.B. „IF 100 = ERL THEN“.

**ERROR n**: Ein Fehler mit der Fehlernummer *n* wird künstlich erzeugt. Dies dient im wesentlichen zum Testen von Fehler Routinen. Hinweis: *n* muß keine legale Fehlernummer des Interpreters sein, d.h. der

Programmierer kann neue Fehler mit eigenen Nummern (empfohlener Bereich: 80 < *n* < 255) einführen.

## 10. Funktionen im MBASIC-Sprachumfang

### Arithmetische Ausdrücke

MBASIC kennt neben den üblichen Rechenarten die Integerdivision ( $5 \setminus 2 = 2$ , dabei wird „\“ durch <Ctrl-B> erzeugt, auf deutschen Tastaturen entspricht „\“ dem „Ö“) und die MODULO-Operation. Hierbei wird der ganzzahlige Rest einer Division ausgegeben:  $9 \text{ MOD } 4 = 1$ . Wichtig: Beide Funktionen wandeln die Argumente vor der Berechnung in Integerwerte um, d.h.  $12.4 \text{ MOD } 3.9$  wird zu  $12 \text{ MOD } 3 = 0!$

### Mathematische Funktionen

Alle unter Applesoft vorhandenen Funktionen – inklusive der DEF FN-Anweisung zur Selbstdefinition durch den Programmierer – sind implementiert. Neu hinzu kommt:

**Y = FIX(X)**: Die Nachkommastellen von X werden abgeschnitten (Beispiel:  $\text{FIX}(1.9) = 1$ ).

Die RND-Funktion zur Erzeugung von Zufallszahlen ist ebenfalls vorhanden:

**Y = RND(X)**: Ist  $X > 0$ , so wird eine neue Zufallszahl erzeugt, bei  $X = 0$  wird die letzte Ausgabe wiederholt. Der große Unterschied zum Applesoft liegt darin, daß bei jedem Neustart des Programmes dieselben Zahlen ermittelt werden, wenn der Zufallszahlengenerator nicht mit einem speziellen Befehl „eingeschaltet“ wird:

**RANDOMIZE INTEGER%**: In Abhängigkeit des Inhaltes von INTEGER% wird der Startwert der Zufallszahlenreihe erzeugt. Wird keine Variable angegeben – also nur z.B. „10 RANDOMIZE“ geschrieben – hält das Programm mit der Meldung „Random Number Seed (-32768 to 32767)?“ an und erwartet die Eingabe einer entsprechenden Zahl.

### Funktionen zur Zeichenkettenverarbeitung

Zusätzlich zu RIGHT\$, LEFT\$, MID\$, LEN, VAL, STR\$, ASC und CHR\$ sind vorhanden:

**X = INSTR (START, ORIGINAL\$, VERGLEICH\$)**: Es wird – beginnend bei der Position START – ermittelt, ob der VERGLEICH\$ in ORIGINAL\$ vorhanden ist. Bei positivem Ergebnis wird X die Positionsnummer des ersten übereinstimmen-

den Buchstabens zugewiesen. Andernfalls erhält X den Wert Null. Der START-Parameter braucht nicht angegeben zu werden. Beispiel: INSTR("TEST", "ES") = 2; INSTR(4, "APPLE", "P") = 0.

**X\$ = SPACES\$(n):** Es werden *n* Leerzeichen erzeugt und der Variablen X\$ zugewiesen.

**X\$ = STRINGS\$(n, QUELLES):** Das erste Zeichen von QUELLES wird *n*-mal in X\$ abgespeichert. Beispiel:

STRINGS\$(3, "123") ergibt „111“.

Neben der MID\$-Funktion gibt es in MBASIC auch einen MID\$-Befehl, der das Austauschen von Zeichen in einer bestehenden Zeichenkette erlaubt:

**MID\$(ZIEL\$, p, n) = INSERT\$:** Der Inhalt von INSERT\$ wird ab der Position *p* in ZIEL\$ kopiert. Dabei werden höchstens *n* Zeichen ausgetauscht. Beispiel:

X\$ = "ABCD"; MID\$(X\$, 2, 2) = "012" ergibt in X\$ die Zeichenkette „A01D“.

### Logische Funktionen auf Bitebene

Integerzahlen können über Funktionen wie AND, OR, XOR usw. verknüpft werden. Im Rahmen dieses Artikels soll darauf aber nicht näher eingegangen werden.

## 11. „Schnittstellen“ für Assemblerprogrammierer

MBASIC enthält neben PEEK und POKE einen CALL-Befehl, der sowohl den Aufruf von Z80- als auch von 6502-Assembler-routinen ermöglicht, die USR- und die VARPTR-Funktion. Mit letzterer kann die Speicheradresse einer Variablen ermittelt werden. Genauere Angaben zu diesem Thema entnehmen Sie bitte der entsprechenden Fachliteratur, z.B. Literatur (1).

## 12. Befehle zur Ein- und Ausgabe

### Einlesen von der Tastatur

Hier wird der Applesoftprogrammierer nicht nur mit neuen Befehlen, sondern auch mit einer geänderten Syntax der INPUT-Anweisung konfrontiert:

**INPUT "Zahl: "; N:** Im Gegensatz zum Applesoft erscheint hier die Meldung „Zahl: ?“ auf dem Bildschirm. Will man das Fragezeichen unterdrücken, muß hinter dem auszugebenden Text ein Komma stehen: **INPUT "Zahl: ", N!**

**INPUT; "Zahl: "; N:** Das zusätzlich eingefügte Semikolon gleich hinter der INPUT-Anweisung bewirkt, daß der Cursor nach erfolgter Eingabe in derselben Zeile bleibt.

**LINE INPUT "Text: "; TEXTE\$:** Dieser Befehl ist zum Einlesen ganzer Textzeilen

gedacht. Dabei wird die gesamte Eingabe bis zum <Return> als eine einzige Zeichenkette aufgefaßt und der Variablen TEXTE\$ zugewiesen. Dies bedeutet insbesondere, daß sowohl Kommata als auch Anführungszeichen eingegeben werden können, was bekanntlich bei der INPUT-Anweisung nicht – oder nur mit Tricks – möglich ist. Auch werden führende Leerzeichen nicht unterdrückt.

**X\$ = INPUT\$(n):** Diese Funktion liest von der Tastatur einen String der Länge *n*, ohne ihn auf dem Bildschirm darzustellen. Die Eingabe kann auch durch <Return> nicht vorzeitig beendet werden. Vorzüglich geeignet zum Abfragen von Kennworten: A\$ = INPUT\$(6): IF A\$ = "PEEKER" THEN... .

**GET E\$:** Dieser Befehl funktioniert wie im Applesoft, ist aber in etlichen MBASIC-Interpretern für andere Computer nicht implementiert.

**X\$ = INKEY\$:** Wie bei der GET-Anweisung erfolgt eine Tastaturabfrage. Dabei wartet die Funktion jedoch nicht auf eine Eingabe des Benutzers, sondern gibt, wenn keine Taste gedrückt wird, einen Leerstring aus ("").

### Ausgabe – auch formatiert

Wer in Applesoft schon einmal versucht hat, eine Zahlentabelle ordentlich auszugeben, weiß, daß dies – wenn man nicht einen entsprechenden Patch „im Hause“ hat – viel Arbeit verursachen kann. MBASIC schafft da Abhilfe, indem es neben den üblichen Befehlen zur Ausgabe (PRINT, SPC- und TAB-Funktion sowie POS(0) zur Ermittlung der aktuellen Ausgabespalte) die PRINT USING-Anweisung zur Verfügung stellt:

**PRINT USING Formatstring; Variablenliste:** Eine Variablenliste wird entsprechend dem Formatstring ausgegeben.

Wie sieht nun ein solcher Formatstring aus? Nehmen wir ein Beispiel zu Hilfe: Sie haben ein Programm, das Rechnungen in folgender Form schreiben soll:

```
5.00 kg Mehl 1.50 DM 7.50 DM
14.00 l Milch 0.99 DM 13.86 DM
```

Die erforderlichen Daten seien in den Variablen MENGE, EINHEIT\$, ARTIKEL\$ und EINZELPREIS abgespeichert. Der dafür erforderliche Befehl würde lauten:

```
10 PRINT USING "##.##□\□\
□□□□\□#.##□DM□##.##□DM";
MENGE; EINHEIT$; ARTIKEL$; EINZEL-
PREIS; EINZELPREIS * MENGE. (□ =
Space)
```

MBASIC druckt dabei den Formatstring aus, ersetzt aber vorher gewisse Platzhalterzeichen durch Variablen aus der Variablenliste. So steht die erste Platzhaltergruppe (##.##) für eine Zahlenvariable, die mit zwei Vor- und zwei Nachkommastellen ausgegeben werden soll. Der Interpreter setzt hier den Inhalt der ersten Variablen der Liste – MENGE – ein. Alle durch „\“ begrenzten Positionen werden – „\“ miteingeschlossen – zur Ausgabe von Zeichenketten reserviert. So steht also „\□\“ für zwei Buchstaben. In unserem Beispiel wird dort EINHEIT\$ ausgegeben. Ist eine Zeichenkette kleiner, als im Formatstring vorgesehen, so wird mit Leerzeichen aufgefüllt. Auf diese Art und Weise wird der gesamte Formatstring abgearbeitet. Der Abschnitt L der Referenz-tabelle gibt einen Überblick über die gebräuchlichsten Platzhalterzeichen und ihre Funktion.

*Ein Hinweis:* Achten Sie bei der Verwendung von PRINT USING darauf, daß Anzahl und Art der reservierten Felder mit Anzahl und Typ der Variablen der Liste übereinstimmen. Da die Sache insgesamt nicht ohne Tücken ist, sollten Sie in Ihren Programmen mit kurzen Ausdrücken wie z.B. PRINT USING "#####.## DM"; BETRAG beginnen.

Hier noch einige weitere, auf die Ausgabe bezogene Befehle:

**WIDTH n:** Diese Anweisung setzt die maximale Anzahl von Spalten auf dem Bildschirm fest (ähnlich dem Applesoft POKE 33, n).

**HOME:** Es erfolgt eine Bildschirmlöschung; dieser Befehl arbeitet auch bei Verwendung von 80-Zeichenkarten korrekt.

### Ausgabe auf dem Drucker

Im Gegensatz zum Applesoft, bei dem die Ausgabe mit „PR#1“ auf den Drucker „umgeleitet“ wird, stellt MBASIC für diesen Zweck eigene Befehle bereit, die sich von den schon bekannten nur durch ein vorangestelltes „L“ unterscheiden: LPRINT und LPRINT USING. Schließlich wird mit „WIDTH LPRINT n“ die Ausgabebreite auf dem Drucker gesetzt; die LPOS(0)-Funktion gibt die aktuelle Position des Druckkopfes (Spalte) aus.

## 13. System- und Editierkommandos

Neben den gewohnten Befehlen – NEW, RUN, LIST sowie DEL, TRACE und NO-TRACE (letztere heißen in den MBASIC-Versionen anderer Rechner meistens DE-

LETE, TRON und TROFF) – gibt es auch hier einiges Neues:

**LLIST:** Diese Anweisung funktioniert wie LIST, nur erfolgt die Ausgabe auf dem Drucker.

**RENUM** *Neu, Alt, n:* Die Zeilennummern eines Programmes werden ab Zeile *Alt* geändert. Diese erhält die Zeilennummer *Neu*; alle weiteren Zeilen folgen im Abstand *n*. Einzelne Parameter können weggelassen werden, z.B. bedeutet RENUM „20: Alle Programmzeilen werden – beginnend mit Zeilennummer 10 – mit dem Abstand 20 neu nummeriert.

**AUTO** *Start, n:* Dieser Befehl – übrigens auch im Integer-BASIC des Apple enthalten – bewirkt die automatische Ausgabe von Zeilennummern im Abstand *n*. Erste Zeile ist Start. Auch hier können die Parameter „unterschlagen“ werden; AUTO alleine bewirkt dasselbe wie AUTO 10,10. Abgeschaltet wird dieses Programmierhilfsmittel mit Eingabe von <Ctrl-C>.

**CLEAR** *,Himem,Stack:* Während CLEAR alleine wie im Applesoft alle Variablen löscht, können über zusätzliche Parameter die höchste von BASIC benutzbare Speicheradresse (Himem) und die Stacklänge geändert werden. Letztere beträgt normalerweise 256 Bytes und sollte nur vergrößert werden, wenn zu viele GOSUB-Befehle im Programm einen „Out of Memory“-Fehler verursachen.

**PRINT FRE(0):** Es wird der freie Speicherplatz (in Bytes) angezeigt; im Gegensatz zum Applesoft jedoch erfolgt eine Garbage-Collection nur bei einem Aufruf der Form  $X = \text{FRE}(\text{““})!$

**RESTORE / RESTORE** *Zeile:* Neben einem globalen RESTORE kann in einer zweiten Form (z.B. RESTORE 100) der interne DATA-Zeiger auf das erste DATA-Element der genannten Zeile gesetzt werden – eine Möglichkeit, die im Applesoft auch durch nachträgliche Patchroutinen realisiert werden muß.

**EDIT** *n:* Diese Anweisung bringt die Zeile *n* in den Editiermodus. Weitere Aufrufformen des eingebauten Editors sind:

**EDIT .:** Die zuletzt eingegebene oder gelistete Zeile wird editiert.

**<Ctrl-A>:** Die gerade eingegebene, aber noch nicht mit <Return> abgeschlossene Programmzeile oder die letzte Direktanweisung wird in den Editor übernommen. Der damit aufgerufene Editor ist zeilenorientiert, d.h. nur innerhalb der angesprochenen Zeile können Änderungen vorgenommen werden. Die bekannten ESC-Sequenzen werden nicht akzeptiert. Der MBASIC-Editor gibt zuerst die Zeilennummer (die nicht geändert werden kann!)

der zu bearbeitenden Zeile aus und positioniert seinen Cursor vor dem ersten Befehl. Dieser kann jetzt durch <Space> nach rechts und „<-“ nach links bewegt werden. Dabei wird der Inhalt der Programmzeile erst sichtbar, wenn er vom Cursor überfahren wurde. Durch Eingabe von „L“ kann jedoch die Ausgabe des gesamten Zeileninhalts bewirkt werden. Eine Übersicht über die wichtigsten Editierkommandos gibt Abschnitt G der Referenztabelle. Ansonsten gilt für diesen Editor, der übrigens bei einem „Syntax-Fehler“ beim Programmablauf automatisch aktiviert wird, ein altbekanntes Sprichwort: „Nur Übung macht den Meister“.

#### 14. Befehle zur Dateienbehandlung

In MBASIC ist das DOS quasi als gleichberechtigter Partner von Anfang an integriert worden, d.h. alle Anweisungen zur Dateienbehandlung sind genauso Befehle wie PRINT, INPUT usw.:

```
100 REM Beispiel 3
110 RUN "TEIL2.BAS"
```

Das vom Applesoft bekannte PRINT CHR\$(4); „RUN TEIL2.BAS“ bewirkt dagegen nichts weiter als eine Bildschirmausgabe. Also: *In MBASIC-Programmen werden „DOS“-Anweisungen genauso behandelt wie alle anderen Befehle!* Dies mag am Anfang verwirrend sein, ist aber prinzipiell viel klarer und auch einfacher als im Applesoft.

Im Rahmen dieses Artikels soll dabei nur auf jene Anweisungen eingegangen werden, die sich auf Programmdateien beziehen (RUN usw.) oder darauf angewandt werden können (KILL u.ä.). Die Möglichkeiten des MBASIC zur Verwaltung sequentieller und Random-Access-Dateien werden aus Platzgründen erst in einer der nächsten Peeker-Ausgaben vorgestellt.

#### Dateinamen

Die im Vergleich zum Applesoft andere Behandlung von Dateibefehlen hat einen syntaktischen Unterschied zur Folge, der leicht übersehen wird und dann Verdrüß bringt: *Alle Dateinamen in BASIC-Befehlen sind entweder Stringkonstanten* (z.B. RUN „TEST“) *oder Stringvariablen* (A\$=„TEST“: RUN A\$). Dagegen erzeugt die „gewohnte“ Eingabe „RUN TEST“ nur einen „Type mismatch“-Fehler, da in diesem Fall TEST für MBASIC eine numerische Variable ist!

Die unter MBASIC verwendbaren Dateinamen müssen sich an die CP/M-Konventionen halten. Dabei gilt: Ein Dateiname be-

steht aus zwei Teilen, dem eigentlichen Namen, der bis zu acht Zeichen lang ist, und einer Typangabe aus drei Buchstaben. Beide werden durch einen „.“ getrennt (Beispiel: TEST.BAS, BUECHER.DAT). Die Typangabe soll dabei stets Auskunft über den Datentyp geben. So steht z.B. „BAS“ für ein BASIC-Programm, „TXT“ für einen Text usw.

Wichtig ist noch zu wissen, daß dem Dateinamen ein Laufwerkskennner (ein Buchstabe, gefolgt von einem „:“, z.B. „A:“) vorangestellt werden muß, wenn man auf eine Disk zugreifen will, die sich nicht in dem Laufwerk befindet, das bei dem Start des MBASIC das CP/M-Bezugslaufwerk war (meistens A:). CP/M ordnet folgendermaßen zu:

A: Slot 6, Drive 1; B: Slot 6, Drive 2  
So wird also mit RUN „B:TEST.BAS“ das Programm „TEST.BAS“ in Slot 6, Drive 2 angesprochen.

#### Befehle ausschließlich für Programmdateien

Die folgenden Anweisungen sind ausschließlich zum Arbeiten mit BASIC-Programmen gedacht. Deswegen ist bei diesen Befehlen das Weglassen der Dateitypangabe möglich. In einem solchen Fall erhält ein File automatisch den Typ „.BAS“. Beispiel: RUN „TEST“ ist gleichbedeutend mit RUN „TEST.BAS“.

**LOAD „NAME“ / RUN „NAME“:** Das Programm „NAME.BAS“ wird geladen bzw. geladen und gestartet.

**SAVE „NAME“:** Es erfolgt eine Abspeicherung des Programmes im Speicher unter dem Filenamen „NAME.BAS“. Dazu wird – wie auch beim Applesoft – eine spezielle Codierung benutzt, so daß derartige Dateien i.d.R. nicht mit einem Texteditor (z.B. Wordstar) bearbeitet werden können.

**SAVE „NAME“, A:** Im Gegensatz zum ersten Beispiel wird jetzt das Programm im ASCII-Format abgespeichert. Es kann dann auch editiert werden. Weiterhin wird diese Form zur Übertragung per Telefonmodem o.ä. empfohlen, insbesondere dann, wenn das Zielgerät kein Apple ist, weil sonst Kompatibilitätsprobleme auftreten können. (Für Interessierte: Bei den verschiedenen Interpretationen des MBASIC sind gleichen „Tokens“ z.T. unterschiedliche Befehle zugeordnet.) Schließlich akzeptiert auch der Microsoft-BASIC-Compiler nur ASCII-Dateien.

**SAVE „NAME“, P:** Diese Option bewirkt eine verschlüsselte Abspeicherung. Das Programm kann später nur noch geladen



und gestartet, aber weder gelistet noch geändert werden.

**MERGE "NAME":** Dieser Befehl, der nur auf Files im ASCII-Format (s.o.) angewendet werden darf, lädt das Programm „NAME“ über das im Speicher befindliche. Das Prinzip läßt sich mit dem eines „EXEC“-Files unter DOS 3.3 vergleichen; MERGE-Dateien können jedoch nur Programmzeilen enthalten. Direkt-Anweisungen sind unzulässig und führen zu einer Fehlermeldung.

**CHAIN "NAME":** Im Zusammenhang mit der COMMON-Anweisung können Variablen an das aufgerufene Programm „NAME“ übergeben werden.

```
10 REM Beispiel 3
20 COMMON A,B()
30 A = 1: B(2) = 9
40 CHAIN "TEIL2"
```

In obigem Beispiel könnte das Programm „TEIL2“ sofort auf die Variablen A und B(2) zugreifen. Der CHAIN-Befehl erlaubt auch das Nachladen einzelner Routinenblöcke. Seine volle Syntax ist CHAIN /MERGE/FILENAME\$, STARTZEILE/, ALL/, DELETE VON - BIS; die Anweisung „CHAIN MERGE "TEIL2.BAS", 1000, ALL, DELETE 100-400“ beispielsweise bewirkt nach der Löschung der Programmzeilen 100-400 das „Dazuladen“ des Programmes „TEIL2.BAS“ (s.a. MERGE); die Programmausführung wird in Zeile 1000 fortgesetzt; alle Variablen stehen weiterhin zur Verfügung.

### Weitere Befehle zur Dateienverwaltung

Bei diesen Anweisungen muß bei File-Namen der Typ mitangegeben werden. Fehlt er, wird nicht automatisch „BAS“ hinzugefügt; KILL "DEMO" und KILL "DEMO.BAS" sind also nicht equivalent!

**KILL "NAME.BAS":** Löschen der Datei „NAME.BAS“.

**NAME "NAME.BAS" AS "NEUNAME.BAS":** Umbenennen einer Datei.

**FILES:** Anzeige des Inhaltsverzeichnisses der Disk im Bezugslaufwerk. Dies ist, wenn MBASIC, wie oben angegeben, von der Masterdisk gestartet wurde, Drive A: . Die Inhalte anderer Disketten können ebenfalls ausgegeben werden, z.B. mit FILES "B:\*.\*" für Drive B:.

**RESET:** Dieser Befehl muß immer dann eingegeben oder vom Programm ausgeführt werden, wenn eine Diskette gewechselt wurde, um die neue Disk ordnungsgemäß im CP/M-Betriebssystem „anzumelden“. *Vergeßlichkeit in diesem Punkt kann beim Versuch, auf eine nicht angemeldete Diskette zu schreiben, zum Absturz des*

*MBASIC-Interpreters und damit zum Verlust des gesamten Programmes führen!*

## 15. Applespezifische Erweiterungen

Alle bis jetzt besprochenen Anweisungen gehören – wenn nicht besonders darauf hingewiesen wurde – zum Standard-MBASIC und sind daher auch auf vielen anderen Rechnern verfügbar. Die folgenden Befehle jedoch sind i.d.R. nur in der Appleversion des MBASIC vorhanden. Bei einigen von ihnen – nämlich den Grafikbefehlen – treten bei Verwendung von 80-Zeichenkarten Schwierigkeiten auf, da CP/M grundsätzlich eine derartige Zusatzkarte – falls vorhanden – zur Ausgabe benutzt. Damit ist es i.d.R. nicht möglich, Grafik und vier Zeilen Text darzustellen. Dagegen arbeiten z.B. VTAB, HTAB, POS und HOME auch bei 80 Zeichen pro Zeile einwandfrei, wobei das natürlich nicht unbedingt für jede 80-Zeichenkarte gilt.

