Nr. 3/85 März
DM 6,50, sfr 6,50, öS 50, Lit 5900, hfl 7,50

# peeker

## MAGAZIN FÜR APPLE-COMPUTER

**Apple für Ingenieure**

**Technik der Disk-Drives**

**Legasthenie und Computer**

**CP/M-Konverter**

**Pascal 1.2**

**Apple-Spiele**

**Masterclock**


**Mac im Test**

Hüthig
PUBLIKATION

# PRO METRIC® IST DA!

| | |
|---|---|
| **B2-Motherboard mit 6502, 65C02C, Z80B, V24, Parallel-Schnittstelle, 80-Zeichenkarte, RGB-Ausgang, 192 KB RAM** | **DM 2299,––** |
| **B3-Motherboard, wie B2, zusätzlich 256 KB Pseudofloppy** | **DM 2899,––** |
| **Prometric® B2 Tischgerät** | **ab DM 3349,––** |
| **Prometric® B3 Tischgerät** | **ab DM 3999,––** |
| **Prometric® Portable B2, Monitor 9"** | **ab DM 4449,––** |
| **Prometric® Portable B3, Monitor 9"** | **ab DM 4999,––** |

**Neu von DRV:** C-DOS, das 80-Track DOS für B2 und Kompatible, verarbeitet Integer, DOS 3.3 und Prodos-Files DM 149,––
DEUTSCHES BASIC, ROM-resident, übersetzt Ihre Apple-Soft Files ins Deutsche und umgekehrt DM 149,––
CALVADOS, deutsche Textverarbeitung DM 299,––

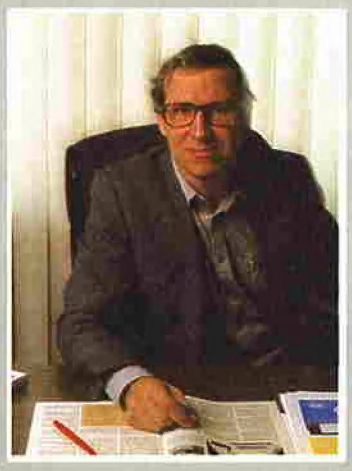
| **TEAC FD55F** | **DM 674,––** |
|---|---|
| TEAC FD 55 B | DM 599,–– |
| FDC4-Controler | DM 179,–– |
| Shugart-Bus-Kabel | DM 45,–– |
| PREH-Commander AK87 | DM 329,–– |
| MONITOR 12", 22 MHZ | DM 349,–– |
| M100-DRUCKER | DM 825,–– |
| Graphik-Par.-Interface | DM 125,–– |
| OPERATOR 2-Tastatur | DM 580,–– |
| Super COM-Tastatur (deutsche Belegung) | DM 369,–– |
| Z 80 B-Karte 6 MHz | DM 899,–– |
| 80-Zeichenkarte 4 Zeichensätze | DM 279,–– |

| **YE-DATA 480** | **DM 694,––** |
|---|---|
| ERPHI-Controler | DM 298,–– |
| SUPER-Controler | DM 510,–– |
| IBM-look Gehäuse | DM 218,–– |
| Monitor 15" | DM 528,–– |
| 32K-PRINTER-BUFFER | DM 389,–– |
| MOTHERBOARD 64 KB + Z80 | DM 699,–– |
| PSEUDO-FLOPPY 256 KB | DM 899,–– |
| 80-Track-Patch DP/m 3.0 | DM 75,–– |
| TIMER PLATINE | DM 39,–– |
| STEP Impuls Verdoppler | DM 39,–– |
| Schaltnetzteil 7 Ampere (VDI gepr.) | DM 269,–– |

**NEU:** 15 MB Winchester Glimline inkl. Controler DM 3999,––
Preh Commander COMM Starlight-1 21 Funktionstasten DM 359,––

Alle Preise inklusive 14% MwSt. Weiteres Zubehör auf Anfrage.