### 15.1. Ein- und Ausgabebefehle

Wie im Applesoft funktionieren NORMAL, INVERSE, HTAB und VTAB.

**Y = VPOS(0):** Mit der VPOS(0)-Funktion kann die vertikale Cursorposition ermittelt werden.

**WIDTH BREITE,HOEHE:** In einer zweiten Form der WIDTH-Anweisung (s.o.) ist es möglich, neben der Ausgabebreite auch die Bildschirmhöhe festzusetzen.

Zur Paddleabfrage steht neben der PDL-Funktion auch eine Möglichkeit zur Verfügung, den Druckknopf abzufragen:

**Y = BUTTON(Paddlenummer):** Diese Funktion ergibt -1, wenn der entsprechende Knopf gedrückt wurde. Beispiel: IF BUTTON(0) THEN ...

**BEEP HOEHE, LAENGE:** Diese neue Anweisung dient zur Erzeugung unterschiedlichster Töne. HOEHE und LAENGE sind dabei Integerausdrücke im Bereich von 0 bis 255.

### 15.2. Kommandos zur Ansteuerung der niedrigauflösenden Grafik

Die bekannten Befehle COLOR=, PLOT, HLIN, VLIN, GR und TEXT sowie die SCRNM-Funktion stehen auch hier zur Verfügung. Der GR-Befehl hat eine zusätzliche erweiterte Form bekommen:

**GR MODUS, FARBE:** MODUS = 0 bewirkt die Ausgabe von Grafik und vier Zeilen Text, bei MODUS = 1 wird der gesamte Schirm für die Grafik verwendet. Je nach dem Wert von FARBE wird der Bild-

schirm gelöscht (FARBE = 0) oder mit der entsprechenden Farbe gefüllt, wobei die Farbuordnung der vom Applesoft entspricht. Beispiel: GR 1,3.

### 15.3. Hochauflösende Grafik

*Hires-Grafik ist nur mit dem GBASIC-Interpreter möglich, der sich ebenfalls auf der CP/M-Masterdisk befindet und von CP/M aus mit „GBASIC <Return>“ gestartet wird. Alle anderen MBASIC-Befehle sind auch im GBASIC gültig. (Anm. d. Red.: Das GBASIC, das in Verbindung mit der Premium Card für den Ile geliefert wird, ist MBASIC und GBASIC in einem. Ein Testbericht zu dieser Karte erscheint in Kürze.)*

Wie im Applesoft funktionieren HCOLOR= und HPLOT sowie HGR.

**HGR m,c:** Dieses erweiterte Kommando funktioniert ähnlich wie die GR-Anweisung; die Bedeutung der Parameter m und c wird in der Referenztabelle Abschnitt Q erläutert.

**Z = HSCRN(X,Y):** Diese neue Funktion ergibt -1 (logisch wahr), falls der entsprechende Punkt (X,Y) der Hires-Grafik gesetzt ist. Beispiel: IF HSCRN(120,99) THEN ...

Shaperoutinen sind im GBASIC nicht vorhanden.

## 16. Literatur für Interessierte

Eine vollständige, alles umfassende Erläuterung eines BASIC-Interpreters ist im Rahmen eines Zeitschriftenartikels natürlich nicht möglich. Zur Vertiefung des Wissens können folgende Bücher dienen, die sich allerdings nur auf MBASIC allgemein beziehen und nicht auf die applespezifischen Erweiterungen eingehen.

(1) Günther Daubach, MICROSOFT BASIC 80, IWT-Verlag. Ein sehr ausführliches Buch, das auf über 300 Seiten neben gut verständlichen Beispielen für den Anfänger fundierte Informationen für den Profi bietet.

(2) J.J. Purdum, BASIC-80 und CP/M. Eine Einführung in MBASIC in Kursform mit vielen Aufgaben.

(3) Thom Hogan, CP/M-Anwenderhandbuch, McGraw-Hill. Eine umfassende Beschreibung der CP/M-Befehle und Dienstprogramme.



## Microsoft BASIC-80 (MBASIC) – Referenztafel

Diese Tafel wurde in ihrer Gliederung der Applesoft-Referenzkarte angepaßt, um einen schnellen Vergleich zu ermöglichen.

Es werden folgende Abkürzungen verwendet :

**wAS** : wie im Applesoft-BASIC, höchstens geringfügige Abweichungen. Zu beachten ist aber, daß im MBASIC Befehle meistens von Leerzeichen eingeschlossen werden müssen und bei Funktionen sich die "(" unmittelbar anschließen sollte.  
Bsp.: 100 GOTO 2000 statt : 100GOTO2000

**AMB** : Befehl oder Funktion ist nur in der Appleversion des MBASIC verfügbar.

### A. Einfache Variablen

Typ	Beispiel	DEF-Fkt.	Konvertierungsfkt.
Integer -32768/+32767	ABC%	DEFINT	CINT
Real, 7 Stellen Genauigkeit	ABC/ABC!	DEFSNG	CSNG
Real, 17 Stellen Genauigkeit	ABC#	DEFDBL	CDBL
String	ABC\$	DEFSTR	—

Variablenamen können bis zu 40 Zeichen besitzen, von denen alle signifikant sind.

Die DEF-Funktionen ordnen Variablen mit bestimmten Anfangsbuchstaben global einem Typ zu, z.B.: DEFSTR A-C erklärt alle Variablen mit Anfangsbuchstaben A-C, die kein Suffix(!, #, %) tragen, zu Stringvariablen.

Die Konvertierungsfunktionen führen Typenumwandlungen durch (z.B. ? CINT(A#)), die aber bei Wertzuweisungen automatisch erfolgen. Bsp.: A% = C# ist gleichwertig mit A% = CINT(C#).

### B. Konstanten

132.35, 337E12	: einfach genaue Realkonstanten
23.56#, 321.4D13	: doppelt genaue Realkonstanten
&HFF3D, &H4F	: hexadezimale Integerkonstanten
&2300, &043	: oktale Integerkonstanten, Bsp.: A% = &03259

### C. Algebraische Funktionen

Alle Funktionen des Applesoft, zusätzlich

**\** : ganzzahlige Integerdivision, Bsp.: 5 \ 2 = 2  
**MOD** : A MOD B ergibt ganzzahligen Rest der Division, Bsp.: 10 MOD 3 ergibt 1.

### D. Vergleichende und logische Operationen

Operationen wie im Applesoft, zusätzlich Funktionen zur logischen Verknüpfung von Integerzahlen auf Bitebene:

A AND B	: logisches "UND"
A EQV B	: negiertes "Exklusiv-ODER" (logische Äquivalenz)
A OR B	: logisches "ODER"
A XOR B	: logisches "Exklusiv-ODER"
NOT(B)	: logisches "NICHT"

Vergleichsausdrücke ergeben den Wert -1, falls sie wahr sind, andernfalls den Wert 0. Bsp.: ? 3=3 ergibt -1!

### E. Systemkommandos

**wAS** : CLEAR, CONT, END, NEW, RUN, STOP, TRACE(AMB), NOTRACE(AMB), <Ctrl-C>

**CLEAR,h,s** : wie CLEAR, zusätzlich wird die Speichergrenze für MBASIC auf h sowie die BASIC-Stacklänge auf s (normal 256) gesetzt.  
**?FRE(0)** : Zeigt freien Speicherplatz in Bytes an. Garbage Collection erfolgt nur bei Aufruf in der Form X=FRE(" ").  
**<RESET>** : verursacht einen RESET-ERROR, der mit ONERR abgefangen werden kann.

### F. Befehle zum Arbeiten auf Maschinensprachenebene

**wAS** : PEEK, POKE, WAIT

**CALL X%(Par)** : Ruft Z80-Assemblerroutine auf und übergibt Parameter.  
**CALL% X%(A,X,Y)** : Ruft 6502-Unterprogramm auf und übergibt Werte an Akkumulator, X- und Y-Register, AMB.

Hinweis: Die jeweilige Startadresse der Maschinensprachenroutine (X%) muß eine Variable sein. Ausdrücke wie CALL &HFEDF sind unzulässig.

**Y = USRn(X)** : Aufruf der Maschinenroutine n (n = 0-9). Die Startadresse des Unterprogrammes muß vorher mit einer Anweisung der Form DEF USRn = Startadresse festgelegt worden sein.  
**Y = VARPTR(var)** : Ergibt die Adresse einer Variablen im Speicher.  
Bsp.: A = 10 : ? VARPTR(A)

### G. Kommandos zum Editieren / Verändern von Programmen

**wAS** : LIST, DEL(AMB)

**LLIST** : wie LIST, aber Ausgabe auf dem Drucker.  
**RENUM n,a,x** : Neunummerierung aller Zeilen im Abstand x, beginnend bei alter Zeile a, die die neue Zeilennummer n erhält. Bsp.: RENUM 10,1,5; RENUM ,,20  
**AUTO s,a** : Erzeugt in jeder Eingabezeile eine Programmzeilennummer, beginnend bei s im Abstand a. Bsp.: AUTO 1000,10; AUTO ,20. AUTO wird durch <Ctrl-C> abgeschaltet.  
**EDIT n** : Programmzeile n in den Editor bringen.  
**EDIT** : Zuletzt gelistete oder eingegebene Zeile editieren.

### Die wichtigsten Kommandos des Editors:

**<SPACE>** : Cursor nach rechts, überfahrene Zeichen werden sichtbar.  
**<-** : Cursor nach links  
**I** : Text ab Cursor einfügen. Der Insert-Modus wird durch <ESC> oder <RETURN> beendet.  
**X** : Zum Erweitern einer Zeile: Der Cursor springt zum Ende der Zeile, der Insert-Modus wird aufgerufen (s.o.).  
**nD** : n Zeichen rechts vom Cursor werden gelöscht. Die Löschung wird durch Ausdruck der entsprechenden Zeichen in eckigen Klammern dargestellt!  
**H** : Rest der Zeile ab Cursor löschen, Insert-Modus wird danach automatisch eingeschaltet.  
**nC** : Die nächsten n Zeichen werden ausgetauscht.  
**<RETURN>** : Editor verlassen, gesamte Zeile ausgeben.  
**Q** : Editor ohne Wirkung verlassen.  
**L** : Gesamte Zeile sichtbar machen und Editieren fortsetzen.

### H. Wichtige Control-Zeichen

**<Ctrl-A>** : Editor mit gerade eingegebener Zeile aufrufen.  
**<Ctrl-B>** : Backslash (\, auf deutschen Tastaturen Ö).  
**<Ctrl-S>** : Hält Programm an.  
**<Ctrl-Q>** : Programm nach <Ctrl-S> weiter fortsetzen.  
**<Ctrl-X>** : Eingabezeile löschen.

### I. Befehle zur Cursorsteuerung

**wAS** : HOME, HTAB(AMB), INVERSE(AMB), NORMAL(AMB), POS, SPC, VTAB(AMB)

**TAB(X)** : bewegt den Cursor in die Spalte X. Falls X<POS(0), wird der Cursor in der nächsten Zeile in die Spalte X gesetzt.

TAB darf nur in PRINT-Anweisungen verwendet werden!

**VPOS(0)** : Gibt die Zeile an, in der der Cursor steht.  
**WIDTHS,z** : Setzt die Anzahl der Spalten zur Bildschirmausgabe auf s und die Anzahl der Zeilen auf z; AMB

### J. Felder

**DIM A(a,b,c)** : Dimensioniert ein REAL-Feld, untere Feldgrenze ist A(0,0,0) bei OPTION BASE 0 oder A(1,1,1) bei OPTION BASE 1.  
**OPTION BASE n** : Setzt untere Feldgrenze auf n (s.DIM). Zulässig sind n = 0 (default) oder n = 1.  
**ERASE FELD** : Löscht den Array FELD, dieser kann dann neu dimensioniert werden.

### K. Zeichenketten

**wAS** : ASC, CHR\$, LEFT\$, LEN, MID\$, RIGHT\$, STR\$, VAL

**X\$ = HEX\$(n)** : Ergibt hexadezimale Darstellung von n, wobei n im Integerbereich liegen muß.  
**X = INSTR(n,X\$,Y\$)** : Überprüft beginnend bei Position n in X\$ diesen auf Übereinstimmung mit Y\$. Das Ergebnis ist gleich der ersten übereinstimmenden Position. Ist Y\$ nicht in X\$ erhalten, ergibt INSTR den Wert 0. Bsp.: INSTR("1234","23") ergibt 2.

MID\$(X\$,p,m) = Y\$ : Beginnend ab Position p werden die Zeichen in X\$ durch Y\$ ersetzt, es werden max. m Buchstaben ausgetauscht.  
X\$ = OCT\$(n) : Ergibt die oktale Darstellung von n, s.a. HEX\$.  
X\$ = SPACES\$(n) : Ergibt n Leerzeichen.  
X\$ = STRING\$(n,m) : Ergibt n Zeichen mit ASCII-Code m.  
Y\$ = STRING\$(n,X\$) : Ergibt n-mal das 1. Zeichen von X\$.

#### L. Ein- und Ausgabebefehle

wAS : GET(AMB), DATA, READ, RESTORE, PRINT.

INPUT "X:":X : Drückt "X:?" und wartet auf Eingabe.  
INPUT "X:":X : Gibt vor der Eingabe "X:" aus.  
INPUT;"X:":X : Unterdrückt Zeilenvorschub nach der Eingabe, d.h. Cursor bleibt nach <RETURN> in Eingabezeile.  
X\$ = INKEY\$ : Fragt Tastatur ab und übergibt, falls Taste gedrückt wird, das entsprechende Zeichen, sonst "".  
X\$ = INPUT\$(n) : Liest n Buchstaben, ohne sie auf dem Bildschirm darzustellen.  
LINE INPUT X\$ : Liest eine Zeichenkette bis zum <RETURN> ein und ermöglicht dabei die Eingabe von Kommata und Anführungszeichen im Text.  
RESTORE n : Setzt DATA-Eingabezeiger auf die DATA-Liste in Zeile n.  
PRINT USING : Ausgabe einer Variablenliste entsprechend einer Formatliste. Bsp.: PRINT USING "####.## DM";2 ergibt "###2.00 DM". Es erfolgt also eine Ausgabe der Formatliste, wobei gewisse Platzhalterzeichen durch Variablen der Liste ersetzt werden.

#### Die wichtigsten Platzhalterzeichen und ihre Bedeutung:

& : Ausgabe einer Zeichenkette beliebiger Länge.  
\ \ : Markierung eines Ausgabefeldes für Strings. Die Anzahl der ausgegebenen Zeichen ist gleich der der eingeschlossenen Leerzeichen +2.  
# : Markierung einer Ziffer im Ausgabefeld.  
. : Markierung eines Kommas in der Ausgabe. Bei ##.## werden von einer Variablen zwei Vor- und zwei Nachkommastellen ausgegeben.  
+ : Zeigt die Stelle an, an der das Vorzeichen einer Zahl ausgegeben werden soll: Bsp.: ##.##+  
- : Ein Minuszeichen wird hinter der Zahl ausgegeben, Pluszeichen werden nicht gedruckt. Bsp.: ##-  
\*\* : Alle Leerstellen im Zahlenausgabefeld werden mit "\*" gefüllt (s.o.); sog. "Scheckformat". Bsp.: ####.##  
↑↑↑ : Erzwingt Exponentialdarstellung einer Zahl. Bsp.: #↑↑↑.  
Reicht bei einer Ausgabe ein Zahlenfeld nicht aus, wird ein "%" als Warnzeichen ausgedruckt.

#### M. Drucker Ausgabe

LPRINT : wie PRINT, Ausgabe erfolgt auf dem Drucker. Ebenso LPRINT USING.  
X=LPOS(0) : Ergibt die aktuelle Position (Spalte) des Druckkopfes.  
WIDTH LPRINT n : Setzt die maximale Spaltenzahl für die Drucker Ausgabe auf n. Bsp.: WIDTH LPRINT 40

#### N. Befehle zur Programmablaufkontrolle

wAS : GOTO, GOSUB, ON...GOTO, ON...GOSUB, POP(AMB), RETURN.  
IF A=B THEN : wAS, jedoch kann für den Verneinungsfall (A ist nicht gleich B) die ELSE-Alternative benutzt werden: IF A=B THEN B=B-1 ELSE ?"ERROR".  
WHILE A=B : Die von WHILE/WEND eingeschlossenen Programmzeilen werden solange ausgeführt, wie A=B erfüllt ist.  
WEND  
FOR...NEXT : wAS, aber: Die Schleifenbedingung wird vor dem ersten Schleifendurchlauf getestet, d.h. eine Schleife FOR L = 3 TO 1 wird nicht durchlaufen, im Applesoft würde dagegen eine einmalige Ausführung erfolgen.

#### O. Anweisungen zur Fehlerbehandlung

ON ERROR GOTO n : Im Fehlerfall wird eine Fehlerbehandlungsroutine in Zeile n angesprungen.  
ON ERROR GOTO 0 : Normale Fehlerbehandlung (Ausgabe von Fehlermeldungen) wird wieder eingeschaltet.  
RESUME /RESUME 0 : Fehlerbehandlungsroutine verlassen und in die Zeile zurückkehren, in der der Fehler auftrat.  
RESUME NEXT : Wie RESUME, aber in der dem Fehler folgenden Zeile die Ausführung fortsetzen.  
RESUME n : Das Programm in Zeile n fortsetzen.  
X = ERR : Ergibt die Nummer des aufgetretenen Fehlers.  
X = ERL : Gibt die Zeile an, in der der Fehler auftrat.  
ERROR n : Erzeugt einen Fehler mit Fehlernummer n.

#### P. Lores-Grafik

wAS : COLOR=, HLINE, GR, PLOT, SCRIN, TEXT, VLINE.

GR n,c : Der Grafikbildschirm wird mit Farbe c (0-15) gelöscht; n=0 schaltet auf Grafik und 4 Textzeilen, n = 1 auf volle Grafikdarstellung. Bsp.: GR 1,12

#### Q. Hires-Grafik (nur mit GBASIC-Interpreter möglich sowie Standard bei Premium Card)

wAS : HGR, HCOLOR=, HPLLOT.

HGR m,c : Schaltet HGR-Modus m ein und löscht Schirm ggf. mit Farbe c. Bsp.: HGR 1,3

m	Modus	Schirmlöschung
0	280 * 160 Pkte., 4 Textzeilen	ja, Farbe c
1	280 * 192 Pkte.	ja, Farbe c
2	280 * 160 Pkte., 4 Textzeilen	nein
3	280 * 192 Pkte.	nein

Y=HSCRIN(X,Y) : Ergibt -1, falls Punkt auf Schirm gesetzt. Bsp.: IF HSCRIN(120,120) THEN ?"CRASH !"

#### R. Gameport und Lautsprecher

wAS : PDL.

Y = BUTTON(p) : Fragt Druckknopf an Paddle p ab, ergibt -1, falls Knopf gedrückt, sonst 0. Bsp.: IF BUTTON(0) THEN  
BEEP h,l : Erzeugt einen Ton der Höhe h und Länge l, h und l sind Integerwerte im Bereich 0-255.

\* Alle Befehle, die Grafik und Gameport betreffen, gehören nicht zum Standardsprachumfang des MBASIC.

#### S. Mathematische Funktionen

wAS : ABS, ATN, COS, EXP, INT, LOG, SGN, SIN, SQR, TAN.

Y = FIX(X) : Die Nachkommastellen von X werden abgeschnitten.  
Y = RND(n) : wAS, aber: Jeder Programmablauf erzeugt dieselben Zufallszahlen, wenn nicht mit RANDOMIZE ein neuer Startwert erzeugt wurde.  
RANDOMIZE A% : Erzeugung einer Zufallszahlenreihe in Abhängigkeit von der Integerzahl A%.  
RANDOMIZE : Programm hält an und fordert zur Eingabe eines Wertes auf. ("Random number seed ...")  
SWAP A,B : Der Inhalt der Variablen A und B wird ausgetauscht.  
DEF FNA(X,Y)=X\*Y : Definiert die Funktion A mit den Parametern X und Y. Aufruf mit z.B. ? FNA(2,3) ergibt 6.

#### T. Befehle zur Programmdateiverwaltung

Dateienbefehle werden wie normale BASIC-Befehle gehandhabt. Bsp.: 120 RUN "TEIL2.BAS", dagegen ist 120 ?CHR\$(4);"RUN TEIL2.BAS" falsch!

NAME\$ ist ein Stringausdruck (Konstante oder Variable), der einen File-Namen entsprechend CP/M (fname.ftyp) darstellt. Bsp.: "TEIL1.BAS", "TEST.TXT"

RUN NAME\$ : Programm laden und starten.  
LOAD NAME\$ : Programm laden.  
SAVE NAME\$ : Programm abspeichern.  
SAVE NAME\$,A : Programm als ASCII-Text abspeichern.  
SAVE NAME\$,P : Speichern und Listschutz setzen.  
MERGE NAME\$ : Programm über das im Speicher befindliche laden, NAME\$ muß ein ASCII-Programmfile sein.  
CHAIN NAME\$ : Programm laden und mit COMMON vereinbarte Variablen übergeben.  
COM: N var1,var2 : Festlegung der Variablen, die bei einem CHAIN-Aufruf an das nachgeladene Programm übergeben werden sollen. Bsp.: COMMON A,B(),C\$  
KILL NAME\$ : Löschen einer Datei.  
NAME ALT\$ AS NEU\$ : Umbenennen einer Datei. Bsp.: NAME "TEST.BAS" AS "PROBE.BAS"

Bei KILL und NAME muß unbedingt der Dateityp angegeben werden. Alle anderen Befehle nehmen ".BAS" als Typ an, wenn diese Angabe fehlt. Bsp.: RUN "TEST" ist equivalent mit RUN "TEST.BAS".

FILES : Ausgabe des Inhaltsverzeichnisses des Bezugslaufwerks.  
FILES "d:\*. \*" : Ausgabe des Inhaltsverzeichnisses der Disk d. Bsp.: FILES "B:\*. \*" für Slot 6, Drive 2.  
RESET : Neuansmeldung einer Diskette im CP/M Betriebssystem. Befehl muß nach jedem Diskettenwechsel eingegeben werden.  
SYSTEM : Verlassen des MBASIC und Rückkehr in das CP/M-Betriebssystem.

## Peeker-Sammeldisk #9

Diese Diskette enthält die Pascal-Programme aus den Heften 1-10/1985 im **Pascal-Format**. Einzelpreis 28,-; Fortsetzungspreis DM 20,-. Die Fortsetzungsbezieher werden benachrichtigt, da keine Pflichtabnahme für Pascal-Disketten.

PEEKER 9:			
DUPDIR.TEXT (1/85)	10	1-Dec-84	6
DUPDIR.CODE	4	1-Dec-84	16
GETDOS.TEXT	18	8-Dec-84	20
GETDOS.CODE	7	8-Dec-84	38
MOUSESTUFF.TEXT (4/85)	6	29-Jan-85	45
MOUSESTUFF.CODE	6	29-Jan-85	51
TESTMOUSE.TEXT	6	29-Jan-85	57
TESTMOUSE.CODE	2	29-Jan-85	63
DRAWMOUSE.TEXT	22	29-Jan-85	65
DRAWMOUSE.CODE	5	29-Jan-85	87
MOUSE.ASS.TEXT	20	29-Jan-85	92
MOUSE.ASS.CODE	6	29-Jan-85	112
MOUSE.LIBRARY	7	29-Jan-85	118
TRANSCEND.TEXT (5/85)	20	3-Dec-84	125
TRANSCEND.CODE	7	3-Dec-84	145
RAMDISK94.TEXT (6/85)	4	8-Feb-85	152
INIT.TEXT	14	8-Feb-85	156
INSTALL.TEXT	4	8-Feb-85	170
RAMDISK94.CODE	3	8-Feb-85	174
IDSEARCH.TEXT (8/85)	10	28-Nov-84	177
IDSEARCH.CODE	3	28-Nov-84	187
ALLG.TEXT (9/85)	8	21-Jul-85	242
ALLG.CODE	5	27-Jul-85	250
MUEL.TEXT	8	12-Jul-85	255
MUEL.CODE	6	27-Jul-85	263
CRUNCH.TEXT (10/85)	4	13-Apr-85	190
CRUNCH.CODE	3	13-Apr-85	194
MOVER.TEXT	14	13-Apr-85	197
MOVER.CODE	3	13-Apr-85	211
CRUNCHER.TEXT	24	13-Apr-85	214
CRUNCHER.CODE	4	13-Apr-85	238

**Der nächste Peeker  
Heft 10/1985  
erscheint am  
23. 9. 1985**

## Peeker-

Einzelbezug DM 28,-  
Fortsetzungsbezug DM 20,-  
Pascal- und CP/M-Disketten können vom Fortsetzungsbezug ausgeschlossen werden.

(Jederzeit kündbar, jedoch mindestens 6 Disketten)

(\* = nur auf Diskette, nicht im Peeker gelistet! Seitenangaben beziehen sich auf Beginn des Listings)

Hüthig Software Service  
Postfach 10 28 69 · 6900 Heidelberg 1

### Disk #1

(Heft 1+2, 1984, DOS-Format)

T.DISASSEMBLER.65C02 (1/84, S. 15)  
DISASSEMBLER.65C02

T.ACCEL.WAIT (1/84, S. 22)

ACCEL.WAIT  
T.ACCEL.BOOT  
ACCEL.BOOT  
ACCEL.LC.KOPIERER  
T.ACCEL.LC.KOPIE  
ACCEL.LC.KOPIE  
T.ACCEL.ROM.KOPIE1  
ACCEL.ROM.KOPIE1  
T.ACCEL.ROM.KOPIE2  
ACCEL.ROM.KOPIE2

TURTLE.GRAFIK.MIT.REMS (1/84, S.29)  
TURTLE.GRAFIK.OHNE.REMS \*

DOUBLE.LORES.SOFTSWITCH.DEMO (1/84, S. 37)  
DOUBLE.LORES.APPLESOFT.DEMO  
AMPER.DOUBLE.LORES.DEMO  
T.AMPER.DOUBLE.LORES  
AMPER.DOUBLE.LORES  
T.DOUBLE.LORES  
DOUBLE.LORES

HIRES (1/84, S. 41)  
T.PRINTHIRES  
PRINTHIRES

DHGR.APSOFT.DEMO (2/84, S. 30)  
AMPER.DOUBLE.HIRES.BAS  
AMPER.DOUBLE.HIRES  
T.AMPER.DOUBLE.HIRES  
DHGR.LINEPLOTTER

INSTRING.TEST (2/84, S. 43)  
INSTRING.OBJ  
T.INSTRING.OBJ  
INSTRING.LISA.SOURCE

LOESCHEN.EINES.ARRAYS (2/84, S. 52)

ULTRATERM.ENGLISCH \* (2/84, S. 60)  
ULTRATERM.DEUTSCH \*

PRIMZAHLEN.OVERMEYER \* (2/84, S. 70)  
PRIM.OBJ0 \*  
PRIM.OBJ1 \*

PRIM.TEST \*  
PRIM.TOOLKIT.SOURCE \*

### Disk #2

(Heft 1-2, 1985, DOS-Format)

T.RAMDISKLC (1-2/85, S. 14)  
RAMDISKLC

T.IBS.RAMDISKDRIVER (1-2/85, S. 20)  
IBS.RAMDISKDRIVER

T.AP20.RAMDISKTEST  
AP20.RAMDISKTEST

T.QUICKCOPY (1-2/85, S. 26)  
QUICKCOPY  
QUICKCOPY.PUFFER  
PRODOS.COPYA  
T.PRODOS.COPYOBJ \*  
PRODOS.COPYOBJ

PRODOS.PATCH (1-2/85, S. 31)

T.APPLESOFT.FRE (1-2/85, S. 36)  
T.LC.FRE  
LC.FRE  
FRE.TEST  
T.RAM.FRE \*  
RAM.FRE

T.SCHIRMDISK (1-2/85, S. 44)  
SCHIRMDISK.LISA.SOURCE  
SCHIRMDISK

T.VIDEXT  
VIDEXT.LISA.SOURCE  
VIDEXT

GETPAS (1-2/85, S. 70)  
T.GETPAS.ASS \*  
GETPAS.ASS  
GETDOS.PASCAL.SOURCE  
COPYDUPDIR.PASCAL.SOURCE

PRODOS.EDITOR\_MACROS (1-2/85, S. 86)

### Disk #3

(Heft 1-2, 1985, CP/M-Format)

STEUER.84 (1-2/85, S. 47)  
PASS.BAS  
MENUE.BAS  
HELP.BAS \*

A.BAS  
B.BAS  
C.BAS  
D.BAS  
E.BAS  
F.BAS  
G.BAS  
H.BAS  
I.BAS



# Sammeldisketten

J.BAS K.BAS L.BAS M.BAS N.BAS	SCREEN80.SAVER (4/85, S. 76)	PAGE.SWAP T.PAGE.SWAP	FETT * FETT.INVERSE *
<b>Disk #4</b> (Heft 3+4, 1985, DOS-Format)	<b>Disk #5</b> (Heft 5, 1985, DOS-Format)	WANDERNDER.STRICH (6/85, S. 16) KOMPRESSOR.DEMO KREIS.1 KREIS.2 KREIS.3 FLIPPER T.FLIPPER KOMPRESSOR T.KOMPRESSOR	PASTOPRO.1D (7/85, S. 62) * PASTOPRO.2D T.PASTOPRO.O PASTOPRO.O
TESTGENERATOR (3/85, S. 26) SAETZE BAHNFAHRT * ZU * TUN.UND.SOLLEN * IRGEND *	T.FM.BSP (5/85, S. 9) FM.BSP	OLYMPIA (6/85, S. 34) T.OLYMPIA	T.CONVERT (7/85, S. 69) CONVERT
MULTIPRECISION (3/85, S. 32)	T.SLOTTRAMDISK (5/85, S. 13) SLOTTRAMDISK SLOTTRAMDISK.HELLO	FOURIER.MAIN (6/85, S. 38) FOURIER.SYN FOURIER.SPEC	T.VORLESER (7/85, S. 71) VORLESER
T.WS.TRANSFER (3/85, S. 36) WS.TRANSFER T.WS.TRANSFER.2 * WS.TRANSFER.2 * GETCPM	PLOT.2.0 (5/85, S. 20) T.PLOT.B PLOT.B PLOT.PROTECTOR	AS.ERWEITERUNG (6/85, S. 43) T.AS.ERWEITERUNG AS.ERW.PROSTART AS.ERW.PRO * T.AS.ERW.PRO *	<b>Disk #8</b> (Heft 8, 1985, DOS-Format)
PRIM.0.SC.SOURCE (3/85, S. 62) PRIM.0.BIN	T.CONVERT560 (5/85, S. 26) CONVERT560 CONVERT560.DEMO	INSTALL.PASCAL.SOURCE (6/85, S. 48) RAMDISK94.PASCAL.SOURCE INIT.PASCAL.SOURCE	HELLO * ASMDIV *
PRIM.1.SC.SOURCE PRIM.1.BIN PRIM.FP	T.EDA (5/85, S. 33) EDA	RAMDISK.INIT.DOS (6/85, S. 55) AUXDRIVER T.AUXDRIVER MOVEDRIVER T.MOVEDRIVER RAMDISK.FORMATTER T.RAMDISK.FORMATTER	DISKTEST (8/85, S. 14) DISKTEST.START T.DISKTEST
ACCELERATOR.ABSTELLEN (3/85, S. 66)	TRANSCEND.PASCAL.SOURCE (5/85, S. 36)	SOLITAIRE.START (6/85, S. 64) SOLITAIRE SOLITAIRE.B T.SOLITAIRE.B	KOPY (8/85, S. 22) BATCHKOPY T.GETSETINFO GETSETINFO BILDTEST
T.WILDCARD.TEST * (3/85, S. 72) WILDCARD.TEST1 * T.WILDCARD.TEST2 * WILDCARD.TEST2 *	T.BLOCKTRACER (5/85, S. 51) BLOCKTRACER T.BLOCKTRACER1 BLOCKTRACER1	<b>Disk #7</b> (Heft 7, 1985, DOS-Format)	T.BOX.COPY (8/85, S. 26) BOX.COPY T.HSCRN HSCRN GRAF.QUATTRO.1
XPLOT.DEMO (4/85, S. 18) XPLOT.ROUTINE T.XPLOT.ROUTINE	FORMAT.LC (5/85, S. 56) FORMAT.LC.START	PYRAMID.PITTY (7/85, S. 6) * T.PYR.PITTY.0 * T.PYR.PITTY.1 * PYR.PITTY.0 * PYR.PITTY.1 * PYR.PITTY.BACK * PYR.PITTY.SHAPE *	T.DOUBLE.LORES (8/85, S. 34) DOUBLE.LORES DOUBLE.LORES.DEMO
MENUE.GENERATOR (4/85, S. 22)	T.DISKDRIVER.DEMO DISKDRIVER.DEMO	T.MEGAWARP.REL (7/85, S. 8) * MEGAWARP.REL * T.MEGAWARP.9900 MEGAWARP.9900 T.SPEEDTEST SPEEDTEST	START.CMD (8/85, S. 40) HMENUE.CMD * AUFNAHME.CMD * AUFMASKE.CMD * AUSGABE.CMD * SUCH.CMD * EDITFNAME.CMD * SUCHVNAME.CMD * SUCHBEME.CMD * SCHREIBA.CMD * SCHREIBL.CMD * LOESCH.CMD *
T.MACROS.65C02 (4/85, S. 31)	RANDOM.DEMO (5/85, S. 69) COLUMN80.DEMO	FORMAT (7/85, S. 20) T.FORMAT.OBJ FORMAT.OBJ	IDSEARCH.PASCAL.SOURCE (8/85, S. 49)
TERMINAL (4/85, S.36) TERMINAL.B T.TERMINAL.B	SUPERDUMP.EPSON (6/85, S. 22!) SUPERDUMP.IMAGEWRITER SUPERDUMP.BILD T.SUPERDUMP * SUPERDUMP EPSON IMAGEWRITER	BITEDITOR (7/85, S. 29) NORMAL *	FAKULTAET.DEMO (8/85, S. 57) T.FAKULTAET FAKULTAET
CAT.ARRAY (4/85, S. 44) CAT.SAVER EINTRAG.SUCHER EINTRAG.ANALYSE PRODOS.READER T.PRODOS.READER.OBJ PRODOS.READER.OBJ	<b>Disk #6</b> (Heft 6, 1985, DOS-Format)		GRAFIK.DEMOS (8/85, S. 68)
MOUSESTUFF.PASCAL.SOURCE (4/85, S. 51) MOUSE.ASS.PASCAL.SOURCE TESTMOUSE.PASCAL.SOURCE DRAWMOUSE.PASCAL.SOURCE	HELLO (6/85, S. 72) * ASMDIV *		ZEICHENJAGD (8/85, S. 70)
INALL.DATA (4/85, S. 70) SCREEN80.DATA (4/85, S. 33)	CURSOR1 (6/85, S. 6) T.CURSOR1 CURSOR2 T.CURSOR2 LINIE T.LINIE VIERECK T.VIERECK BOX T.BOX HINTERGRUND T.HINTERGRUND		T.RAM.FRE.NEU (8/85, S. 70) * RAM.FRE.NEU

# Pascal-Preisausschreiben

Um es gleich vorweg zu sagen: Das Pascal-Preisausschreiben aus Heft 7/85 war „ein Schuß in den Ofen“. Als das erste Peeker-Heft im September 1984 erschien, wurde ein Primzahlen-Wettbewerb ausgeschrieben, an dem sich über 270 Leser beteiligten, und das, obgleich der Peeker damals naturgemäß noch kaum bekannt war. Demgegenüber gingen jetzt, nachdem sich der Peeker mittlerweile zu einer bekannten und weitverbreiteten Apple-Zeitschrift gemausert hat, insgesamt nur ein knappes Dutzend Lösungen ein, von denen nur ganze zwei richtig waren. Läßt sich daraus schließen, daß es unter den Apple-Programmierern nur ganz wenige Pascal-Programmierer gibt und daß es unter den Pascal-Programmierern nur ganz wenige gibt, die wirklich schwierige Programme schreiben können? Ich kann es kaum glauben, aber es muß wohl so sein. Zwar wird Pascal an Schulen gelehrt, doch scheint man dort nur selten über die Grundlagen hinauszukommen. Und Hobbyisten sind wohl eher geneigt, sich mit Applesoft und Assembler zu befassen, zumal das Pascal-Betriebssystem erst bei einem voll ausgebauten Apple-System interessant wird. Außerdem kostet das Pascal-Betriebssystem knapp DM 1000,-, während der Applesoft-Interpreter gratis im ROM mitgeliefert wird. Für mich stellt sich natürlich jetzt die Frage, ob ich in Zukunft mehr oder weniger Pascal-Themen behandeln soll. Einerseits müßte man weniger Pascal-Programme bringen, weil das Interesse an Pascal zu gering ist. Andererseits könnte man mehr Pascal-Themen behandeln, weil ein Teil der Apple-Besitzer noch keinen rechten Zugang zu Pascal gefunden hat, obgleich ein Interesse vorhanden ist. Wie dem auch sei, schreiben Sie mir, ob wir in Zukunft mehr oder weniger Pascal-Aufsätze bringen sollen. Verwenden Sie zu diesem Zweck die in diesem Heft eingeklebte Info-Karte, auf der Sie nur das Gewünschte oder Vorhandene anzukreuzen brauchen.

Zurück zum Wettbewerb. Der 1. Preis mit DM 500,- geht an Herrn

## Ulrich Allgeier

in Stuttgart. Sein Punktesaldo beträgt 4394. Den 2. Preis mit DM 250,- hat Herr

## Gil Müller

in Köln mit einem Punktesaldo von 5877 gewonnen. Die Programme beider Gewinner laufen unter Pascal 1.1 und 1.2, elimi-

nieren Ctrl-Zeichen und Bit-7-on-ASCII-Zeichen aus dem ProDOS-Textfile, finden jeden beliebigen Textfile-Eintrag im ProDOS-Volume-Directory (also nicht nur den 1. Eintrag des 1. Blocks), können bis zu 32K große Dateien verarbeiten, die auf der ProDOS-Diskette nicht als kontinuierliche Blocks abgelegt sind, und laufen auf jeder normalen 35-Spur-Pascal-Diskette. Bei allen anderen Lösungen war die eine oder andere Forderung nicht erfüllt. Einige „Lösungen“ liefern sogar nur auf ihrer eigenen Diskette; dies war nun doch wohl etwas zu plump. Übrigens mußte man Pascal 1.2

nicht kennen, um die Lösung zu erstellen, denn bei seriöser Programmierung wäre eine Pascal-1.1-Programm auch stets unter dem Pascal-1.2-Betriebssystem gelaufen. Von absoluten Systemadressen wie „Time“ durfte man dann jedoch nicht mehr ausgehen.

Auf der Peeker-Sammeldiskette #9 befinden sich die beiden Lösungen der zwei Gewinner unter den Dateinamen ALLG.TEXT und ALLG.CODE für die PROTOPAS-Version von Ulrich Allgeier sowie MUEL.TEXT und MUEL.CODE für die PROTOPAS-Version von Gil Müller. us

PROTOPAS-Quellcode (ALLG.TEXT)  
von Ulrich Allgeier  
(läßt sich normal compilieren)

```
TYPE B=INTEGER;Y=PACKED ARRAY[0..1]OF 0..255;VAR
I,L,H,N,M:B;F:FILE;PROCEDURE P(V:B);VAR T:RECORD CASE BOOLEAN OF
FALSE:(A:B);TRUE:(P:↑Y);END;BEGIN T.A:=I;T.P↑[0]:=V;I:=I+1;END;PROCEDURE
O;BEGIN
I:=513;P(104);P(133);P(18);P(104);P(133);P(19);P(104);P(133);P(0);P(104);P(133)
;P(1);P(104);P(104);P(160);P(1);P(177);P(0);P(153);P(6);P(0);P(185);P(0);P(0);P
(153);P(2);P(0);P(153);P(4);P(0);P(136);P(16);P(239);P(162);P(255);P(198);P(6);
P(208);P(4);P(198);P(7);P(48);P(89);P(230);P(2);P(208);P(2);P(230);P(3);P(160);
P(2);P(177);P(2);P(41);P(127);P(168);P(201);P(13);P(240);P(40);P(201);P(32);P(
144);P(227);P(138);P(48);P(37);P(152);P(73);P(32);P(240);P(12);P(202);P(48);P(
29);P(208);P(10);P(169);P(32);P(32);P(113);P(2);P(208);P(20);P(232);P(208);P(
204);P(169);P(16);P(32);P(113);P(2);P(138);P(105);P(32);P(32);P(113);P(2);P(208)
;P(4);P(162);P(0);P(240);P(2);P(162);P(255);P(152);P(32);P(113);P(2);P(208);P(
179);P(230);P(4);P(208);P(2);P(230);P(5);P(132);P(16);P(160);P(2);P(145);P(4);P
(164);P(16);P(96);P(169);P(0);P(32);P(113);P(2);P(165);P(1);P(69);P(5);P(41);P(
3);P(208);P(243);P(165);P(0);P(197);P(4);P(208);P(237);P(160);P(1);P(169);P(0);
P(145);P(0);P(136);P(165);P(5);P(229);P(1);P(74);P(145);P(0);P(165);P(19);P(72)
;P(165);P(18);P(72);P(96);I:=-320;P(0);P(2);END;PROCEDURE
R;VAR A:PACKED ARRAY[0..2047]OF 0..255;D:ARRAY[-1..16383]OF B;BEGIN REPEAT
UNITREAD(5,A,2048,2);I:=-16151;P(0);Writeln('PROTOPAS U.Allgeier');I:=0;FOR
N:=0 TO 3 DO FOR M:=0 TO 12 DO BEGIN H:=(A[N*512+M*39+4])MOD
16;IF(A[N*512+M*39+20]=4)AND(H>0)THEN BEGIN
(*$!-*)I:=I+1;D[I]:=N;D[I+99]:=M;WRITE(I,' ');FOR L:=1 TO H DO
WRITE(CHR(A[N*512+M*39+4+L]));Writeln;END;END;READLN(L);UNTIL L
IN[1..I];L:=D[L]*512+D[L+99]*39;H:=A[L+4]DIV
16;D[-1]:=(A[L+25]+A[L+26]*256);WRITE('*** START',CHR(7));CASE H
OF 1:UNITREAD(5,D[0],512,A[L+21]+A[L+22]*256);2:BEGIN
UNITREAD(5,A,512,A[L+21]+A[L+22]*256);H:=0;N:=A[H]+256*A[H+256];M:=N;I:=1;
REPEAT IF N+I=A[H+I]+256*A[H+I+256]THEN BEGIN I:=I+1;END ELSE BEGIN
UNITREAD(5,D[256*H],512*I,N);H:=H+1;I:=1;N:=A[H]+256*A[H+256];END;UNTIL
N=0;END;END;I:=-16151;P(0);UNITSTATUS(9,D,1);H:=BLOCKWRITE(F,D[0],D[-1],2)
;WRITE(CHR(7),'*** ENDE');CLOSE(F,LOCK);END;BEGIN
REWRITE(F,'TEMP.TEXT');O;R;END.
```

## HIB

### Der Computerladen in Nürnberg

Erst seit gut einem Jahr, seit dem 2. Mai 1984, befindet sich der HIB Computerladen in Nürnberg, in der Äußeren Bayreuther Straße 72. Entgegen der Meinung verschiedener Hardware-Hersteller hat

sich das Konzept mehr als bewährt: Man hat sich nicht auf einen Hardware-Hersteller fixiert und mit dieser Methode bereits im ersten Jahr einen Umsatz von mehr als 1,5 Mio. DM erreicht.



#### Personal Computer

Im Bereich Personal Computer werden die Rechner von Zenith, Z-150 und Z-160, und der Rainbow von Digital angeboten. Zu diesen Computern werden angemessenen leistungsfähige Drucker von Brother, Epson und NEC betriebsbereit vorgeführt. Farbmonitore von Taxan und IBM-kompatible Interface-Karten von AST werden ebenfalls im Betrieb gezeigt. Als autorisierter Fachhändler von „Microsoft“, „Lotus“ und der „SM-Software-AG“ konzentriert sich das Software-Angebot in diesem Bereich auf den Anwender im Büro.

#### Home-Computer

Das Home-Computer-Angebot reicht vom Commodore 64, der als Lern-Computer für Kinder oder Spielcomputer für Jugendliche und Erwachsene seine Käufer findet,

über den Schneider CPC464 bis zum Alphatronic PC, der meist als kostengünstiges Textverarbeitungssystem optimal eingesetzt wird.

Hier findet der Besucher auch ein umfangreiches Angebot an Fachliteratur und Software für den Heimbereich. Wie bei den Personal Computern werden auch für diese Rechner in Preis und Leistung der Anwendung angemessene Drucker, Monitore, Joysticks und immer die neuesten Peripherie-Geräte angeboten. Druckerpapier und Farbbänder für alle angebotenen Drucker sind ebenso vorrätig wie Disketten in jeder Qualität und Ausführung.