**E. Böhmer, DRV, Am Kellersbusch, 6072 Dreieich, 0 61 03 / 8 46 47**



# EDITORIAL

Jetzt ist es also „amtlich": Nachdem im Herbst des letzten Jahres bereits der Apple III eingestellt wurde, ist nunmehr bekannt, daß die Lisa nicht mehr produziert werden wird. Wir erinnern uns: Die Lisa wurde vor knapp zwei Jahren anläßlich der Hannover-Messe als Computer sui generis vorgestellt. Wenn man in den alten Computer-Zeitschriften nachliest, so ist man heute peinlich berührt von der Kluft zwischen Werbung und Wirklichkeit. Dem perennierenden Universalcomputer war nur ein ephemeres Dasein beschieden. Quo vadis malum? Wie geht es jetzt bei Apple weiter? Der Apple III war über 3 Jahre erhältlich, die Lisa bereits nur noch 2 Jahre. Wird sich dieser Trend zur Kurzlebigkeit beim Macintosh fortsetzen?

Werfen wir einen Blick auf den altbewährten und von uns hochgeschätzten Apple II in seinen verschiedenen Varianten (II+, IIe, IIc). Obgleich dieses Gerät im technologischen Vergleich zum Macintosh und zur Lisa altertümlich anmutet, hat es doch zahlreiche und zugleich entscheidende Vorteile: Es ist ein offenes System („Open System"), hardwaremäßig erweiterbar, bestens erforscht und verfügt über sehr viele Spezial- und Hilfsprogramme, die einem das Arbeiten mit dem Gerät erleichtern. Demgegenüber sind Apple III, Lisa und Macintosh geschlossene Systeme („Closed System"), bei denen von Apple viel Geheimniskrämerei betrieben wird mit der Folge, daß weder Softwarehäuser noch „Freaks" hierfür in nennenswertem Umfang Anwenderprogramme und Utilities entwickelt haben.

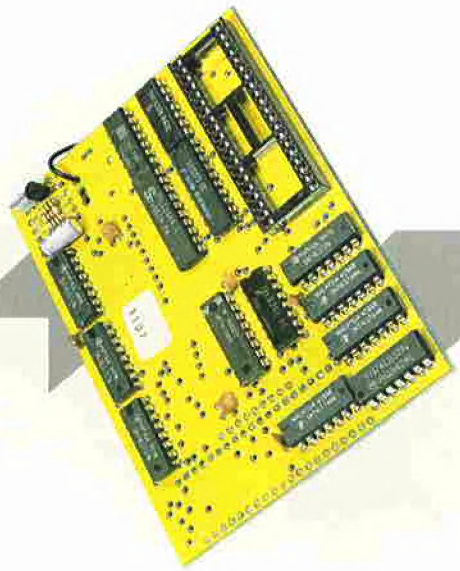
Beispiel 1: Heute erreicht mich eine Pressemitteilung zum Apple III, die ein neues „Hilfsprogramm der Superlative" ankündigt, womit *erstmals* gelöschte Dateien UNDELETEt werden können. Abgesehen davon, daß jeder DOS-Kenner eine solche Utility ohne Mühe schreiben könnte, fällt das Wort „erstmals" in der Presse-Info auf. Was beim Apple II quasi ein alter Hut ist, wird beim Apple III *nach* der Produktionseinstellung als die Sensation par excellence angepriesen.

Beispiel 2: Auf der Frankfurter Micro-Computer '85 nahm ich mir den Macintosh-Prospekt „Programme über Programme" mit. Darin heißt es: „Auch für den Macintosh sind *Tausende* von Programmen in Vorbereitung. Über einhundert finden Sie in dieser Übersicht". Ich zählte dann einmal die mit „deutsch" gekennzeichneten Programme durch. Es waren genau *sieben.* Hier liegt eine eklatante Diskrepanz zwischen Wunschvorstellung und Realität vor. Anfang Dezember 1984 hatte ich bei der Firma Apple in München technische Unterlagen über den Macintosh angefordert. Ich warte noch heute. Es wäre gut, wenn sich die Firma Apple wieder zu Ihrer früheren Einsicht bekennt, daß die Verbreitung eines Microcomputers mit dessen Erforschung korreliert, denn der Apple II wird zwar noch lange, aber gewiß nicht ewig leben.