#### Apple-Computer

Bei Hobby-Anwendern genauso beliebt wie bei Büromenschen sind die Rechner Apple IIc, Apple IIe und der Macintosh.

Deshalb wurden sie auch im mittleren Teil des Ladens, neben und gegenüber der Kasse aufgestellt. Für den Macintosh werden an Software neben der gesamten Produktpalette der Firma Microsoft und „Jazz“ von Lotus hauptsächlich Spiel- und Grafikprogramme angeboten. Eine ganz besondere Stärke des HIB-Computerladens liegt in der Anpassung verschiedenster Geräte an alle gängigen Microcomputer. Neben Anschlußmöglichkeiten von z.B. Typenraddruckern am Macintosh wird seit neuestem ein serieller Drucker-Puffer angeboten, der für alle für den Macintosh erhältlichen Druckmodelle lieferbar ist.

Die Apple-IIc-Besitzer sind beeindruckt von der Qualität des TAXAN-Monitors, der über die TAXAN-RGB-Karte betrieben wird, und vom günstigen Preis des angeschlossenen Chinon-Laufwerks.

Das umfangreichste Angebot an Peripherie kann man für den Apple II+ und IIe vorweisen. Neben Farbmonitoren, RGB-, 80-Zeichenkarten, IC-Test-Karten und Interface-Karten für fast jede Anwendung sind die Kunden immer wieder von der Kapazität (640K formatiert) und einfachen Bedienung der angeschlossenen 5 1/4"- und 3 1/2"-Laufwerke begeistert. Da diese Rechnerreihe nach wie vor ausgesprochen vielfältig einsetzbar ist, bietet der HIB-Computerladen von der Hobby-Anwendung bis zum Einsatz als Meß- oder Steuergerät fast alles, was den Kunden interessiert. Als erstes Standard-Software-Produkt der Firma HIB befindet sich seit Februar 1985 ein Terminalprogramm, „HIB-Modem-Transfer“, auf dem Markt, das zusammen mit einem Anschlußkabel (vom Game-I/O des Apple II+ oder Apple IIe zur V.24-Schnittstelle eines Akustik-Kopplers) geliefert wird und für den Hobby-Anwender eine preiswerte Alternative zu herkömmlichen Kommunikationsmöglichkeiten bietet.

Neben dem Ladengeschäft ist auch der Versand, bedingt durch die intensive Anzeigenwerbung in Fachzeitschriften, zu einem wichtigen Vertriebszweig geworden. Sowohl Endanwender wie auch Händler im gesamten Bundesgebiet, Berlin, Österreich und der Schweiz werden schnell und zuverlässig beliefert.

Für die Zukunft ist geplant, die kommerziellen Anwender stärker als bisher anzusprechen. Dazu wird die Produktpalette nach oben hin abgerundet und das gesamte Computer-Programm für diesen Bereich in einer neuen Zweigstelle des HIB-Computerladens untergebracht.

## Erfahrungsbericht OPERATOR I

von Reiner Hammerschmidt

Die OPERATOR-Tastatur von AFC ist eine ergonomisch flache Tastatur, die zum Anschluß an verschiedene Rechner geeignet ist. Sie enthält neben dem Haupttastenfeld mit 62 Tasten 15 weitere Funktionstasten sowie rechts einen Zehnerblock. Alle Tasten – auch die programmierbaren – haben Autorepeat, d.h. wird eine Taste länger gedrückt, wird das entsprechende Zeichen (oder auch mehrere) automatisch wiederholt.

Der Vorteil dieser auf den ersten Blick nicht gerade billigen Tastatur (ca. 450,- DM) liegt in der Programmierbarkeit von 39 Tasten in je 2 Ebenen (normal und Shift). Zu den programmierbaren Tasten gehören die 15 Funktionstasten, der Zehnerblock und die 4 im Haupttastenfeld befindlichen Cursor-Tasten (!) sowie die Tasten DEL und TAB. Ein entscheidender Vorzug dieser Tastatur gegenüber anderen besteht in der Programmierbarkeit der Cursor-Tasten, so daß man nicht umlernen muß, wenn man z.B. einmal mit DOS und das andere Mal mit CP/M arbeitet, denn die Cursor-Steuerzeichen sind nicht überall die gleichen. Da aber jede Taste doppelt programmiert werden kann, ist es möglich, die eine Ebene für DOS und die andere für CP/M vorzusehen. Auf jeder Taste können beliebige Zeichenfolgen bis zu je 254 Zeichen abgelegt werden, allerdings zusammen nur maximal 2032 Zeichen. Dabei können alle Bytes von \$00 bis \$FF verwendet werden, auch wenn beim Apple hardwarebedingt nur die Zeichen bis \$7F erkannt werden, was aber keine Einschränkung darstellt, denn der ASCII-Code umfaßt ohnehin nur 7 Bits.

Mit dem Haupttastenfeld sind alle ASCII-Zeichen von \$00 bis \$7F mit Ausnahme von \$60 (einfaches Abführungszeichen) erzeugbar, was man sich jedoch – falls benötigt – auf eine programmierbare Taste legen kann. Ebenfalls in das Haupttastenfeld integriert ist die Reset-Taste, die nur zusammen mit der Control-Taste gedrückt ein Reset-Signal an den Prozessor abgibt.

Da die Tastatur universell gehalten ist, wird sie mit einem ca. 1m langen offenen Kabel geliefert (die

Tastatur kann sowohl als serielle wie auch, am Apple, als parallele Peripherie betrieben werden). Zum Betrieb muß man sich dann selbst um eine Steckverbindung bemühen, deren Anschluß aber, dank der mitgelieferten Dokumentation, nicht schwierig ist. Die Dokumentation selbst umfaßt ca. 20 Seiten und beschreibt Anschluß, Bedienung und Programmierung. Außerdem ist ein Schaltbild enthalten; aber leider kein Listing des Betriebsprogramms für die eingebaute CPU 6511. Beim Kauf der OPERATOR sind alle programmierbaren Tasten mit DOS- und CP/M-Befehlen für den Apple belegt. Dies läßt sich aber ändern (s.o.).

### Bedienung

Die Bedienung der Tastatur ist einfach, die Tasten sind griffig und auch die Lage der Cursortasten prägt sich schnell ein. Besonders gut gelungen ist die große RETURN-Taste. Als Nachteil muß man jedoch die verwirrende Doppelbeschriftung der Tasten Y und Z anführen. Wenn man beim Schreiben auf die Tasten sieht, muß man des öfteren Überlegen, ob die angepeilte Taste nun ein Z oder ein Y erzeugt, was ja mittels Lötbrücke umschaltbar ist (QWERTZ – QWERTY). Auch wenn einmal ein anderer am Rechner arbeitet, stellt sich jedesmal die Frage: Y oder Z? Dies hätte der Hersteller besser machen können durch normal beschriftete zusätzliche Tastenkappen, die dann bei Bedarf umgesteckt werden könnten. Eine Nachfrage bei AFC zu diesem Problem ergab, daß es keine normalen, nur mit Y bzw. Z beschriftete Tastenkappen gibt.

Die Programmierung der Tastatur geschieht durch Ablegen der entsprechenden ASCII-Zeichenfolgen in einem EPROM 2716, welches dann in einen einzulötenden IC-Sockel auf der Tastaturplatine einzusetzen ist. Da das Editieren der zu programmierenden Befehle nicht ganz einfach ist, wurde hierzu ein Programm entwickelt, das Interessenten beim Autor gegen Erstattung der Unkosten erhalten können. (Zuschriften werden von der Pecker-Redaktion weitergeleitet.)

### Fazit

Zusammenfassend kann nach 6monatigem Gebrauch gesagt werden, daß die OPERATOR ihr Geld wert ist. Es macht Spaß, an der Tastatur zu arbeiten, auch wenn es am Anfang schwerfällt,

unter den programmierten Tasten jeweils direkt diejenige mit dem gewünschten Befehl zu finden. Das flache Design und das prellfreie Arbeiten tun ein übriges. Bleibt nur der Nachteil der Doppelbeschriftung der Tasten Y und Z ...



Tastatur Operator IIe

## Die OPERATOR IIe

getestet von Harald Grumser

Mit der OPERATOR IIe kann nun auch der Apple IIe mit einer komfortableren Tastatur ausgestattet werden, die darüber hinaus einige weitere Vorzüge gegenüber der OPERATOR I aufweist.

Ein spezielles Interface für den (Original-)Apple IIe verbindet die Tastatur mit dem Rechner. Dabei werden außer der Reset- auch die beiden Apfeltasten als Hardware-Tasten unterstützt. Die Umschaltung zwischen deutschem und amerikanischem Zeichensatz erfolgt danach von der Tastatur aus (QWERTZ – QWERTY).

Der Einbau dieses Interfaces auf der Mutterplatine (wegen dieser Hardware-Tasten kann kein Slot benutzt werden) gestaltet sich etwas schwierig, da zwei ICs entnommen und an deren Stelle die Interface-Platine eingesetzt werden muß.

Da die OPERATOR I bereits besprochen wurde, soll hier nur auf die Änderungen eingegangen werden.

### Die Unterschiede zur OPERATOR I

Die neue OPERATOR entspricht im wesentlichen ihrer Vorgängerin. Um für die Apfeltasten Platz zu schaffen, wurde die Cursorsteuerung in das Feld der Funktionstasten verlegt, wodurch nur noch 13 zur Verfügung stehen.

Als Erweiterungen sind die Möglichkeiten eines Software-Schlusses (Password) und eines wählbaren akustischen Signals zu nennen.

Der wichtigste Unterschied dürfte in der freien und stets wiederholbaren Programmierung der Funktions-, Cursor-, Tab- und Delete-Tasten sowie des Arithmetikblocks (10er-Block mit Rechenzeichen) liegen, die in drei Ebenen (zwei Ebenen bei der OPERATOR I) erfolgen kann. Dadurch lassen sich über 100 eigene Funktionen (maximal 14 Zeichen pro Taste) realisieren. Diese Programmierung ist von der Tastatur aus sehr einfach zu handhaben und läßt sich jederzeit wiederholen, so daß auch im Betrieb „auf die Schnelle“ eine eigene Funktion festgelegt werden kann (als zusätzliche Erleichterung kann ein ASCII-Wert nun auch hexadezimal eingegeben werden). Nach dem Ausschalten des Rechners bleibt die Programmierung erhalten (EEPROM).

Als sehr vorteilhaft im täglichen Gebrauch erweist sich die Möglichkeit, die drei Ebenen (Unshift, Shift und Ctrl) zu vertauschen. Somit kann beim Betriebssystemwechsel auch die jeweilige erste Ebene gewählt werden; die gewünschte Funktion (z.B. CATALOG unter DOS 3.3, CAT unter ProDOS und DIR unter CP/M) wird somit

stets durch einfachen Tastendruck ausgelöst.

## OPERATOR als Schnittstelle

Die Operator I und II kann mit einem Barcode- oder Magnetkartenleser ausgestattet werden. Die Signale werden automatisch in die entsprechenden ASCII-Werte umgewandelt und von der Tastatur an den Rechner geschickt. Somit ist ohne Programmänderung eine einfache Umstellung auf eine alternative Eingabe möglich. Auch kann eine Maus oder ein Rollball angeschlossen werden, deren Bewegungen in wiederholte Cursorgänge verwandelt werden.

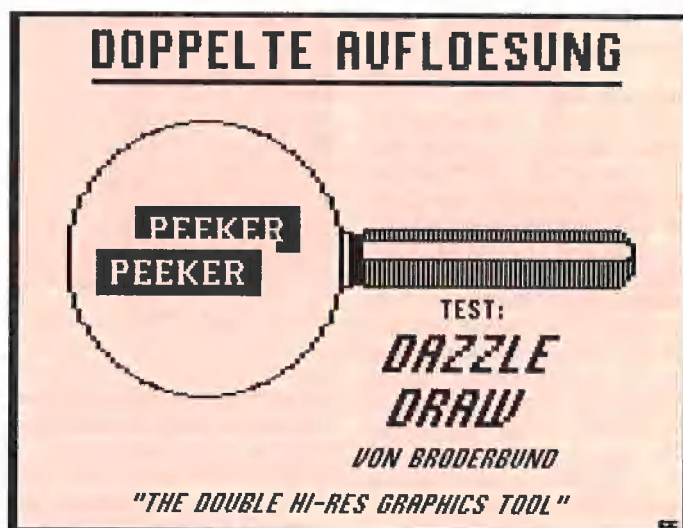
## Fazit

Das bereits zur OPERATOR I Gesagte gilt auch für die OPERATOR II. Als kleines Manko bleibt die Shift-Lock-Taste zu nennen, deren Zustand nicht abgelesen werden kann. Ansonsten gibt nicht nur der Preis von DM 755,- Anlaß dazu, dieses Produkt als einen Mercedes unter den Tastaturen zu bezeichnen.

*Anmerkung:* Diese Tastatur wird auch als OPERATOR II zu einem Preis von DM 695,- für Rechner mit seriellem oder parallelem Anschluß vertrieben.

## Dazzle Draw und Mousepaint

Erfahrungsbericht von Dieter Charchot



Ausdruck von Dazzle Draw

Dazzle-Draw ist ein Double-Hires-Zeichenprogramm für den Apple IIc bzw. IIe mit 64K-Karte, das von der Firma Broderbund produziert wird und unter ProDOS läuft.

Wenn Sie das Vergrößerungsglas aus der Grafik zu diesem Artikel ein wenig herumschwenkten, dann würden Sie ein kleines Nagetier sehen, das sich schamhaft in eine dunkle Ecke drückt. Es ist wahrscheinlich eine Apple-Maus, und der Grund für das Unbehagen des sonst recht großspurig daherkommenden Mäuschens dürfte das Apple-Programm **Mousepaint** sein, mit dem auch Apple-IIe-Besitzer in den Genuß eines von der Maus unterstützten Grafikprogramms kommen. Wenngleich von

den kreativ nutzbaren Möglichkeiten recht brauchbar, hat es jedoch einige unverständliche Mängel, die den Spaß an spielerischer Gestaltung relativ schnell und nachhaltig verderben.

So sieht man beim Erstellen der Grafiken leider nur einen Teil der Gesamtbildfläche. Wer alles sehen möchte, muß erst umständlich manipulieren. Ist man dann endlich fertig, speichert man stolz sein Meisterwerk auf Diskette. Leider muß man den Dateinamen auf einem kleinen Notizzettel vermerken, denn ein Catalog-Aufruf ist leider nicht möglich. Hat man das Zettelchen verlegt oder verloren, muß man das Programm verlassen, um auf der Diskette nach dem Rechten zu sehen.

Was hat dies alles mit der Lupe zu tun? Nun, die Mängel dieses „ge-mausten“ Programmes werden dann unübersehbar, wenn man es an einer Alternative messen kann, und diese Möglichkeit bietet **Dazzle Draw**, ein neues Grafikprogramm von Broderbund. Dieses Programm ist ein echtes Double-Hires-Programm mit einem hervorragenden Darstellungsvermögen. Neben vielen anderen interessanten Möglichkeiten kann der „Apple-Maler“ z.B. unter 16 Farben und 30 Mustern wählen, die darauf warten, Grafiken von verblüffender Wirkung zu zieren.

Zuvor bootet man die Systemdiskette, wählt per Menü als Eingabemöglichkeit Maus, Grafiktablett, Koala Pad oder Joystick und installiert schnell noch durch „Anklicken“ der richtigen Daten in einer auf Wunsch angebotenen Liste von Druckern und Interfacekarten seine Konfiguration – fertig!

Nun kann man sich entscheiden, ob man gleich in das Programm einsteigen oder zuvor noch eine Sicherheitskopie anfertigen möchte. Hier erlauben die Entwickler von Broderbund die unproblematische Anfertigung einer (und wirklich nur einer einzigen) Backup-Kopie des Programms zu Sicherheitszwecken.

Anschließend startet ein jungfräulicher Bildschirm den Apple-Grafiker an, nur unterbrochen von einer Befehlszeile am oberen (und gelegentlich zusätzlich am unteren) Rand. Hier findet man die einzelnen Sparten, die man mit dem Cursor ansteuern kann. Zur Ehrenrettung der kleinen Maus sei bestätigt, daß man damit den normalerweise benutzten kleinen Pfeil besser und exakter im Griff hat als mit dem Joystick. Aber auch hier macht Übung den Meister.

Hat man „getroffen“, bieten sich in Form der inzwischen wohl hinlänglich bekannten „Pull-Down-Menüs“ viele weitere Möglichkeiten der Bildgestaltung und Manipulation. So findet man unter dem Firmensignet der Krone einige Grundfunktionen wie z.B. die jederzeit wieder aufrufbare Printer-Installation, die Programmdefunktio-n oder auch eine recht nützliche Hilfsfunktion, die den Cursorpfeil in ein Fragezeichen verwandelt, mit dem man alle Programmfunktionen anklicken kann, zu denen dann prompt ein Fenster mit Kurzzinformationen erscheint, die dem Benutzer weiterhelfen. So

kann man durch das ganze Programm gelangen, ohne das ausführliche Manual studiert zu haben, das aber seine Berechtigung durch die Vermittlung von Feinheiten und Zusatzinformationen hat.

Hat man seine Neugier durch einen ersten Streifzug durch alle Rubriken und Untermenüpunkte erst einmal gestillt, kann man seinem gestalterischen Drang wirklich freien Lauf lassen. Es sind fast keine Grenzen gesetzt. Man kann Sprühen und Malen. Linien verschiedenster Ausführungen, Rechtecke, Felder, Kreise oder Ellipsen sind kinderleicht. Alle Flächen können mit Farben oder Mustern gefüllt werden. Auch nachträgliche Änderungen sind kein Problem. Hat man z.B. mit dem normalen Programm zum Koala Pad eine Fläche mit kleinem Muster gefüllt, sind weitere Füllversuche so gut wie hoffnungslos. Nur kleinste Stückchen werden bearbeitet. Bei Dazzle Draw „schaltet“ man diese Schwierigkeiten durch eine Zusatzfunktion zu „Fill“ einfach aus.

Man kann Bildteile ausschneiden, drehen, invertieren, kippen, kopieren, löschen und sogar in selbst definierten Ausschnitten oder in dem ganzen Bild eine Farbe durch eine andere ersetzen oder zwei Farben miteinander vertauschen. Invertieren funktioniert nicht nur bei Schwarzweiß-, sondern auch bei Farbbildern.

Ist ein kleines Mißgeschick passiert, kann man das Bild retten, indem man „Undo“ anklickt. Dadurch wird die zuletzt vorgenommene Änderung rückgängig gemacht und der alte Zustand wiederhergestellt.

Will man die Qualitäten von Dazzle Draw voll auskosten, so sollte man über die Möglichkeit der Farbdarstellung, z.B. auf einem Farbfern-seher, verfügen. Auf einem monochromen Monitor werden die Farben wie gewohnt durch unterschiedliche Schraffuren dargestellt. Das Programm ist dann zwar weiterhin benutzbar, aber es erschließt nicht die Wirkung, die unter Anwendung der Farben erzielt wird.

Eine weitere interessante Option stellt der Textmodus dar, in dem Beschriftungen in verschiedenen Größen und unterschiedlichen Schriftarten vorgenommen werden können. Sollte der Text nicht auf Anheb sitzen, kann auch hier mit der Ausschneide/Einklebe-Methode nachgeholfen werden.

Genauigkeitsfanatiker werden sich über das Zoom freuen, das einen kleinen Bildausschnitt auf das volle Format vergrößert und in einem (sogar veränderbaren) Raster die Platzierung, Einfärbung oder Löschung einzelner Punkte möglich macht, während ein Ganzbild im Miniformat am rechten, unteren Bildrand den Überblick bewahrt. Überhaupt wird außer dem oberen Bildrand, der ja zur Bereitstellung der Haupttribunen benutzt wird, auch der untere Bildrand fast immer beim Aufruf einer Funktion für weitere Optionen benutzt. Um das Bild auch hier bearbeiten zu können, kann es einfach mit einem „Hebel“ nach oben oder unten um genau diese Streifenbreite verschoben werden. Ferner wird eine raffinierte Spiegelfunktion mit verschiedenen Symmetrieebenen angeboten, die zu selbstgemalten Kaleidoskopbildern verhilft. Mit der Slide-Show-Option werden ausgewählte Bilder auf der PRO-DOS-formatierten Diskette gespeichert, wobei Reihenfolge, Standdauer und Art des Bildwechsels frei definierbar sind – eine faszinierende Möglichkeit, die Spaß macht und auch auf vielerlei Art sinnvoll genutzt werden kann. Viele Anwender, die z.B. Werbung betrei-

ben, werden sich über einen solchen individuell gestaltbaren Blickfang freuen.

Die Programmierer der Firma Broderbund haben Ihre Schularbeiten gut und gründlich gemacht. Dazzle Draw bietet weit mehr Möglichkeiten als Mousepaint. Es läuft auf einem Apple IIc oder Iie mit 128 K Speicher, benötigt keine weitere Hardware-Ergänzung (wie z.B. das Maus-Interface) und kann z.B. problemlos mit einem normalen Joystick gesteuert werden.

Die angefertigten Slide-Show-Disketten sind selbständig lauffähig und können unabhängig vom Hauptprogramm eingesetzt werden. Die Anfertigung von anspruchsvollen und hochwertig wirkenden, bildlichen Darstellungen sollte auch für den ungeübten Anwender kein Problem darstellen.

Ein weiterer Pluspunkt dürfte der erstaunlich niedrige Preis sein. Für nur ca. DM 160,- ist das Programm mit Manual bei einschlägigen Importeuren erhältlich. Broderbund beschreitet hier einen (längst fälligen) Weg, der durch das Angebot von außerordentlich attraktiven Programmen zu vernünftigen Preisen den Raubkopierern das Geschäft verderben dürfte.

## DMP-Charger – eigene Zeichensätze auf dem Matrixdrucker

getestet von Harald Grumser

Einer der wesentlichsten Vorteile von Matrixdruckern gegenüber Schönschreibern ist die Möglichkeit der Grafikausgabe und der Erstellung eigener Zeichensätze. Wer dieses Problem beim Imagewriter in Angriff nehmen wollte, wurde durch die spärliche Dokumentierung und die z.T. fehlerhaften Angaben im Druckerhandbuch jäh enttäuscht.

Die Firma Hunstig, Labor für Nachrichtentechnik, vertreibt ein Programm, mit dessen Hilfe eigene Zeichensätze erstellt werden können, um dann in Anwenderprogrammen (Appleworks, Applewriter, Wordstar...) oder eigenen Programmen eingesetzt zu werden.

### Arbeitsweise

Der DMP-Charger (Dot Matrix Printer CHARACTER GENERATOR) nutzt die Möglichkeit aus, eigene Zeichensätze in den Druckerpuffer

zu laden und diese dann bei Bedarf zu aktivieren. Dabei kann die Auflösung dieser neuen Zeichen durch entsprechende Anordnung verdoppelt werden, so daß „Mac-Like Zeichen“ (Bezeichnung der Firma Hunstig) entstehen können. Das hier vorgestellte Programm übernimmt nun die Übertragung dieses neuen Zeichensatzes und bietet neben einigen mitgelieferten Zeichensätzen einen komfortablen Editor, mit dessen Hilfe neue Zeichen erstellt oder alte Zeichen geändert werden können.

### Bedienung

Nach dem Booten der Pascal-1.2-Diskette (das Pascal-Betriebssystem ist nicht erforderlich) erscheint ein Menü, das den kompletten Ablauf regelt und den Benutzer sicher ans Ziel führt. Beim erstmaligen Gebrauch muß der Druckertyp festgelegt werden, was

später nicht mehr geändert werden kann (sollte nur auf einer Diskettenkopie erfolgen). Wenn kein neuer Zeichensatz erstellt werden soll, genügt das Laden eines bereits vorhandenen, der auch sofort aktiviert werden kann.

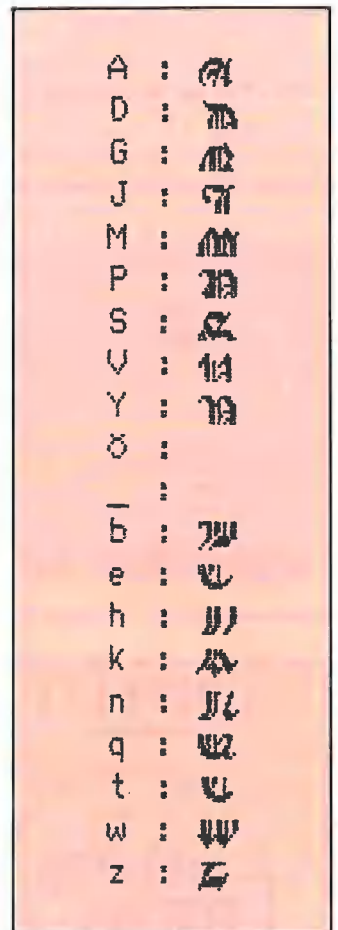
Im anderen Fall wählt man den Editor, um neue Zeichen Punkt für Punkt zu erstellen oder einzelne Zeichen zu ändern, wobei die Möglichkeit des Mischens die Flexibilität erheblich vergrößert. Der so neu gewonnene Font kann dann auf Diskette gespeichert werden. Nach Beendigung der Arbeit verläßt man das Menü und bootet das gewünschte Betriebssystem oder Programm. Danach stehen die neuen Zeichen auf dem Drucker zur Verfügung (der natürlich nicht ausgeschaltet werden darf). Die Umschaltung erfolgt dann druckerspezifisch (z.B. „ESC CHR\$(39)“ zur Aktivierung des alternativen Zeichensatzes beim Imagewriter).

### Beschreibung

Das ca. 50 Seiten umfassende, gedruckte Handbuch (Ringbuch) enthält im wesentlichen alle Hinweise, die beim Gebrauch des DMP-Charger unter den verschiedenen Sprachen und Anwenderprogrammen von Interesse sind. (Appleworks kann durch einen zu wählenden Patch derart angepaßt werden, daß die Unterstreichung das Umschalten auf den eigenen Zeichensatz bewirkt.) Mit etwas Experimentierfreude – die Voraussetzung jeder dauerhaft erfolgreichen Arbeit mit Druckern – bereitet die Einarbeitung keine Schwierigkeiten.

### Fazit

Der nicht gerade geringe Preis von DM 198,- läßt erkennen, daß der DMP-Charger vorwiegend für den kommerziellen Bereich konzipiert



DMP-Charger-Beispiel

wurde. In diesem Sinne ist seine Handhabung durchaus gerechtfertigt. Für den ausgesprochenen Selbstprogrammierer dürfte der „Stand-alone“-Charakter des Programms mit leichten Abstrichen an seine Verwendbarkeit verbunden sein. Wünschenswert wäre hier die Möglichkeit der Einbindung in Turnkey-Systeme und die Übertragung der Zeichensatz-Files in andere Betriebssysteme.

## Apple II/IIe Assembler-Kurs

getestet von Dr. Jürgen B. Kehrel

Wer den richtigen Zugang zu 6502-Assembler findet, für den ist die Maschinensprache schon bald kein Buch mit sieben Siegeln mehr, schon gar nicht ein zu fürchtendes Zahlenungeheuer. Mit dem **Assembler-Kurs** (1984, 240 S., Sybex-Verlag, mit Diskette DM 64,-) wird jetzt ein Lehrgang vorgelegt, der den direkten Einstieg

wagt. Damit bietet es eine Alternative zum Buch „Apple Maschinensprache“ (Te-Wi Verlag), das aufbauend auf Basic-Kenntnissen in die Tiefen des Apple vordringt. Da der Apple mit dem Mini-Assembler als Teil des Integer-Basic nur sehr bescheidene Möglichkeiten bietet, gehört zum Buch ein „richtiger“ Assembler, der symbo-

lische Bezeichnungen (Labels) für Sprungstellen erlaubt, Makros zuläßt und im Dezimalformat disassembliert. Laden und Speichern sind ebenso möglich wie ein Verschieben des Codes.

Was mich etwas gestört hat ist, daß Labels mit „Label Adresse“ festen Speicherplätzen zugeordnet werden, während alle anderen mir bekannten Assembler „Label EQU Adresse“ benutzen. Lokale Labels werden mit „\*Label Opcode Adresse“ definiert. Demgegenüber benutzen alle anderen den Stern als Kennzeichen für eine Kommentarzeile. Das erschwert unnötig den späteren Übergang auf große Assembler.

Ein einmal eingegebenes Programm kann mit seinen Labels nicht mehr aufgelistet werden, und auch der Editierkomfort ist minimal. Der in Applesoft geschriebene Assembler läßt nur 20 Labels, 20 symbolische Sprungadressen und 10 Makros zu, was allenfalls für kleinere Vorhaben reicht. Der vom Programm und seinen Variablen belegte Speicherplatz ist nicht geschützt. Wenn Sie also z.B. mit einer Startadresse von \$0800 assemblieren, überschreibt sich der Assembler selbst. Sie sollten also (für so wenig Geld) keinen ausgewachsenen Assembler wie Merlin oder Lisa erwarten, aber für den Einstieg ist das Programm durchaus ausreichend.

Das Buch ist, wie eigentlich immer bei Sybex, typografisch sauber hergestellt und fast fehlerfrei. Nur auf S. 49 muß es „Null-Flag Z“ statt „Übertrags-Flag C“ heißen, CHKCOM (S. 223) prüft auf Kom-

ma, nicht auf „<“, und in der Aufstellung auf S. 206 fehlt der Befehl BCS unter BCC.

Inhaltlich sollte für eine straffere Ordnung gesorgt werden. Die vorliegende 1. Auflage hat unmotivier- te Vorgriffe (NOP wird auf S. 62/63 benutzt, aber erst auf S. 70 eingeführt, ebenso ergeht es CLC und CLD), unnötige Wiederholungen (STA wird auf S. 10 und S. 12 mit geringer Variation erklärt) und ungerechtfertigte Kapiteleinteilungen (Kapitel 8 behandelt so grundverschiedene Dinge wie Interrupts, Breaks, Vorzeichen und Fließkommazahlen).

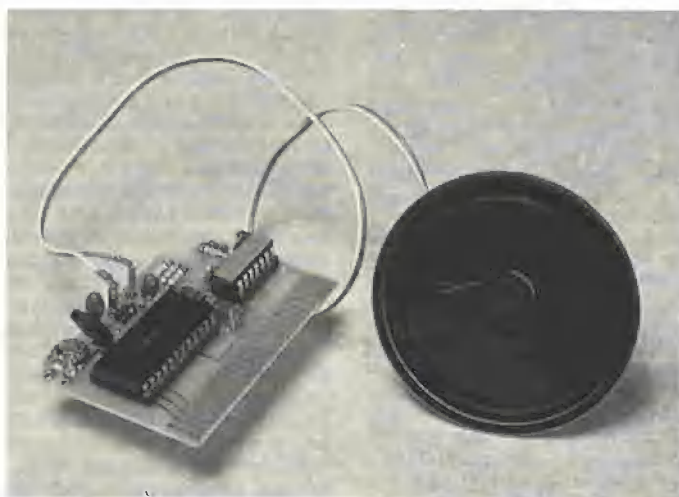
Der Leser sollte Basic kennen, Hexadezimalzahlen brauchen ihm aber nicht geläufig zu sein. Ein Trainingsprogramm für das Einüben der wichtigsten Zahlenumwandlungen wird auf der Diskette mitgeliefert. Für die fortgeschrittenere Programmierung sind in einem Anhang einige Einsprungstellen in den Applesoftinterpreter und den Monitor aufgeführt, allerdings oft mit spartanischen Angaben. „HPLOT F453: ruft HPOSN auf und zeichnet einen Punkt“ oder „COPY DAB7: String zeitweise freisetzen“ dürften nur erfahrenen Programmierern genügend Information geben.

Trotz der beschriebenen Mängel ist die Einheit aus Buch und Assembler, die Lernen durch praktisches Nachvollziehen am Gerät ermöglicht, ein erfolversprechender Weg zu einer für den Anfänger nicht ganz einfachen Sprache, die aber für viele Anwendungen (z.B. schnelle Steuerungen, bewegte Grafiken) unerlässlich ist.

phabet (Y = ypsilon). Auch einzelne Zeichen und Zahlen werden erkannt und durch die entsprechenden Wörter ersetzt (\* = mal, 3 = drei).

Wer auf eine höhere Sprachqualität Wert legt, kann die Lautfolgen auch phonetisch umschreiben. Dieses Verfahren bietet sich an, wenn gesprochene Befehle in eigene Programme aufgenommen werden sollen und sich der größere Aufwand für die Erstellung lohnt.

Deutsch treten mitunter recht lange Sprechpausen auf, so daß für längere Sätze die Lautschrift herangezogen werden sollte. Einen Anhaltspunkt für diese Schreibweise liefert die Möglichkeit, Strings nach ihrer Eingabe im „phonetischen Alphabet“ auszugeben. Etwas ärgerlich erscheint die Tatsache, daß die Ausgabe nicht durch Reset abzubrechen ist; man muß den Computer also stets ausreden lassen. (Dieses „Hängen“ nach



Bönig Sprachausgabe

Durch Verwendung dieser „Lautschrift“ – das Programm kennt 24 Phone mit jeweils 10 Tonhöhen, 10 Sprechgeschwindigkeiten und 10 Lautstärken – ist es auch möglich, z.B. englische Wörter auszugeben oder die Intonation derart zu gestalten, daß Fragesätze an der Betonung erkannt werden (heben der Stimme am Satzende). Auch wird die Betonung einzelner Vokale unterstützt, ohne alle Sprechparameter angeben zu müssen. Mit etwas Übung (und Mut zur Technik) kann so der Apple jeden feierlichen Anlaß mit einem Gedicht verschönern.

Als Manko erscheint auf den ersten Blick der geringe Umfang an Lauten. Die Praxis zeigt jedoch, daß fast alle Wörter bei entsprechender Umschreibung verständlich werden. Bei Umsetzung von Schriftdeutsch in gesprochenes

Reset kann durch POKE 49254 + SLOT \* 16, 255 unterbrochen werden.)

*Anmerkung zur Lautschrift:* Die Internationale Lautschrift (Alphabet der API) weist 57 Phone aus. Davon treten ca. 10 Laute nicht in deutschen Wörtern auf. Der Phonschatz eines Sprachprozessors ist nicht identisch mit diesen Lauten (der hier verwendete kennt 64 Laute).

**Fazit:** Die Sprachqualität dieser Karte überrascht durch ihren deutschen Akzent. Die mitgelieferte Software ermöglicht einen problemlosen Einbau in eigene Programme, wobei die Umsetzung von geschriebenem Wort in gesprochenen Laut geglückt ist. Mit einem Preis von DM 244,- tritt sie nicht nur qualitativ in Konkurrenz zu teureren Sprachkarten.

## Deutsche Sprachausgabe für den Apple getestet von Harald Grumser

Die Ausgabe erfolgt beim Rechner i. allg. auf Bildschirm oder Drucker. In interaktiven Programmen können Anweisungen an den Benutzer oft nicht auf dem Bildschirm ausgegeben werden. Auch ist der Vergleich von Zahlenkolonnen oder Texten zwischen Papier und Bildschirm eine unangenehme Tätigkeit. Hier kann eine akustische Ausgabe nützlich sein, wie sie von der Firma Marco Bönig angeboten wird. Der Apple wird mit dieser kleinen Sprachkarte, die vorzugsweise in Slot 4 steckt, in die Lage versetzt, deutsche Texte zu sprechen.

Mit Hilfe der mitgelieferten Programme können deutsche Texte direkt zur Ausgabe gebracht werden, wodurch sich mit minimalem Aufwand eigene Vorleseprogramme schreiben lassen. Die Umsetzung in die einzelnen Phone erfolgt automatisch, so daß ein String mit deutschen Wörtern an das Sprachprogramm übergeben werden kann, ohne daß man sich über die Lautschrift Gedanken zu machen braucht. Die Wörter sind erstaunlicherweise recht gut verständlich. Darüber hinaus erkennt das Programm einzelne Buchstaben und spricht diese wie im Al-

**Lerne zu pieken  
ohne zu quieken!**

## Kurzberichte

(Gekürzte Firmenmitteilungen)

### 512K für Apple III

Ab sofort können alle Apple-III-Anwender ihren Rechner auf die maximale Kapazität von 512K aufrüsten. Bei bestehenden 256K-Rechnern wird das vorhandene Erweiterungsboard einfach gegen die neue Karte ausgetauscht, ohne daß ein Slot benötigt wird. Die Montage ist so einfach, daß dafür kein Händler hinzugezogen werden muß. Lediglich bei bestehenden 128K-Rechnern muß der Einbau von einem Apple-Händler durchgeführt werden. Alle SOS-Programme mit dynamischem Speicherzugriff (Applewriter Version 1.0 also nicht) nutzen den neuen Speicherplatz vollständig. Die Speichererweiterung ist zu einem Preis von ca. DM 3200,- bei Dipro-Software erhältlich. *Quelle: Dipro-Software, Pulheim*

### Apple IIe-Emulation für Apple III

Mit einer neuen Interfacekarte für den Apple III ist es möglich, Apple-IIe-Programme mit einem Speicherbedarf von 64K und mehr auf dem Apple III zu betreiben. Die Karte ist mit der Language-Card software-kompatibel und enthält gleichzeitig einen Game-Port zum Anschluß von Joysticks oder Paddles. Versionen mit 16, 64 oder 128K sind lieferbar. Damit stehen im Emulation-Mode maximal 176K über Bank-Switching zur Verfügung. Ebenfalls kann das ProDOS-Betriebssystem gefahren werden. *Quelle: Dipro-Software, Pulheim*

### Lisa/Mac-Screenswitcher

Apple stellt die Produktion des Macintosh XL (Lisa) im Herbst 85 ein. Den Lisa-Besitzern wird empfohlen, ab sofort die Macintosh-Emulation „Macworks“ als Betriebssystem zu benutzen. Das Problem dabei ist, daß Lisa rechteckige Bildschirmpunkte verwendet, Macintosh dagegen quadratische. Die auftretende vertikale Verzerrung des Bildschirms im „Macworks“-Modus wird durch den Lisa/Mac-Screenswitcher verhindert. Durch einfaches Umschalten wechselt das Bildschirmformat von Lisa auf Macintosh. Sowohl die Lisa- als auch die Macintosh-Software kann weiterhin problemlos benutzt werden. Der Lisa/Mac-Screenswitcher ist so konzipiert, daß keine äußerlichen Veränderungen an der Lisa nötig sind. *Quelle: Pythia GmbH, München*

### 640K-Drive für IIc

Viele Anwendungen im kommerziellen Bereich lassen sich mit einem Apple IIc gut abwickeln. Mit dem Essen kommt jedoch der Appetit – und die Erkenntnis, daß sich mit dem eingebauten 140K-Drive der Apple IIc nicht seiner Leistungsfähigkeit entsprechend einsetzen läßt. Nun ist es endlich möglich, in Verbindung mit dem Apple IIc größere Laufwerke zu verwenden. Es stehen dem Anwender zusätzlich 640K (neben

dem eingebauten Laufwerk) zur Verfügung, 640K deshalb, weil das zusätzliche Drive zwei Schreib/Leseköpfe besitzt und so zwei Diskettenseiten lesen bzw. schreiben kann. Leider können diese beiden Diskettenseiten nicht ohne weiteres als ein einziges Volume verwaltet werden. Das resultiert aus den fehlenden Signalleitungen im Apple-IIc-Stecker, die zur Umschaltung benötigt werden. Um trotzdem die Rückseite der Diskette nutzen zu können, wurde eine Schaltung entworfen, welche zwischen Hauptplatine und CPU eingesetzt wird. Außerdem läßt sich das Laufwerk auch auf 40 bzw. 35 Spuren umschalten, um es als „normales“ Drive zu benutzen (nur das 5.25-Zoll-Drive). Ein menügesteuertes Installationsprogramm ermöglicht auf einfachste Weise das Modifizieren der drei häufigsten Betriebssysteme (DOS 3.3, ProDOS und UCSD-Pascal), um die Zusammenarbeit dieser bis jetzt ungewöhnlichen Laufwerkskombination zu gewährleisten. Eine Anpassung für die CP/M-Erweiterung ist in Vorbereitung. Geliefert werden ein Teac FD 55F (5,25 Zoll) bzw. ein Teac FD 35F (3,5 Zoll) im Gehäuse inkl. Anschlußkabel, Zusatzplatine und einer „Patchdiskette“, womit die Betriebssysteme DOS 3.3, UCSD-Pascal und ProDOS angepaßt werden müssen. *Quelle: ccp Datentechnik, Hamburg*

### Apple-Paketpreise

Preisgünstige Mikrocomputer-Pakete, die sich ohne zusätzliche Zubehörkäufe sofort kommerziell einsetzen lassen, bietet die Apple Computer GmbH. Ab dem 1. Juli 1985 haben die rund 300 Apple-Fachhändler sechs Systemkonfigurationen im Programm, die einschließlich professioneller Software im Preis etwa 15% günstiger sind. Das Apple IIc „Profi-Paket“ kostet 5990,- DM und umfaßt das Apple-IIc-System, den Apple-IIc-Monitor, den Apple-IIc-Monitorstand und ein externes Laufwerk. Als Software ist das integrierte Programm „Appleworks“ mitverpackt, das Tabellenkalkulation, Textverarbeitung und Datenbank umfaßt. Das Apple IIe „Profi-Paket 128K“ kostet 6990,- DM und umfaßt den Apple IIe 128K mit 80-Zeichendarstellung, einen Apple-IIe-Monitor und das Apple-Duodisk-Laufwerk. Als Software ist das derzeit meistverkaufte integrierte Software-Programm in den USA, „Appleworks“, mitverpackt. *Quelle: Apple-Händler*

### Apple-Geschichte

#### Apple USA

1977: Steven P. Jobs (21) und Stephan G. Wozniak (26) gründen die Firma Apple Computers Inc. Auf der West-Coast-Computermesse in San Francisco wird der Apple II vorgestellt.

1980: Apple baut in Cork/Irland eine Fabrikation für den Apple II. Von hier

aus wird der europäische Markt beliefert. Eine weitere Fabrik wird in diesem Jahr in Dallas/Texas in Betrieb genommen.

1981: Apple eröffnet in München das deutsche Tochterunternehmen. Zehn Mitarbeiter zählt Apple Deutschland bei seiner Gründung. Weitere Apple-Töchter gibt es in Großbritannien, Frankreich und Italien. Die europäischen Aktivitäten werden vom Pariser Vorort Neuilly aus koordiniert. Die Apple-Produktion in Singapur wird aufgenommen.

1982: Apple gelingt der Eintritt in den Club der 500 führenden Unternehmen der USA.

1983: Der 1.000.000ste Apple verläßt das Werk in Dallas. Im Juni tritt John Sculley (45) seinen Posten als Präsident von Apple an. Der erfahrene Marketingprofi Sculley kommt aus dem Pepsi-Cola-Management. Steven P. Jobs übernimmt in der Firmenleitung den Aufsichtsratsvorsitz. Apple stellt als neue Produkte die Lisa (später umbenannt in Macintosh XL) und den Apple III vor. Tochterfirmen in Japan und Australien werden gegründet.

1984: Apple erreicht im Geschäftsjahr 1983/84 einen Umsatz von über 1,5 Mrd. US-Dollar (etwa 4,6 Mrd. DM). Apple stellt im Januar den Macintosh vor, der in weniger als 100 Tagen zum dritten Meilenstein der Personal-Computer-Geschichte wird. Der Apple IIc wird in San Francisco 4.500 Journalisten, Händlern und Mitarbeitern vorgestellt. Die Präsentation wird via Satellit in andere Apple-Niederlassungen übertragen. 4.750 Beschäftigte arbeiten weltweit für Apple. Monatlich verlassen 120.000 Apple-Computer die Produktionsstätten.

Insgesamt werden im Geschäftsjahr 1983/84 1,2 Mio. Apple-Systeme verkauft.

1985: Im Januar kündigen Apple und Linotype in Frankfurt die Zusammenarbeit mit dem Ziel an, daß im Macintosh eingegebene Texte und Grafiken auf dem Laserbildgerät von Linotype in eine Druckvorlage umgesetzt werden können. Die technischen Voraussetzungen dazu sollen in der zweiten Jahreshälfte 1985 verfügbar sein. Damit bekommt der Macintosh von Apple bald eine zusätzliche Aufgabe im modernen Büro. *Quelle: Apple Deutschland*

1981: Die deutsche Apple-Tochter in München nimmt mit zehn Mitarbeitern die Arbeit auf. Es ist die dritte europäische Niederlassung des amerikanischen Unternehmens.

1984: Ralph M. Deja (42) übernimmt den Posten als deutscher Geschäftsführer von Apple. Die mittlerweile 90 Mitarbeiter beziehen die neue Münchener Apple-Zentrale an der Ingolstädter Straße. Fünf regionale Niederlassungen in Hamburg, Düsseldorf, Frankfurt, Stuttgart und München haben die Aufgabe, das Händlernetz zu betreiben. Über 300 Fachhändler vertreiben Apple-Computer in Deutschland.