So setzen wir denn alle Hoffnungen auf den in den USA bereits unter Bezeichnungen wie „Apple IIx" u.ä. angekündigten 16-Bit-Rechner, der die Vorteile der Apple-II-Linie bewahren und die Nachteile der „32"-Bit-Linie vermeiden soll.

**Ulrich Stiehl**

Hüthig
PUBLIKATION



# PEEKER



# INHALT

**3/85**

**Impressum**

Peeker
Magazin für Apple-Computer
2. Jahrgang 1985
ISSN 0176–9200


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

# PEEKER

## MAGAZIN FÜR APPLE-COMPUTER


Editorial . . . . . 3
Impressum . . . . . 4
Produkte . . . . . 65
– Pascal 1.2 . . . . . 65
– Double-Hires-Grafik-Programm . . 70
– Hardbreaker and Softbreaker . . 72
– Beliebte Apple-Spiele . . . . . 73
– Memdos Junior . . . . . 75
– CP/M-Karte für den Apple IIc . . 75
– HOCO-Masterclock . . . . . 76
Leserbriefe . . . . . 77

**6 Der Apple II als persönliches Ingenieurwerkzeug**

**10 Aufbau und Funktion von Diskettensystemen**

**22 Testgenerator für Legastheniker**
Ein Applesoft-Programm für den Schuleinsatz

**30 Höhere Präzision bei den vier Grundrechenarten**

**35 Wordstar-Transfer-Refiner**
Konvertierung von CP/M-Wordstar- in DOS-Applewriter-Textfiles

**43 GETCPM**
Konvertierung von CP/M- in DOS-Textfiles

**44 Die Maske fällt**
Der Macintosh im Leistungstest

**56 Geschwindigkeit ist keine Hexerei**
Eratosthenes erneut betrachtet



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

Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement DM 58,–, einschließlich MwSt,
im Inland portofrei. Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089/319 10 67, Telex 0 522 656

**Zahlungen:** an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: **Postscheckkonten:** BRD: Karlsruhe 485 45-753; Österreich: Wien 7555888; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 47718; Belgien: Brüssel 7230 26; Dänemark: Kopenhagen 349 69; Norwegen: Oslo 994 24; Schweden: Stockholm 5477 76-5

**Bankkonten:** Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 0 2165 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