1985: Stephan G. Wozniak verläßt Apple. Steven P. Jobs tritt vom Macintosh-

Vorsitz zurück, Aufhebung der Firmenspaltung Macintosh-Apple II, weltweite Personalreduzierung um über 20%, in den USA um über 1500 Mitarbeiter, in Deutschland auf 56 Mitarbeiter, Einschränkung der regionalen Niederlassungen, Beginn der Reorganisation und Konsolidierung.

*Quelle (außer dem letzten Absatz): Apple Computer GmbH, München*

### Mactablet



Das Mactablet ist ein Digitalisiergerät, speziell entwickelt für den Macintosh. Das Mactablet hat eine aktive Fläche von 152 x 228 mm. Durch die neuartige elektromagnetische Technik ist es möglich, eine Auflösung von bis zu 20 Linien/mm zu erreichen. Die maximale Vorlagenstärke kann bis zu 12 mm betragen. Geliefert wird das Mactablet mit einer Stromversorgung, Abtaststift, deutscher Bedienungsanleitung und einer Installationssoftware-Diskette. Die mitgelieferte Software ermöglicht es, das Mactablet für jede Macintosh-Software als „Desk Accessory“ zu installieren. Das Datenkabel des Mactablet wird in den Printer- oder Modem-Port des Apple Macintosh eingesteckt. Daraus ergibt sich die Möglichkeit, Maus und Mactablet je nach Anwendung zu benutzen. Das flache Kunststoffgehäuse ist mit einer Schrägstellvorrichtung versehen. Als Abtastvorrichtung kann außer dem Stift auch ein Cursor verwendet werden, der ein präzises Übertragen von technischen Unterlagen erleichtert. Für freies Zeichnen, z.B. in Verbindung mit Macpaint, ist der Stift das ideale Werkzeug. Mit dem Mactablet von Summagraphics kann man endlich alle grafischen Möglichkeiten des Macintosh nutzen. *Quelle: Summagraphics, München*

### Macpack

Macpack ist ein neues Programm im Macintosh-Design zur schnellen und komfortablen Abwicklung des Paketverands mit dem United Parcel Service für Unternehmen mit mittlerem bis großem Paketvolumen. Durch die Wahl des Macintosh ist das Programm besonders leicht zu erlernen und zu bedienen. Der geringe Platzbedarf des kompletten Rechnersystems erlaubt den Einsatz auch unter räumlich beengten Bedingungen. Das robuste Gerät ist ideal für den Einsatz im Lager geeignet. *Quelle: Format Software GmbH, Köln*

### Incircuit-Emulator

Die Incircuit-Emulatorkarte EMU macht aus einem Apple II Plus oder kompatiblen Rechner ein preisgünstiges Hard-



ware- und Software-Entwicklungssystem für die 6502-CPU. In Verbindung mit einem guten Assembler und Debugger lassen sich selbst größere Hardware-Entwicklungen durchführen, wenn kleine Einschränkungen in Kauf genommen werden (ein Lade- und ein Speicherbefehl bei Überschreiten einer 2K-Seite). Der geübte 6502-Assemblerprogrammierer wird nach kurzer Zeit mit dem Emulator zurecht kommen.

#### Prinzip:

Durch Ersetzen der CPU im Zielsystem durch den 40-poligen DIL-Stecker des Emulators wird das Zielsystem mit dem Apple verbunden. In das Zeigerregister wird derjenige 2K-Bereich des Zielsystems eingeschrieben, der angesprochen werden soll. Durch das „2K-Fenster“ des Apple (\$C800-\$CFFF) ist das Zielsystem ansprechbar. Es kann mit Interrupts gearbeitet werden, wenn im Apple in die Adressen \$03FE das Low-Byte und in \$03FF das High-Byte der Adresse der Interruptroutine geschrieben ist. Das auszuführende Programm wird im Apple abgelegt (z.B. ab \$0803).  
Quelle: Gerhard Brecht, Stuttgart

#### Apple-Kompatibler MK II

Kurzbeschreibung in Stichworten:

- Apple-kompatibler Rechner mit BASIC im ROM.
- 8 Erweiterungssteckplätze. Steckplatz 6 enthält den Floppy-Controller.
- 2 Disketten-Laufwerke sind eingebaut. Gegen Aufpreis Controller + Diskettenstationen 640K möglich.
- Formschönes 19"-Gehäuse beige/braun.
- Abgesetzte Preh-Tastatur AK87 mit Zehner-Block; gleiche Gehäusefarbe.
- Auf der Rückseite des Rechnergehäuses sind Ausbrüche für Steckverbindungen vorgesehen, 2 geschaltete Schukosteckdosen für den Anschluß von Monitor und Drucker sind bereits eingebaut.
- Drucker-Interface in Steckplatz 1 (grafikfähig).
- 16K-Karte.
- Prozessor-Karte mit Z80 gegen Aufpreis (Original Microsoft-Karte mit CP/M-Betriebssystem, MBasic usw.).
- 80-Zeichen-Bildschirmdarstellung mit Software-Umschaltung auf Grafik und 40 Zeichen. Zwei programmierbare Helligkeiten (Hilite/Lolite), deutscher Zeichensatz, Block- und Linien-Grafik.
- Monitor mit 22 MHz Bandbreite, bernstein.

Quelle: Ewald Balfer, Krombach



MK II

#### Erno Unibox

Kompaktes Disketten-Aufbewahrungssystem mit direktem Zugriff auf bis zu 80 Disketten im 5,25-Zoll-Format; 8 weit aufklappbare, staubgeschützte Abteile; im aufgeklappten Zustand kein größerer Raumbedarf erforderlich; komplett mit Archivierungsetiketten; hochwertiger Kunststoff mit Anti-Statik-Behandlung; Sockelfarbe computerbeige; Abteillfarbe in braun-rauchglas als Lichtschutz; Kompaktmaße: L 257 x B 173 x H 143 mm.  
Quelle: Erno-Photo GmbH, Albrück



Unibox

#### 20M-Festplatte für Mac

Ab Herbst dieses Jahres wird der Macintosh 512K mit einer externen 20M-Festplatte geliefert. Wie die Apple Computer Inc. ankündigt, erweitert die neue Macintosh-Konfiguration das modular aufgebaute „Macintosh Office“ nach oben. Die neue Festplatte ist für den Macintosh als individueller Arbeitsplatz entwickelt worden und kann 50mal mehr Daten speichern als eine 400K-Floppy-Disk. Die externe 20M-Festplatte ergänzt den auf der Hannover-Messe in diesem Jahr angekündigten File-Server, der innerhalb des neuen Apple-Netzwerks „Appletalk“ ein Hauptelement für den Multiuser-Betrieb des „Macintosh Office“ ist.

Quelle: Apple Computer GmbH, München

#### Appletalk für Macintosh

Appletalk heißt das neue Netzwerk von Apple, das Computer und Peripherie zu einem elektronischen Büro zusammenfügt. Über Appletalk können bis zu 32 Macintosh-Benutzer untereinander kommunizieren. Mit Appletalk können sie auch mit Anwendern anderer Personalcomputer und mit der Groß-EDV kommunizieren. Appletalk ist ein komplexes, steckfertiges Netzwerk, das die Teamarbeit in Büros unterstützt. Es kostet rund 200,- DM pro Verbindung.

#### Technische Stichworte

Elemente: Anschlußkabel mit Anschlußbox, 2m Verbindungskabel, Endloskabel für lose Verlegung; Anschlußkapazität: bis zu 32 Geräte; Entfernung: bis zu 300m; Übertragungskapazität: 230 Kilobits/s; Standard: Offener Allzweck-Standard (ISO), alle Ebenen dokumentiert; Kompatibilität: Gateways zu SNA und Ethernet.

Quelle: Apple Computer GmbH, München

#### Apple-Schul-Programm AULA

Ein weiteres Sonderprogramm hat die Apple Computer GmbH, München, für den Ausbildungsbereich eingeführt, um die starke Position in diesem Sektor auszubauen. Das neue AULA-Programm richtet sich an die Sekundarstufen I und II aller öffentlichen und staatlich anerkannten Schulen. In Kooperation mit seinen autorisierten Fachhändlern bietet Apple den jeweiligen Schulträgern bestimmte System-Pakete ab einer Mindestabnahmemenge von 5 bzw. 9 Konfigurationen zu Sonderpreisen. Das AULA-Programm umfaßt die Apple-II-Familie sowie den Macintosh 128K. In Verbindung mit einer Hardware-Bestellung kann Apple-Schul-Software zu Sonderpreisen geliefert werden.

Quelle: Apple Computer GmbH, München

#### Laserwriter von Apple

Laserwriter heißt das neue Flaggschiff unter den Apple-Peripherie-Geräten, das sich mit seinen multifunktionalen Leistungsmerkmalen deutlich aus der allgemeinen Druckerwelt hervorhebt. Eingebaute Zeichensätze geben dem Anwender den Vorteil, Zeichen und

Grafik in Laser-Auflösung zu erhalten und zu mischen. Die Qualität kommt der von Satzmaschinen nahe und ergibt sich aus der ungewöhnlich hohen Auflösung von 300 gegenüber 70 dpi bei „Normaldruckern“. (Vergleiche hierzu Pecker, 7/85, S. 61: Lichtsatzanlagen haben über 1000 dpi.)

Verschiedene Schrifttypen bieten die uneingeschränkte Möglichkeit, Formulare, Berichte, Geschäftsgrafiken und sogar Präsentationsfolien selbst zu erstellen.

Erreicht werden diese vielfältigen Leistungsmerkmale durch einen eingebauten Mikrocomputer mit 68000-Prozessor, 1,5M RAM und 0,5M ROM. Natürlich ist sämtliche Apple-Macintosh-Software kompatibel mit dem Laserwriter, so daß z.B. professionelle Software-Pakete wie Jazz von Lotus oder die Software-Serien von Microsoft genutzt werden können. Preis ca. DM 27.500,-.

Quelle: Apple Computer GmbH, München

#### Design-Preise für Apple IIc

Gekrönt wurde der Markterfolg des Apple IIc jetzt durch die zweifache Design-Auszeichnung „if Die gute Industrieform“ für den Scribe-Drucker und für den Apple-IIc-Monitor von einer internationalen Jury anlässlich der diesjährigen Hannover-Messe 1985. Den beachtlichen Erfolg verdankt der Apple IIc seiner professionellen Einsatzmöglichkeit im „Persönlichen Computern“ sowohl bei kommerziellen Anwendungen als auch in der Schule und zu Hause.

Quelle: Apple Computer GmbH, München



Spooler

#### Spooler für Epson-Drucker

Einen Hardware-Spooler speziell für den Einbau in Epson-Drucker der Serien RX, MX und FX stellt die Firma Leunig vor. Er ist lieferbar in zwei Versionen: mit serieller oder mit paralleler Rechnerschnittstelle. Die Speicherkapazität ist in Stufen aufrüstbar bis 64K und wird durch interne Datenkompression im Mittel um etwa 50% vergrößert. Die Steuerung der Spooler-Funktion übernimmt ein eigener Spezialprozessor, der den Bauteileaufwand und Stromverbrauch auf ein Minimum reduziert. Die Übernahmegeschwindigkeit vom Rechner beträgt 1600 Zeichen/s. Der Einbau dauert etwa 5 Minuten. Die umgerüsteten Epson-Drucker bleiben völlig hard- und software-kompatibel.

Quelle: Leunig GmbH, Neunkirchen

#### Transfer II Terminalprogramm

Die Firma Gerhardt und van Mergem liefert ein neues Terminalprogramm und V24-Verbindung vom Game-Port zum

Akustikkoppler für Apple II Plus/IIe und Kompatible, Charakteristik in Stichworten:

Terminal-Mode: Drei verschiedene Zeichendarstellungen auf dem Bildschirm mit jeweils deutschem oder englischem Zeichensatz, 40 Zeichen/Zeile, Apple-Schriftsatz, 70 Z/Z, 70-130 Z/Z Proportionalsschrift.

Transfer-Mode: Up/Download beliebiger DOS-Files zwischen zwei Apple-II/Transfer-II-Besitzern.

Text-Editor-Mode: Screeneditor mit 40 Z/Z und Fenstertechnik, verschiedene Move-Befehle, wie z.B. Seitenblättern, Textanfang/-ende, gelöschte Zeichen versetzen, Suchkommandos.

Print-Mode: Ausdrucken des Editorfiles, paralleler Ausdruck der Kommunikation, Parameter einstellen.

DOS-Befehle: Catalog, Diskette löschen, Diskette initialisieren, Drive wechseln.

Quelle: Jochen Gerhardt und Bettina van Mergern GbR, Düsseldorf



#### Druckertisch

Über die Firma Dahlhoff ist ein neuer Druckertisch erhältlich. Charakteristika: Maße: 74 x 55 x 75 cm (HBT) – Schalldämmung um 6db oder 50% – Ausführung: elfenbein mit Glasdeckel – fahrbar, damit er überall einsetzbar ist – separate Steckerzufuhr, damit sich kein Kabel mehr verknottet – zwei Ablageböden – separate Papierzu- und -abfuhr – Preis ca. DM 300,-.

Quelle: Jürgen Dahlhoff, Soest

#### Notstromaggregat

Das TCS SL-10 ist ein Notstromaggregat, daß sich bei Netzausfall selbständig zwischen Netz und Computer schaltet, bevor der Rechner abstürzen kann und Daten verloren gehen. Ein Akku im Inneren des Gerätes erlaubt es – bei 200 Watt Abgabeleistung – bis zu zwei Stunden netzunabhängig und lautlos zu arbeiten, so daß genügend Zeit besteht, alle ungesicherten Daten zu speichern. Ist die Netzspannung wieder hergestellt, schaltet das SL-10 automatisch auf Netzversorgung um und erneuert die Leistungsreserven des integrierten Akkumulators. Eine ständige Ausgangsspannung von 220 V/50 Hz ist somit garantiert. Preis ca. DM 2300,-.

Quelle: TCS Computer GmbH, St. Augustin 2

## Leserbriefe

### Wie Apfel-Tasten programmieren?

Ich bitte Sie, mir bei der Programmierung der „Apple“-Tasten zu helfen. Da sie nicht im ASCII-Code enthalten sind, weiß ich nicht, wie ich die Tasten ansteuern soll (aus einem BASIC-Programm).

Thomas Effert, Kretz

Antwort: Dies ist ganz einfach: Wenn PEEK (49249) größer als 127 ist, dann wurde die linke, und wenn PEEK (49250) größer als 127 ist, die rechte Apfel-Taste gedrückt. Beispiel:

```
10 IF PEEK (49249) > 127
   THEN PRINT "LINKS"
20 IF PEEK (49250) > 127
   THEN PRINT "RECHTS"
30 GOTO 10
   US
```

### 80-Zeichenkarte ohne Double-Hires

Im Oktober 1984 bestellte ich bei der Firma Prosoft in Koblenz einen Apple IIe mit Monitor, zwei Laufwerken und einer Apple-80-Zeichenkarte. Nach erfreulich kurzer Zeit erfolgte die Lieferung. Beim Auspacken bemerkte ich, daß die 80-Zeichenkarte nicht, wie die anderen Artikel, in einem Apple-Karton verpackt war. Als blutiger Neuling dachte ich mir jedoch nichts dabei. Die Erleuchtung, daß ich wohl nicht die Apple-Karte, wie auf dem Lieferschein angegeben, erhalten hatte, kam mir erst, als ich Peeker Nr. 1/84 in Händen hielt und die Experimente zur Double-Lores ausprobierte, welche nicht funktionierten.

Nach einer telefonischen Rücksprache, bei der sich die Firma für den Fehler entschuldigte, schickte ich die Nachbau-Karte zusammen mit einem Begleitschreiben zurück. In diesem Schreiben bat ich darum, mir entweder die Apple-Karte oder das bereits bezahlte Geld zurückzusenden. Ich warte noch immer....

Zu guter Letzt noch einige Anmerkungen zu Peeker. Sehr gut gefallen hat mir im letzten Heft (7/85) der Assemblerkurs. Da hatte man endlich einmal das Wichtigste auf wenigen Seiten zusammengefaßt. Aufgrund meiner Erfahrungen würde mich ein Test verschiedener 80-Zeichenkarten im Hinblick auf Kompatibilität interessieren. Als Test eignet sich hierzu ja die Double-Lores bzw. -Hires.

Albert Birkicht, München

### Wie Double-Hires feststellen?

Wir bekommen immer wieder Anfragen bzw. Anrufe der obigen Art bezüglich der Double-Hires in Verbindung mit verschiedenen 64K-Karten. Mit dem nachfolgenden Miniprogramm können Sie sofort ermitteln, ob Ihre 64K-Karte double-hires-fähig ist oder nicht:

```
10 PR#3: PRINT: HGR
20 POKE 49246,0
30 POKE 49237,0: POKE 8192,170
40 POKE 49236,0: POKE 8192,170
50 GOTO 30
```

Wenn Ihre 64K-Karte in Ordnung ist, dann sehen Sie auf dem Bildschirm ganz oben links als Grafikzeile 6 Bildpunkte nebeneinander, ansonsten nur 3 Bildpunkte. Die 6 Bildpunkte sind in 2 „Grüppchen“ zu je 3 Bildpunkten getrennt (mit einem nicht-sichtbaren Bildpunkt dazwischen). Wenn man nicht über Double-Hires verfügt, sieht man nur 1 „Grüppchen“. Da die auf der 64K-Karte befindliche zweite Hälfte des Double-Hires-Speichers mit dem HGR-Befehl nicht gelöscht wird, sehen Sie ggf. zusätzlich ein wirres Grafikumster. Dies ist normal. us

### LoB

Herzlichen Glückwunsch zu Ihrer rundum gelungenen Zeitschrift. Es dürfte wohl keine einzige deutsche Publikation geben, die ähnlich gute und reichhaltige Informationen und Beiträge über die Apple-II-Serie veröffentlicht.

Karlheinz Eberhardt, München

### 40-Spur-Patch für ProDOS-Filer

In Ihrer Zeitschrift sind zwar bereits mehrere Formatierungsprogramme für ProDOS abgedruckt worden. Diese hatten aber alle eine Länge, die das Abtippen zur Gedulds- und Nervenprobe machten, denn beim Abtippen eines längeren Assembler- oder Hexlistings schleichen sich immer wieder Fehler ein. Es ist deshalb günstiger, die vorhandene Software zu patchen. Hier die Modifikationen für ProDOS 1.0.1: Erstellen einer Arbeitskopie der ProDOS-Systemdiskette, damit nicht die Originalfiles versehentlich zerstört werden, dann:

1. BLOAD PRODOS, A\$2000, TSYS
2. CALL-151
3. 5209:4C 10 F8 (Hiermit wird die 280-Blocksperrung abgeschaltet)
4. UNLOCK PRODOS
5. BSAVE PRODOS, A\$2000, L15360, TSYS
6. BLOAD FILER, A\$2000, TSYS
7. In \$4244 wird das Low Byte der Blockanzahl (Spuren \* 8) eingetragen (für 40 Spuren 4243:40)
8. In \$4246 wird das High Byte der Blockanzahl eingetragen (4246:01 für 40 Spuren)
9. In \$79F4 wird die Spurzahl eingetragen (für 40 Spuren: 79F4:28)
10. BSAVE FILER, A\$2000, L25600, TSYS

Im übrigen gefällt mir Ihre Zeitschrift recht gut, da Sie genau das bringen, was in anderen Zeitschriften meist fehlt: hochwertige Tips, die das Arbeiten erleichtern.

Detlev Rackow, Wunstorf

### Druckfehlerteufel

Als Leser Ihrer Apple-Computer-Zeitschrift habe ich leider festgestellt, daß in der Ausgabe 6/85 der „apple-worm“ (siehe Spektrum der Wissenschaft 5/85) Ihnen undel mitgespielt hat, denn in den Programmlistings finden sich verschiedene schwerwiegende Fehler:

1. RAM-Disk-Driver für Pascal 1.1: Im Maschinenprogramm INIT.PASCAL.SOURCE fehlt die Unteroutine LOAD. (Bereits nachgetragen im Peeker 7/85, S. 52, us).

2. Fourier-Analyse: Der Unterprogrammteil 4310, aufgerufen von Zeile 1600 fehlt (siehe Rahmen, us).

Generell möchte ich die Gelegenheiten ergreifen, noch einige kritische Anmerkungen zu Ihrer Zeitschrift anzuführen:

1. Ich finde es nicht gut, daß einige Programme auf mehrere Hefte und somit auf mehrere Monate verteilt werden. Zum Beispiel wird die Graf-quattro-Serie auf zu viele Hefte verteilt (wird abgestellt; nach einem vorläufigen Abschluß wird Graf-quattro wahrscheinlich als Sonderheft erscheinen, us)

2. Es wäre sicherlich sinnvoller, die Sammeldisketten nicht chronologisch, sondern thematisch zusammenzufassen (z.B. Grafik, DOS-Hilfen, Kopierprogramme, Pascal-Hilfen, Spiel-, Mathematikprogramme, BASIC-Hilfen usw.). Meines Erachtens würde sich dadurch die Attraktivität der Disketten erhöhen.

Weiterhin wünsche ich mir folgende Programme im Peeker, die von sehr allgemeinem Interesse sein dürften:

1. Ein Eingabeprogramm für Maschinenprogramme in Hexadezimal-Code.
2. Ein Kontrollprogramm für Maschinenprogramme, das die Richtigkeit der Eingabe überprüft (liegt bereits vor; erscheint demnächst, us).
3. Ein Hires-Druckausgabeprogramm (ähnlich die Superdump) für Pascal-Programme (liegt bei Jürgen Geiß bereits vor; erscheint zu einem späteren Zeitpunkt, us)

Jan Leutenantsmeyer, Bad Bentheim

### Verbesserungsvorschläge

Weiter so: Peeker ist eine Zeitschrift, die ihr Geld wert ist. Aber trotzdem habe ich einige Verbesserungsvorschläge:

1. Ihre Programme, Bücher und Disketten sind so gut, daß sie so viel Werbung gar nicht nötig haben. Nutzen Sie die kostbaren Druckseiten für Wichtigeres.
2. Wir wissen nun alle, daß Sie etwas gegen den Mac haben – wie ich im übrigen auch. Aber laßt doch denen ihren Mac, für die der Mac bestimmt ist und nutzt den Platz.
3. Die meisten Artikel sind sehr interessant. Beim ersten Überfliegen der Artikel läuft mir das Wasser im Munde zusammen. Doch bald folgt die ernüchternde Feststellung: Das Mahl war nicht für mich bestimmt. Das leidige Problem der Kompatibilität. Kein Apple gleicht dem anderen. Drei Beispiele:

1. Heft 6/85 berichtet über die RAM-Disk für CP/M 2.20, 56K. Die RAM-Disk läuft sehr gut unter der 56K-Version. Da ich aber lieber CP/M 2.23, 60K benutze, denn diese verträgt sich besser mit meiner Maus, kann ich die RAM-Disk nicht benutzen. Nicht nur, daß die RAM-Disk die Bank 2 der unteren 64K nicht nutzen darf, sondern diese Version belegt die Seite \$FC00 und hat auch andere Einsprungsadressen.