# Der Apple II als persönliches Ingenieurwerkzeug

**von Dipl.-Ing. Bernd Worms**

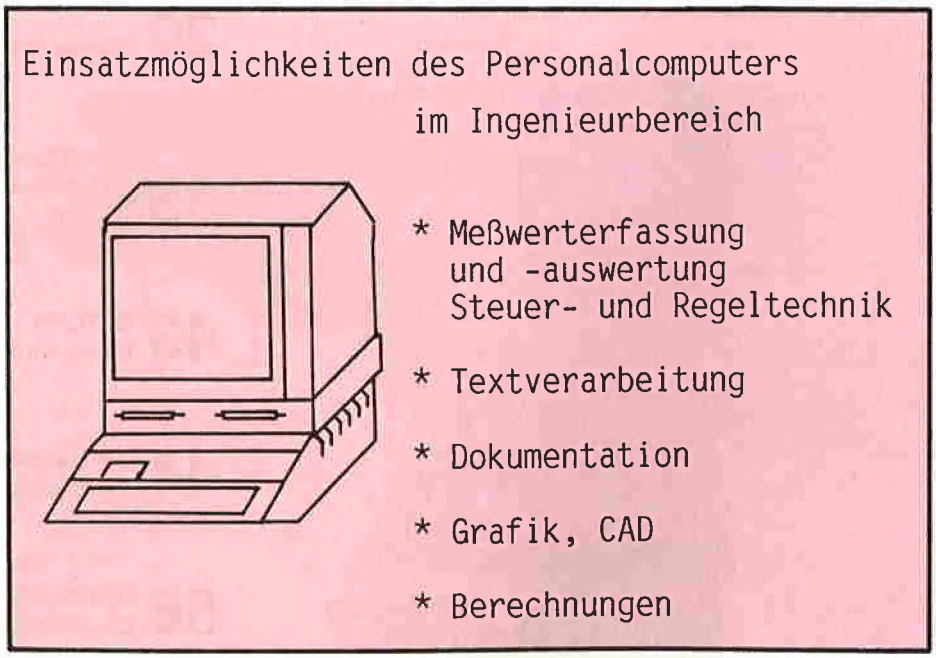



*Bild 1: Einsatzmöglichkeiten eines Personalcomputers im Ingenieurbereich*

Der Personalcomputer hat in den letzten Jahren einen wahrhaften Siegeszug im Bereich der Bürotechnik angetreten. Er hat das Gesicht eines modernen Büros entscheidend verändert. Auch in anderen Bereichen hält er Einzug. Langfristig wird er wohl an vielen Arbeitsplätzen einen festen Platz erobern, so auch im Ingenieurbüro. Ein aufgerüsteter Personalcomputer wird in Zukunft in der Hand eines Technikers für vielerlei Meßaufgaben eingesetzt werden, für die man bisher mehrere, unterschiedliche Meßgeräte benötigte. Der Ingenieur kann den Rechner zu einem individuellen und universellen Werkzeug ausbauen, das ihm erlaubt, seine Arbeitszeit effektiv zu gestalten.

Bedingt durch die Slot-Architektur (insgesamt 7 oder 8 freie Steckkartenplätze) ist der Apple II/II Plus/IIe (nicht aber der IIc

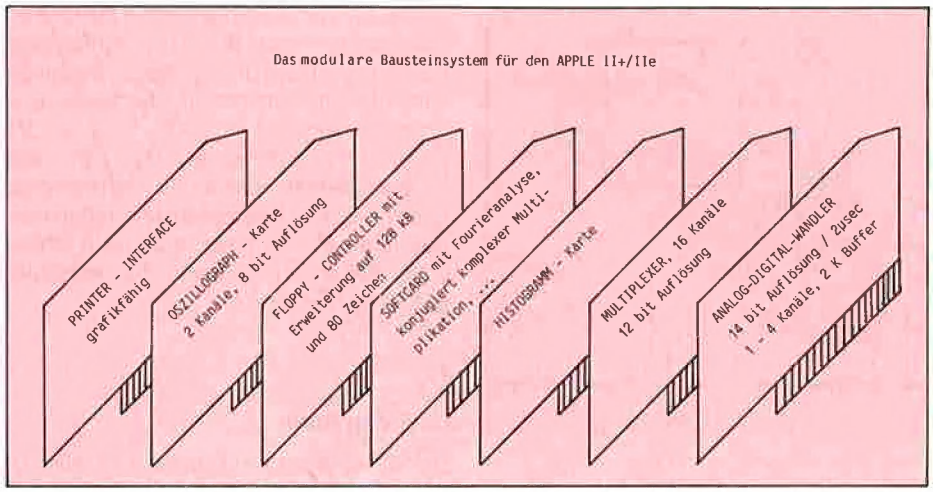



*Bild 2: Steckkartenkonzept für den Apple II*

wegen der fehlenden Slots) der am vielseitigsten einsetzbare Personalcomputer. Damit ist er für die in Ingenieurbüros und Entwicklungslabors anfallenden Aufgaben besonders gut geeignet. Gleichzeitig ermöglicht dieses Slot-Konzept, mit der technischen Entwicklung über einen längeren Zeitraum Schritt zu halten, ohne sich regelmäßig einen neuen Computer anzuschaffen.

Die typischen Aufgabenstellungen in Ingenieur- und Entwicklungsabteilungen reichen von der Erfassung und Echtzeitauswertung von Signalen in dynamischen Prozessen (insbesondere in der Regeltechnik) bis hin zu Langzeituntersuchungen von Meßsignalen aus der Umwelt. Hinzu kommen Aufgaben wie die Dokumentation von Arbeiten, die Kommunikation mit Kollegen und externen Stellen, die Erstellung von Zeichnungen und grafischen Darstellungen sowie das Durchführen komplexer Berechnungen. Für all diese Tätigkeiten kann der Personalcomputer unterstützend eingesetzt werden.