2. In Heft 7/85 wurde der Programmtext eines Fonteditors für die Vindex-Karte veröffentlicht. Ich habe aber einen Apple IIe mit 80-Zeichenkarte. Wo sind meine Zeichensatz-EPROMs und wie sind diese aufgebaut? 7 \* 8 Pixel?

3. Im gleichen Bericht wird die Anpassung von Multiplan an die inverse Darstellung von Feldern angedeutet. Aber wie?

Ich erwarte keine Programme für jede Version, aber vielleicht ist es möglich, am Ende aller Berichte Literaturhinweise anzugeben oder an anderer Stelle kleine Tips zu geben, damit alle die Programme besser nutzen können.  
*Wilfried Pohl, Hamburg*

## CP/M-RAM-Disk

Herzlichen Dank für die Veröffentlichung des RAM-Disk-Drivers für die 64K-Karte unter CP/M. Schwierigkeiten gab es allerdings bei der Eingabe des Assemblerprogramms RDSKINIT mit Hilfe des DDT-Programms: Meines Erachtens stimmt der auf S. 61 abgedruckte „Hexdump“ nicht mit dem Assemblerprogramm S. 60/61 überein. Erst bei Eingabe des im Quellprogramm enthaltenen Hexcodes mit DDT stellt sich der versprochene Erfolg ein! Sofern ich den Artikel nicht gründlich mißverstanden habe, sollten Sie Ihre Leser auf diese Unstimmigkeit hinweisen bzw. einen korrigierten und übersichtlicheren „Hexdump“ abdrucken. (Die jeweils letzte Hexzahl jeder Zeile ist eine Prüfziffer und darf nicht mit abgetippt werden. us)

*Roland Schenkel, Otterbach*

## Fourier-Analyse

Das Programm FOURIER.MAIN, von dem ich nur den Analyseteil benutzte, hatte leider einige kleine Fehler. Die Bildschirmgrafik arbeitete nicht und die numerischen Ergebnisse waren offensichtlich falsch. Da ich kein Mathematiker bin, mußte ich erst einmal zu den zitierten Quellen hinabsteigen. Denn vor der Reparatur steht das Verständnis der dem Programm zugrunde liegenden Algorithmen. Es fehlte zunächst die Umschaltung von 80 auf 40 Zeilen vor Aufruf der HGR-Routinen. Dazu fügte ich ein Zeile

3135 Print CHR\$(21)

(Ist nicht erforderlich bei Ile/c. us), und danach arbeitete die Grafik. In Zeilen 1920, 1990, 2110 einerseits und Zeilen 1920, 2000, 2110, 2170 andererseits stehen gleichlautende Ausdrücke der Form: IF L (bzw. K) = M ... Diese wurden geändert in:

IF L (bzw. K) = > M ...

Danach rechnete das Programm mit der erwarteten Genauigkeit. Letztlich schrieb ich die Druckausgabe für meinen FX-80 völlig um und fügte noch einiges hinzu. Nach meinem Geschmack müssen die eingegebenen Stützstellenwerte und die Form der Fouriergleichung mit ausgedrückt werden, nicht nur deren Koeffizienten. Letztlich müssen bei mir auch die Zahlen geordnet untereinander stehen. Die Epson-Interface-Karte hat dafür eine eingebaute Formatierungsroutine. Die Genauigkeit allerdings ist nur ab einem Grad der Gleichung von 12 wirklich zufriedenstellend, wenn man eine Funktion mit Sprungstellen, wie die im Belegartikel vorgeführte, betrachtet. Rechnet man nur bis zum 2. Grad, sind

die Ergebnisse schlichtweg falsch. Aber ab dem 4. Grad sind die Ergebnisse akzeptabel. Die Genauigkeit kann aber verbessert werden, wenn, wie beim Handrechnen üblich, die Stützwerte vor Beginn der Rechnung daraufhin untersucht werden, ob bei der Kurve gewisse Symmetrie-Eigenschaften vorhanden sind. Bei Vorhandensein können Cos-Terme oder Sin-Terme völlig ausgeschlossen werden. In der Beispielfunktion würden dann die jetzt erscheinenden Cos-Werte entfallen.  
*Rolf Sauer, Heidelberg*

## Grafik-Dump auf Ilc

Zuvor ein dickes Lob für Ihre Zeitschrift! Als Besitzer eines Apple Ilc stellt sich mir die Frage, wie man mit ihm Grafiken ausdrucken kann, obwohl man mangels Slots keine Grafik-Interfaces benutzen kann. Ferner bitte ich Sie, mir mitzuteilen, ob der Tandberg Data Tintenstrahldrucker beim Ilc zum Ausdruck von Grafiken benutzt werden kann.  
*Anton Bredonius, Pforzheim*

## Superdump für Seikosha?

Veröffentlichen Sie auch zum Beitrag Superdump Heft 6/85 die Modifizierung für den Seikosha GP-100A Mark II? Ein riesiges Lob auch an Ihre Aufmachung der Titelseiten. Ich finde, so etwas sollte es auch als Poster geben. Weiter so!  
*I. Kübler, Backnang*

## Lochrand

Beim Abheften von Listings in Plastik-Ringbinder geht der linke Teil der Zeilennummern verloren (Lochung!). Gibt es einen Poke, um das Listing 2 cm nach rechts zu schieben?  
*Heinz Specht, Bietigheim-Bissingen*

## Bit-Editor auch für Ile?

Mit großem Interesse habe ich den Artikel von Herrn Klant in Heft 7/85 gelesen. Die Verbesserung der Zeichenlesbarkeit ist enorm! Da sich mit dem Problem der guten Lesbarkeit der 80 Z/Z nicht nur die Besitzer von Videx-Karten herumschlagen, wäre es bestimmt eine gute Idee, dieses Programm auch für die Apple Ile-Karte (80 Z/Z + 64K) zu veröffentlichen (evtl. sogar mit der Liefermöglichkeit entsprechender EPROMs, da nur wenige Anwender diese EPROMs selbst brennen können).  
*Werner König, Hanau*

## Wie schnell ist der Mac?

Falls Sie zu diesem Artikel (Heft 5/85, S.43) noch keinen Berg von Leserbriefen erhalten haben, kann dies eigentlich nur daran liegen, daß der Fehler zu offensichtlich ist. Ohne groß darüber her-zuziehen, ist Ihnen ja hoffentlich klar, daß Herr Capitain nur die effektive Taktfrequenz gemessen hat. Bevor ich Ihnen das aber genauer erläutere, erkläre ich, daß ich weder mit der Firma Apple auf irgendeiner Weise verbunden bin noch einen Mac besitze, sondern mich als Student der Physik (im Rahmen meiner Diplomarbeit) mit der Beschaffung eines 68000-Systems beschäftige. Der 68000 hat die Eigenschaft, auch mit langsameren Speicherbausteinen zusammenzuarbeiten, indem er Waitzy-

klen einfügt. Nicht jeder Speicherbaustein schafft die notwendigen 125ns Schreib-/Lesezeit, um mit einer 8 MHz CPU zusammenzuarbeiten.

Untersuchen man die verwendeten Befehle zwischen E0E und E1E genauer (ohne die Piepser), so stellt man folgende Anzahl von Taktzyklen fest, die für Schreib- und Lesezugriffe benötigt werden:

Befehl – Takte – Speicherzugriffe  
MOVE.W #7F, D1 – 12 – 3/0  
MOVEQ #FF, D0 – 4 – 1/0  
MOVE.L (A7), (A7) – 20 – 3/2  
DBRA DO, E14 – 10 – 3/0 – (verzweigt)  
DBRA DO, E14 – 14 – 3/0 – (nicht verzweigt)  
CLR.W - (A7) – 14 – 2/1

Das ergibt eine Gesamtanzahl von  $32 + 30 * D0 * D1 + 18 * D1$  Takt, davon  $6 + 8 * D0 * D1 + 4 * D1$  Schreib- oder Lesezugriffe. (Wie Herr Capitain auf den Term von  $32 * D0 * D1$  kommt, ist mir ein Rätsel, erklärt aber nicht seinen Fehler. Außerdem wird der Befehl 51xx normalerweise nicht als DBRA, sondern als DBF übersetzt.)

Dies bedeutet eine effektive Taktfrequenz von 5,21 MHz. Man braucht nur 2 Waitzyklen einzulegen (ein Waitzyklus reicht nicht aus, denn 250ns-Bausteine sind auf dem freien Markt auch noch selten und dort muß Apple ja kaufen, da sie nicht selbst Speicherbausteine produzieren), um dann auf die gesuchte Taktfrequenz von 8 MHz zu gelangen.

Leider ist der Beweis nicht auf die schnelle zu führen, da fast alle Befehle das Verhältnis von 1 Schreib-/Lesezugriff auf 4 Taktzeiten haben. Unter Opferung von 6 Minuten kann der Beweis angetreten werden. Nach meiner Behauptung müßte die gleiche Schleife mit zehn MOVE.L (A7), (A7) 331,4s dauern, während eine Taktfrequenz von 5,21 MHz ohne Waitzyklen 340,2s benötigen würde. Diese 9s Unterschied müßten ausreichen, um Meßfehler auszuschließen. Denn eine Steigerung der Taktfrequenz um 0,1 MHz nach Belieben des Rechners ist mir noch nicht vorgekommen.

Wer dem Macintosh als (ersten billigen) „Grafik“-Rechner noch anlasten will, daß er zu langsame Speicherbausteine hat, sollte sich tatsächlich einen anderen Rechner zulegen, um dem Traum von Geschwindigkeit nachzukommen.  
*Georg Verweyen, Aachen*

## 1. Replik zum Mac-Brief

Zu den einzelnen von Ihnen angesprochenen Punkten möchte ich folgendes bemerken:

– Ob man den Befehl 51xx mit DBF oder mit DBRA übersetzt, ist Geschmackssache; beide Versionen werden vom Assembler akzeptiert. Sollte jemand die Form DBF bevorzugen, so kann er dies leicht durch entsprechende Änderung der Zeile 1842 erreichen.

– Der Term  $32 * D1 * D0$  in der Formel für die Zahl der Takte kommt dadurch zustande, daß in meinen Unterlagen manche Befehle andere Ausführungszeiten haben, als Sie sie angegeben

haben. So benötigt insbesondere der Befehl MOVE.L (A7), (A7) 22 Takte (siehe MC68000-Benutzerhandbuch, deutsche Ausgabe 1980). Welche Angabe richtig ist, weiß ich nicht.

– Die von Ihnen vorgeschlagene Schleife mit zehn Befehlen MOVE.L (A7), (A7) benötigt nach meinen Messungen 345,0s.

– Leider haben Sie die Absicht mißverstanden, die ich mit dem Beispielprogramm gehabt habe. Ich wollte keineswegs aus dem Mac einen langsamen Rechner machen. Vielmehr bin ich als Assembler-Programmierer, der seine Programme (fast) immer im RAM ablegt, daran interessiert, wie schnell diese Programme dann laufen. Und von diesem Standpunkt aus ist es mir wichtig, daß der Mac im RAM eine (effektive) Taktfrequenz von ca. 5-6 MHz und nicht 8 MHz erreicht. Und die für den Physiker sicher interessante Feststellung, daß der Mac 8 MHz schaffen könnte, wenn nur die Speicherbausteine schneller wären – diese Feststellung macht meine Programme dann aber auch nicht schneller.  
*Pit Capitain*

## 2. Replik zum Mac-Brief

Mein Lieblingsphilosoph, Nicolai Hartmann, schrieb dicke Wälzer über das Thema „Möglichkeit und Wirklichkeit“. Wenn ich mich mit meinem alten Golf GTI von der Hebebühne hochhieven lasse, den Motor starte und dann im fünften Gang das Gaspedal bis zum Anschlag durchtrete, gaule mir die Tacho-Nadel eine Geschwindigkeit von 220 km/h vor. Warum gibt VW dann nicht in der Werbung an, daß der GTI 220 „Sachen schafft“? Weil sich niemand ein Auto kauft, um auf einer Rampe zu „fahren“! Soweit die Werbung in der Autoindustrie. Betrachtet man demgegenüber die Werbung in der Computerindustrie, so wird man schnell gewahr, daß hier vielfach nur das Mögliche und nur ganz selten das Wirkliche herausgestellt wird. Einige Beispiele:

1. Takte/s: Wenn ich den Mac-68000-Processor in der Hand halte, kann ich von der Möglichkeit von 8 MHz sprechen. Steckt der 68000 jedoch im Macintosh, so sieht die Wirklichkeit ganz anders aus. Bereits vor anderthalb Jahren wurde in der „Byte“ konstatiert, daß das Video-RAM des Macintosh nur eine Taktfrequenz von ca. 5,5 MHz zuläßt.

2. Zeichen/s: Wenn man in den Epson-Handbüchern nachschlägt, findet man z.B. für den FX-80 eine Druckgeschwindigkeit von 160 Zeichen/s. Der bestmögliche Wert, den ich gemessen habe, beträgt 103 Zeichen/s. Unter welchen Umständen könnte die Möglichkeit der 160 Zeichen/s zur Wirklichkeit werden? Wenn ich die Zeit messe, die der Druckkopf benötigt, während er inmitten einer Textzeile, nachdem er bereits „auf vollen Touren“ ist, quasi „im freien Flug“ eine Leertaste überspringt, komme ich „hochgerechnet“ möglicherweise auf 160 Zeichen/s. Mit der Realität hat dies allerdings nichts gemein.

3. Bits/s: Bei Diskettencontrollern ist meist von „Megabits“ die Rede. Sind derartige Übertragungsraten realistisch? Wohl nur dann, wenn man sich das jeweilige Betriebssystem wegdenkt. Doch wer arbeitet schon mit einem Controller ohne DOS? Beispielsweise fragte mich nachträglich die Firma Frank und Britting, warum in dem Megacore-Bericht (Heft 7/85, S.77) von einer Übertragungsrate von nur 4000-15400 Bytes/s die Rede sei, wo doch die Megacore viele, viele Megabits/s übertragen würde. Was will jedoch der potentielle Käufer erfahren: einen physikalisch „richtigen“ Laborwert oder einen physikalisch „falschen“ Praxiswert? Was für die Autoindustrie gilt, sollte auch für die Computerindustrie gelten. Wer sich nicht an den theoretischen Physiker wendet, sollte die Geschwindigkeit niemals auf der Hebebühne, im Vakuum oder im freien Fall von der Zugschleife messen.  
*U.Stiehl*

### Peeker und Schule

Großes Lob für die reichhaltigen Beiträge, die auch Laien einiges geben können. Gibt es einen besseren Beweis als folgenden? Wir haben den Peeker an meiner Schule als Fachzeitschrift abonniert. Dennoch hat sich nach mir ein weiterer Kollege zum privaten Abo entschlossen, weil zum einen das Schulexemplar genutzt wird wie keine andere Zeitschrift sonst (und somit ständig „unterwegs“ ist) und es zum anderen doch etwas anderes ist, informatives Material ständig parat zu haben.  
 ...Völlig stiefmütterlich behandelt der Staat seine Lehrer, denn an Aus-, Fort- und Weiterbildungsmaßnahmen läuft äußerst wenig. Fast alles – insbesondere im Sekundarstufen-I-Bereich – muß(te) autodidaktisch erworben werden: Hier sehe ich das weite Feld für den Peeker, wobei Sie sicher darauf achten werden, daß der Peeker nicht zur Klammern der Pädagogen wird.  
*Hermann Döring, Liederbach*

## Inserentenverzeichnis peeker 9/85

aaa electronic gmbh, Freiburg	49
ACS, Detmold	29
Bühler Elektronik, Baden-Baden	9
ccp-datentechnik, Hamburg	23
J. Dahlhoff Computertechnik, Soest	68
D.O.S. Computersysteme, Schwäbisch Hall	29
Franzis-Verlag, München	35
Ingenieurbüro Fricke, Berlin	25
HIB, Nürnberg	35
IBS-Computertechnik, Bielefeld	44
Interkom electronic, Isernhagen	70
Intus, Waldshut-Tiengen	16
Jeschke, Kelkheim	70
KFC, Königstein	70
MCI GmbH, Bergisch Gladbach	70
E.-W. Meyer, Frohnhausen	23
Micromint Computer GmbH, Erkrath	29
U. Mohwinkel Electronic, Leverkusen	29
Pandasoft, Berlin	42
Softline, Oberkirch	29
Dipl.-Ing. R. Springmann, Hannover	29
Summagraphics, München	9
Sybex-Verlag GmbH, Düsseldorf	23
TEWI-Verlag, München	43
Tombstone-Micro, Berlin	70
Ueding electronics, Menden	35
Zechnersche Buchdruckerei, Speyer	47

## Vermischtes

### Paul Lutus im ZDF

Als ich mir am 18.07.85 im ZDF die Sendung „Aus Forschung und Technik“ ansah, traute ich meinen Augen kaum, als plötzlich Paul Lutus für wenige Sekunden auf der Matscheibe erschien. Der Schöpfer des „Applewriter“ lebt in einem Holzhaus im tiefsten Wald von Oregon mit einer riesigen Funkantenne im „Vorgarten“, die ihm den Kontakt zur Außenwelt ermöglicht. Die Waschbären scheinen also seinen Programmierdrang noch nicht gebremst zu haben. us

### Visicalc eingestellt

Man darf wohl mit Fug und Recht behaupten, daß der Apple II in der Gründerzeit erst durch „Visicalc“ kommerziell salonfähig wurde. Viele Kaufleute erwarben damals einen Apple II nur aus dem einen Grunde, um das berühmte Tabellenkalkulationsprogramm einzusetzen. Im Jahre 1979 von Software Arts alias Visicorp für den Apple konzipiert, wurde es bald für zahlreiche andere Rechner umgeschrieben und entwickelte sich zu einem der erfolgreichsten Mikrocomputerprogramme aller Zeiten. Als dann „Multiplan“ von Microsoft und „1-2-3“ von Lotus erschienen, ließen die Visicalc-Verkäufe merklich nach. Jetzt machte die Firma Lotus dem Longseller vollends den Garaus, indem sie zuerst Visicorp übernahm und dann prompt Visicalc aus dem Verkehr zog. us

### Pyramid Pitty als Hex-Dump?

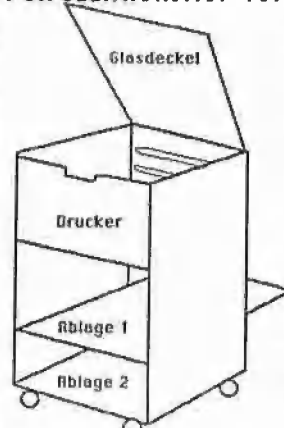
Zu dem in Heft 7/85 nur beschriebenen, aber nicht gelisteten Programm erhielt ich einige verbitterte Zuschriften von Peekerlesern, die darauf bestanden, daß zumindest der Hex-Dump hätte abgedruckt werden sollen. Offensichtlich war den Schreibern nicht bewußt, daß professionelle Computerspiele extrem umfangreich sind. So umfaßt der reine Hex-Dump aller Module von Pyramid Pitty ca. 20400 Bytes. Bei dreispaltigem Druck mit je 90 Zeilen zu je 8 Bytes pro Spalte würden sich insgesamt mindestens 9 3/4 Druckseiten im Peeker ergeben. Diese „Byte-Masse“ kann niemand fehlerfrei abschreiben. Und der Quellcode selbst hätte übrigens mehr als ein ganzes Peekerheft gefüllt. Ich hoffe, daß damit ausreichend begründet ist, warum das Listing des Programms nicht abgedruckt wurde. us

### Sammeldisk #10

Die Programme aus diesem Heft 9/85 finden sich auf der Sammeldisk #10, die in Verbindung mit Heft 10/85 ausgeliefert wird.

## Druckertisch

→ s. redaktioneller Teil



- \* 50% Schalldämmung
- \* Öffnungen f. Papier- und Steckerzufuhr
- \* 74x55x75 cm (HBT)

Preis: 298,- DM  
 bei Vorkasse Lieferung  
 frei Haus

Computertechnik  
 J. Dahlhoff  
 Högenstraße 1a  
 4770 Soest  
 Tel. 02921-12582

## Errata

### Fourier-Analyse

In Heft 6/85, S. 42 hat unsere Druckerei bei der Datenübertragung exakt 1024 Bytes "unterschlagen", und ich habe es beim Klebeumbuch nicht gemerkt... Nachfolgend der fehlende Abschnitt. Auf der Sammeldiskette #6 ist das Programm komplett.

```
4220 RETURN
4230 PRINT CHR$(12): PRINT "Im folgenden Katalog
der Disk sind die entsprechenden": PRINT :
PRINT "Files wie folgt gekennzeichnet:"
4240 PRINT
4250 HTAB 10: PRINT "xxxxxKOEFF beinhaltet die
Fourier-Koeffizienten einer bereits": PRINT:
HTAB 10: PRINT "analysierten Funktion."
4260 PRINT : HTAB 10: PRINT "xxxxxWERTE beinhaltet
eingegebene Funktionswerte."
4270 PRINT : INPUT "S1ot: "; S
4280 PRINT : INPUT "Drive: "; D
4290 PRINT D$;"CATALOG ,S";S";",D":D
4300 RETURN
4310 REM
4320 REM *** Eingabe mit der analytischen Funktion ***
4330 REM
4340 PRINT CHR$(12)
4345 LIST 4410: PRINT
4350 INPUT "Ist die zu analysierende Funktion
bereits in Zeile 4410 einprogrammiert? (J/N)": PF$
4360 IF PF$ = "J" THEN 4410
4370 PRINT
4380 PRINT "In Zeile 4150 ist nach Programmstopp
die zu berechnende Funktion": PRINT :
PRINT "einzuschreiben"
4390 PRINT : PRINT "Dies muß in der Form:
4410 DEF FN F(I)=... geschehen.": PRINT :
PRINT "Nach Eintragen der Funktion wird das
Programm mit RUN neu gestartet."
4400 STOP
```

### Formatter

Laut Pecker, Heft 7/85, S. 22, steht noch ein Patch aus, um dieses Formatierungsprogramm optimal für Disk-II- und Duodisk-Laufwerke einzustellen. Als Korrektur gibt der Autor, Arne Schäpers, an:

```
BLOAD FORMAT.OBJ
CALL -151
20PF6: 50 (statt bisher 30)
BSAVE FORMAT.OBJ, A$2000, L983
```

Man beachte, daß es sich bei dieser Korrektur des "Armbewegungswertes" nicht um einen Fehler im engeren Sinne handelt. Vielmehr erreicht man durch Erhöhung des Wertes von \$30 auf \$50, daß der DOS-Read-Write-Zugriff auf unterster RWTS-Ebene ausgeglichen ist. Der alte Wert \$30 bewirkt, daß der Sektor-Read sogar geringfügig schneller als normal ist (ca. 5%); dies kann bisweilen erwünscht sein. Besitzer anderer Laufwerke können mit anderen Werten im Bereich \$30-\$50 experimentieren, us

### PLOT 2.0

Im Programm PLOT 2.0 aus Pecker 5/85 sind im Listing (nicht auf der Sammeldiskette) folgende Tippfehler:

```
Zeile 3330 I = 0...
statt 3330 I = I
```

```
Zeile 3940 .... THEN 3910
statt 3940 .... THEN
```

```
Zeile 5400 .... GOSUB 2 ....
statt 5400 .... GOSUB 1010 ....
```

Ferner muß die vorletzte Zeile des EXEC-Files PLOT.PROTECTOR

```
POKE 50,255
statt POKE 50,225 lauten.
```

Die Sammeldiskette ist, wie gesagt, korrekt.

## Pressestimmen zur Wirtschaftlage von Apple

„Die Revolution\*  
hat gerade erst begonnen“  
(Apple-Werbeslogan 1985)

Die anhaltende Spekulation beruht auf der Überlegung, daß sich Apple, dieses Symbol der Auflehnung gegen konventionelle Umgangsformen, an einen starken Konzern anlehnen muß, um auch in den Bürocumputermarkt einzudringen. Das braucht keine Zusammenlegung zu sein. Ein Vertriebsabkommen mag genügen.

Im Gespräch sind die Giganten Atandt, General Electric, General Motors, Wang und Xerox. Sie dementieren durchweg. Eine „feindliche Übernahme“ gilt als unwahrscheinlich, weil Apple-Vorstandsmitglieder gute 20 Prozent des Aktienkapitals kontrollieren.  
*Handelsblatt vom 30.05.1985*

Die Schwäche auf dem Heimcomputermarkt zwang jetzt auch die Nr. 2 in den USA, Apple Computer Inc., zu einer

drastischen Reorganisation, in der sie ihre gewachsene Firmenstruktur über Bord warf und sich dem konventionellen Aufbau des Erzrivalen IBM anpaßte. Auf der Strecke blieb dabei Apple-Mitgründer Steven Jobs, ein 30jähriges Wunderkind, der zusammen mit seinem Freund den Markt für Personalcomputer erst kreierte. Insider gehen davon aus, daß Jobs seine Verantwortung für Entwicklung, Produktion und Verkauf des Macintosh erst aus der Hand legte, nachdem ihn der Aufsichtsrat davon überzeugen konnte, daß eine Beibehaltung der Firmenspaltung in einen Apple-II- und Macintosh-Flügel zur Existenzbedrohung werden könne. Nach der neuen Struktur gibt es nur noch eine Funktions- und keine Produktpaltung mehr. Jobs wurde zum Vorsitzenden mit „globalen Funktionen“ befördert, die Probleme bei Apple ließen den Kurs der Aktie gegenüber seiner Höchstnotierung um annähernd 75% fallen.  
*Handelsblatt vom 03.06.1985*

Aufgrund der weltweiten „konjunkturellen Schwächetendenz des Computermarktes“ wird die Apple Computer GmbH München die Zahl ihrer Mitarbeiter abbauen. Wie Geschäftsführer Ralph M. Deja mitteilte, wird die Belegschaft zum Oktober dieses Jahres um 25 auf 62 Mitarbeiter reduziert.  
*Stuttgarter Zeitung vom 29.06.1985*

Das akute Apple-Problem – die Wachstumsraten der Industrie und der „Lebensstandard“ des Herstellers (Managerschulung auf Syll!) passen nicht zusammen – wird durch die von Sculley verordnete Diät nicht entschärft. Zwar lacht das Herz des Apple-Kassierers, daß mit den Personal-Entlassungen die finanzpolitisch richtigen Entscheidungen getroffen wurden, doch bleibt die Frage der Marktpräsenz: Wie will die Apple-Rumpfmannschaft Stärke demonstrieren und – was viel wichtiger scheint – ihren Kunden den nötigen Support bieten? Beispiel: Es erscheint fast als eine Anmaßung, im „Büro“ sein

zu wollen, wie Sculley vorgibt, wenn die Produkte („Mac-Office“) noch nicht da sind. Wie will man so, etwa gegenüber gestandenen Großunternehmen, Seriosität als Computerhersteller auf die Beine bringen? Es geht hier immerhin gegen die IBM, die ihre Marktmacht bei den „Large Accounts“ über Spezialisten in den zentralen DV-Shops ausübt. Noch ist nicht zu sehen, daß Apple vom hohen Roß heruntersteigen will. So macht beispielsweise Deutschland-Chef Ralph Deja die schlechte Konjunktur im Computermarkt für die Apple-Schwäche verantwortlich. Diese Aussage ist, was die Bundesrepublik betrifft, schlichtweg falsch. Once again: Wie will Apple Seriosität auf die Beine bringen? Von der Beantwortung dieser Frage hängt es ab, wie heuer die „Apfel“-Ernte ausfällt.

*Computerwoche vom 05.07.1985*

\* Revolution kommt von lat. revolvare = zurückdrehen

## Für Apple II, IIE

<b>Z-80-Karte</b>	69,-	<b>80-Zeichen-Karte</b>	139,-
<b>Disk-Interface</b>	68,-	mit Sorfswitch, neue Vers. m. gest. scharf. Bild	
<b>Centronics-Interf. m. Kabel</b>	69,-	<b>Speech-Karte</b>	55,-
<b>16-K-RAM-Karte</b>	69,-	<b>Clock-Karte</b>	129,-
<b>RS-232-Karte</b>	109,-	<b>Super-Serial-Karte</b>	199,-
<b>EPROMmer (4, 8, 16 K)</b>	139,-	<b>Komp 2E</b>	797,-
<b>128-K-RAM-Karte</b>	279,-	Apple 2E kompatibel, Rechner 64 K im 2E-Design, ohne Firmware	
<b>256-KB-RAM-Karte</b>	448,-	<b>80Z + 64K-Karte</b>	99,-
<b>Wild-Karte</b>	69,-	für 2E kompatibel	
(knackt geschützte Programme)		<b>Apple-Info 1,- DM (Porto)</b>	
<b>Händleranfragen erwünscht!</b>			

### Apple II + Kompatibel

**Komp 48**  
48 K, 6502 ohne Firmware 630,-

**Komp 64**  
64 K, 6502, Z-80, 15er-Block  
ohne Firmware 840,-

**Komp 64 S**  
wie Komp 64, jedoch mit abgesetzter  
Tastatur mit 188 Funktionen. 940,-

**Motherboard 48 K**  
8 Slots, alle IC's gesockelt,  
ohne Firmware, fertig geprüft 369,-

**Motherboard 64 K**  
Wie oben, mit 6502 und  
Z 80, 64 K 379,-



**SUPER PREISE**

**Klaus Jeschke**  
Hard-, Software  
Viertstr. 3-13  
6233 Kelkheim  
☎ (061 98) 75 23

Alle Preise inklusive Mehrwertsteuer, 6 Monate Garantie. Versand erfolgt per NN oder Vorkasse.

### APPLE-INTERFACE

Weilgerhard deutsche Produktion

MESON IH-C, 48k, Apple-Kompatibel	ab 1000,-
Z80, 18k	85,-/95,-
PAL, EPROM, CENTRONICS, RS 232	je 170,-
80 Zeichen Sorfswitch	200,-
ADD 2B, V/A-Card (6522) mit RAM u. Backup	170,-
ADD 4B, P/A-Card (6821) mit RAM u. Backup	150,-
ADD 5, Optokopier-Card 8 Bit	180,-
JOYPORT zum Anschluss von zwei Atari Joystick	40,-
TEAC PD 55 F	520,-
Floppy-Controller APCD, Anpaßsch.	300,-
LOD 103, 40 Track Apple-Kompatible Slim	420,-
Preise pro Stück, inkl. MwSt.	
Ladenverkauf: Mo. 7-18 Uhr, Di. - Do. 15-18 Uhr, Sa. 10-13 Uhr	
Telefon 030-83313103 (wie Ladenverkauf)	

### KLEINTEILE

Standard 8, 1, 1982	1-9
UPD 4116-2000	3,30
M64 4116 2000	3,50
W64 4116 2000	3,80
TMM 41256 C15	17,-
6118 LP3 (SRM 2016)	10,50
2708-4800	13,30
2716-4800	13,50
2725-4800	14,-
2734-4800	14,-
2743-4800	14,-
2752-4800	14,-
2761-4800	14,-
2770-4800	14,-
2779-4800	14,-
2788-4800	14,-
2797-4800	14,-
2806-4800	14,-
2815-4800	14,-
2824-4800	14,-
2833-4800	14,-
2842-4800	14,-
2851-4800	14,-
2860-4800	14,-
2869-4800	14,-
2878-4800	14,-
2887-4800	14,-
2896-4800	14,-
2905-4800	14,-
2914-4800	14,-
2923-4800	14,-
2932-4800	14,-
2941-4800	14,-
2950-4800	14,-
2959-4800	14,-
2968-4800	14,-
2977-4800	14,-
2986-4800	14,-
2995-4800	14,-
3004-4800	14,-
3013-4800	14,-
3022-4800	14,-
3031-4800	14,-
3040-4800	14,-
3049-4800	14,-
3058-4800	14,-
3067-4800	14,-
3076-4800	14,-
3085-4800	14,-
3094-4800	14,-
3103-4800	14,-
3112-4800	14,-
3121-4800	14,-
3130-4800	14,-
3139-4800	14,-
3148-4800	14,-
3157-4800	14,-
3166-4800	14,-
3175-4800	14,-
3184-4800	14,-
3193-4800	14,-
3202-4800	14,-
3211-4800	14,-
3220-4800	14,-
3229-4800	14,-
3238-4800	14,-
3247-4800	14,-
3256-4800	14,-
3265-4800	14,-
3274-4800	14,-
3283-4800	14,-
3292-4800	14,-
3301-4800	14,-
3310-4800	14,-
3319-4800	14,-
3328-4800	14,-
3337-4800	14,-
3346-4800	14,-
3355-4800	14,-
3364-4800	14,-
3373-4800	14,-
3382-4800	14,-
3391-4800	14,-
3400-4800	14,-

Wichtige Beispiele und Komponenten: A. Minisystemer 50,-  
DM, ser. +10,- DM; Foto und Verastaltung 8.550 9.5,- DM  
Verastaltung per NN oder VK, bei NN +1,50 DM

T. Tank & G. Köber · Gardeschützenweg 72 · 1000 Berlin 45

Apple
KFC
Computer
KÖNIGSTEIN

Da guckt jeder, bei den Neutigkeiten!



**Mac 512 Kb** ..... 498,-

Apple IIc mit 2 Laufwerken + Contr. + Drucker (ILO, 128 Zechn, 2Hb Puffer) + Intf. + Monitor 15 MHz 12" barnst./grün.....298,-

Monitor 28 MHz 12" ..... 348,-

**Erweiterte 88 Zeichenkarte (IIc)**.....198,-

Apple IIc + zweites Laufwerk..... 298,-

Apple IIc + zweites Laufwerk + Drucker (ILO, 128 Zechn, 2Hb Puffer) + Intf. .... 398,-

**Neu: Mac Video Adapter Karte mit 4096 RAM**  
kompatibel mit IIc/IIe + Mac 160 Zechn/8 sehr klein 80\*275\*360, Puffer 8Kb opt. 16 Mb Papierzufuhr von hinten/unten (Ethernet) hervorragende Druckqualität.....

Neu Mac-Video ein Videointerface für den Mac Mac Speichererweiterung auf 512 Kb ... 1

Mac Software: Ensemble, Jazz, Omnix 3, Word, File Sorgen, Musicworks, Multipian, EXCEL, Mac Doc, Adress, Lohn, Fibu, Hand, Mac, Gagne, 1 Base, Terminer, Neu: Mac-Tablett das Originaltablett für den Macintosh. Brauchen Sie mehr Speicherplatz oder wollen Sie mehrere Mac's zusammenschließen holen Sie unser Festplattenangebot ein. Ständig Vorrührgüter zu Sonderpreisen !!

**zB: Monitor mit 5känder für Apple IIc ..... 598,-**

**Logistik II/e 39,- Funktionsrechnerkonverter 499,-**  
Rechnerkonverter mit Fernabfrage 0,172 nur.....399,-

**Schulpreise & Anträge**

**Disketten Disketten Papier Geräte - Software Taxon Monitore**

**BTX mit IIc/IIe Vertretungen: Brother Star 041 CORPUS Olympia**

**HEC Summagraphik Datatabe usw. Neu 041 Farldrucker. 893,-**  
die aktuellsten Preise im Mailboxservice: 06174 / 5355

**KFC Wissenzstr. 16 6240 KÖNIGSTEIN Tel. 06174 21953 (3033) BTX\*921069**

## Ausgabe und Eingabe mit TYPETERM®

im Slot Ihres  
**APPLE II/Ile**

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

TYPETERM- Interface für alle BROTHER-Typenrad- schreibmaschinen	DM 479,- incl. MWSt.
Paketpreis: Schreibmaschine CE-51 mit TYPETERM	DM 1348,-



## Erscheinungs- und Anzeigenschlußtermine für peeker

Ausgabe	Erscheinungs- termin	Anzeigenschluß
10	23. 9. 85	23. 8. 85
11	21. 10. 85	20. 9. 85
12	25. 11. 85	25. 10. 85

Die Artikel der Firma Interdata GmbH, Singen, sind auch über die Firma Beltronic, B. Ledergerber, Im vorderen Chapf, CH-Rüdingen

**Tel.: 01/8673141, zu beziehen.**

### brother

QUALITÄT AUS ERSTER HAND.

CE-61 mit TYPETERM .....	DM 1787,-
EM-80 mit TYPETERM .....	DM 2087,-
EM-100 mit TYPETERM .....	DM 3122,-
TYPETERM-Kit für CE-50 ....	DM 468,-
CE-25 mit TYPETERM .....	anfragen

TYPETERM - ein starkes Interface für starke Maschinen! Alle Cursor- und Cit-Befehle, 2k ROM auf der Karte f. DOS, PRODOS, CP/M, PASCAL. Alle Features: Hoch-/Tiefstellen, autom. Unterstreichen, var. Zeichen- u. Zeilenabst., autom. Papierführung usw. Ausführl. Handbuch vorab: 10,- DM auf Konto 14770-306 PGIroA Han (Anrechnung). TYPETERM ein Produkt von



Kock & Mreches GmbH  
Postf. 3004 Isernhagen 4  
Telefon 05139-87993

### XT16PC

16 Bit 8088 Rechner  
vollständig Kompatibel,  
ausgerüstet mit:  
Colorcard,  
256 RAM  
Floppy Controller,  
1 Drive, 360 K,  
deutsche Tastatur,  
130 W Netzteil

**MCI XT16PC**

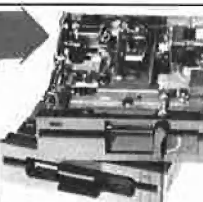


**2999 DM**

**XT 16 PC**  
mit 6 MByte Winchester

**4499 DM**

### FLOPPY-DRIVES



**3,5" Pana. JU 363** 479 DM  
135 tpi 2 x 80 track 1 MByte

**TEAC FD55B** 379 DM  
48 tpi 2 x 40 track 500 KByte

**TEAC FD55F** 448 DM  
96 tpi 2 x 80 track 1 MByte

**SANYO BFT 540** 399 DM  
48 tpi 2 x 40 track 500 KByte

**SANYO BFT 580** 449 DM  
96 tpi 2 x 80 track 1 MByte

**CHINON F-502** 399 DM  
48 tpi 2 x 40 track 500 KByte

**REMAX RFD 480** 329 DM  
48 tpi 2 x 40 track 500 KByte

**Apple AFD55A** 399 DM  
mit Kabel und Gehäuse

**STATIONEN**

2x 8" SDDD KIT 1399 DM  
2x 8" DDDD KIT 1999 DM

**FLÜSTER-FLOPPY**  
CHINON F-051 A  
mit Kabel 379 DM

**Billigversion!!!**  
CHINON A-II  
mit Kabel und Gehäuse 345 DM

### ESPRINT 80/ 801



- MX 80 + Grafrax Compatible
- 100Zeichen/9x11 Matrix, 6Zeichensätze
- Parallel + RS232, Einzelblatt u. Traktor

für IBM angepaßt **749 DM** **699 DM**



G Postfach 20 04 01  
M J.-W.-Lindlar-Straße 6  
B 5060 Berglich Gladbach 2  
H Tel. (022 02) 3 10 07 · Tx. 8 873 518

Auf alle Geräte 6 Monate Garantie · Änderungen, die technischen Verbesserungen dienen, vorbehalten - Lieferbedingungen auf Anfrage · Lieferung solange Vorrat reicht -



# 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, ca. 500 Seiten, Softcover, DM 66,-

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

## Weiterführende Literatur...



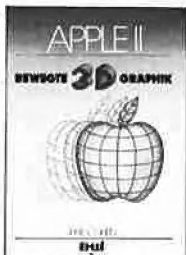
### Reparaturanleitung Computer: **Apple II, IIplus**

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 USA. Für die Computer APPLE II, C-64, IBM PC, ATARI bis 520 ST, TI-99 und CPC 464/664. Hochwertiges Textbuch für Logo-Kurse für zu Hause und im Lehrbereich. 384 Seiten. A4, DM 59,-



### APPLE II - Bewegte 3D-Graphik

(Phil Cohen)  
Selbstentworfenen 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,-



### Computer für Kinder

(Sally Greenwood Larson)  
Ein Buch für Kinder, ihre Lehrer und Eltern.

„Computer für Kinder“ richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern dieses Buch bewusst geschrieben wurde. Unterhaltsam

und leicht verständlich. Ein Handbuch für Beginner. A4 quer. Fadenheftung. DM 29,80

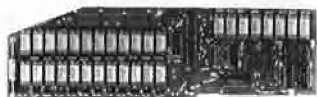


**Erstes deutsches Referenzwerk** sämtlicher Befehle und Systemroutinen von Apple II, IIplus, IIe  
**APPLE II PASCAL** Betriebssystem, 272 S., DM 49,-  
Sprache, 216 S., DM 39,-  
Pascal 1.2 Addendum, 112 S., DM 36,-

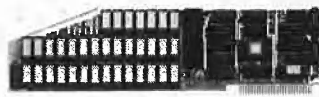
Grundlagenbuch, Bestseller  
APPLE II PASCAL,  
Eine praktische Anleitung,  
544 S., DM 59,-

Noch im Programm:  
6502 – Programmieren in Assembler DM 59,-  
VisiCalc, 50 Programme auf Diskette, DM 79,-

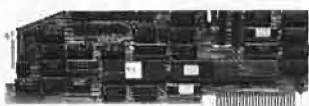
In Vorbereitung:  
Macintosh Programmier-Handbuch  
mit Microsoft BASIC 2.0 DM 59,-



AP 13 und AP 17  
RAM-Karten zum Einsatz als Pseudodisk unter CP/M, USCD und APPLE-DOS. Speichergröße von 64 kByte bis 256 kByte.  
Bestell-Nr.: A 1013 a-b  
A 1017 a-d



AP 33  
RAMDISK der neuen Generation. Für besonders speicherintensive Arbeiten ist der Ausbau in Stufen von 64 kByte bis 1MByte möglich.  
Bestell-Nr. A 1033



AP 14  
Floppy-Controller für alle Anwendungsfälle. 10 Laufwerke können gleichzeitig angeschlossen werden. 4 x 8" DSDD, 4 x 5 1/4" DSDD und zwei Apple-Standardlaufwerke. Maximal ca. 10MByte im Direktzugriff.  
Bestell-Nr.: A 1014



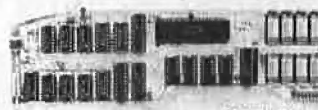
**NEU!** jetzt 512 k-RAM

AP 20  
INTEMEX mit 68 000 CPU und 128 k-RAM. Diese Karte macht aus Ihrem Rechner mit „Applebus“ einen echten 16 bit-Rechner. Eine Zusatzkarte (AP 26) ermöglicht einen Arbeitsspeicher bis zu einem MByte und an Software gibt es einiges. Z.B. stehen drei Betriebssysteme und die wichtigsten Hochsprachen zur Verfügung.  
Bestell-Nr. A 1020

Interfaces  
für  
Computer  
mit  
Applebus  
IBS COMPUTERTECHNIK



AP 19  
12-Kanal AD-DA-Wandler mit 12 bit Auflösung und 25 µ sec Wandlungszeit. Eingangsspannung ±10 V. Ein schneller Wandler für extrem schnelle Anwendungen.  
Bestell-Nr.: A 1019



**NEU!** 8 MHz Takt

AP 22  
INTEMEX mit Z 80 B-CPU und 64 k-RAM. Wenn Sie einmal diese Karte in Aktion gesehen haben, werden Sie auch feststellen: „Geschwindigkeit ist keine Hexerei, man braucht nur die AP 22“. Mit dieser Karte wird Ihr APPLE II zum z.Z. schnellsten CP/M-Computer, und in Verbindung mit dem SPACE 84 erhalten Sie Computerleistung, die wirklich einmalig ist. Wir vermitteln gerne eine Vorführung.  
Bestell-Nr. A 1022

**NEU!!!**

Das Interface-Buch von IBS, ein Buch für Alle, die Ihren APPLE II oder Kompatiblen optimal nutzen wollen. Detaillierte Schaltpläne, Bauteilelisten und Benutzungshinweise zu allen IBS-Interfaces finden Sie jetzt in einem Buch vereint. Ausführliche Abhandlungen über Spezienschaltungen, über Anwendungsmöglichkeiten, über neue Softwarewelten aber auch über die Grenzen des APPLE II-Systems bestimmen den Wert dieses Buches.

Für nur DM 8,00 erhalten Sie dieses Buch ab sofort bei Ihrem Computerfachhändler oder für DM 8,00 + DM 2,00 Versandkosten bei IBS COMPUTERVERTRIEB.

**IBS**  
**COMPUTERTECHNIK**

Olper Straße 10 · 4800 Bielefeld 14 · Tel.: 0521/444032 · W. Germany  
1011 Rose Marie Lane 16 · Stockton CA 95207 · Tel. 209/473-7473 USA