Für die letztgenannten Aufgaben gibt es schon zahlreiche Software, aber auf den Gebieten der Meß- und Regeltechnik, bei denen der Mikrocomputer die Aufgabe der Bedienungs- und Steuereinheit erfüllt, ist der Markt in Deutschland noch überschaubar. Der Einsatz des Personalcomputers vollzieht sich hier langsam, und dies trotz seiner hohen Anpassungsfähigkeit an ständig wechselnde Aufgabenstellungen, wie sie im Forschungslabor und industriellen Prüffeld anfallen.

*Bild 3: Schwingungsmessung*

Für den Apple II Plus bzw.IIe existiert ein modulares Bausteinsystem, das durch das flexible Apple-Konzept ermöglicht wird. Die Elemente dieses Systems bestehen aus

– Hardware zur Meßwertaufnahme,
– Firmware zur Datenanalyse und
– Hard-/Software zur Datendarstellung

Damit gelingt es, den Apple kostengünstig zu einem signalanalytischen Universalgerät aufzurüsten.

Eine sehr interessante Steckkarte ist die Softcard der Firma Wintex Instruments, mit der typische signalanalytische Aufgaben wie z.B. Spektralanalyse (FFT) , Datenblock-Multiplikation, -Differentiation und -Integration erledigt werden können. Hervorzuheben ist, daß hierbei der Standard-Adressenraum des Apples nicht verkleinert wird. Der Anwender besitzt zusätzlich noch die Freiheit, sich eigene Routinen in Form von Zusatzbefehlen zu definieren und sie mittels eines gebrannten EPROMs zu ergänzen.

Dieses modulare Ausbaukonzept ermöglicht verschiedenartige Meßaufgaben. Ob nun bei der A/D-Wandlung 8-Bit- oder 14-Bit-Auflösung gewünscht, ob nun 1–16 Kanäle „gemultiplext" oder ob Daten über 1–4 Kanäle mit unterschiedlichen Zeitbasen aufgenommen werden, dies alles läßt das Konzept zu. Zur Auswertung der Meßdaten können Frequenz-, Korrelations-, Trend- und/oder Histogrammanalysen ebenso wie „Daten-Smoothing" und „Signal-Averaging" durchgeführt werden.

Ein etwas anderes Konzept für den meßtechnischen Ausbau von Personalcomputern sieht die Zusammenfassung der verschiedenen Schnittstellen zu den einzelnen Peripheriegeräten in einer Interfacebox vor, die mit Controller, unabhängiger Stromversorgung und Busplatine ausgerüstet ist. Controller und Rechner können – falls erforderlich – durch eine Optokopplerkarte galvanisch getrennt werden. Für die Verbindung von Mikrocomputer und Interfacebox wird bei paralleler An-

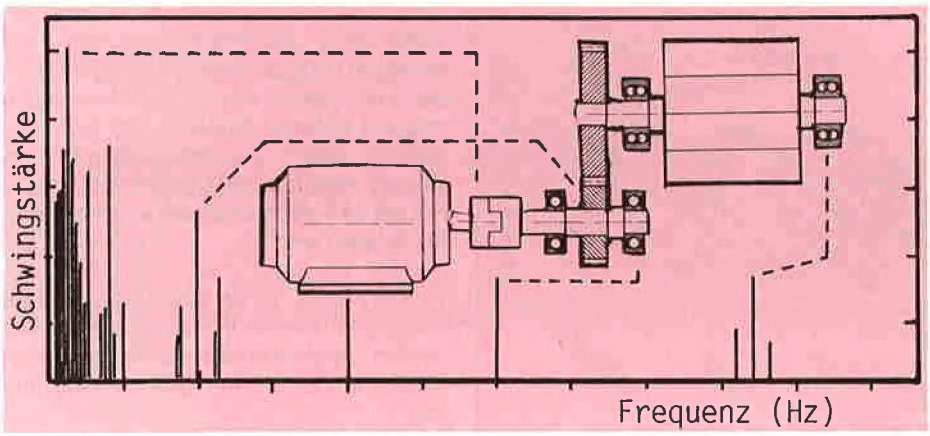



*Bild 4: Frequenzanalyse*

schließung ein bidirektionaler I/O-Port verwendet. Die meisten Anwendungsfälle aus dem Meß- und Steuerbereich werden vom Angebot an Interfacekarten abgedeckt. So ist z.B. mit diesem System auf der Zugspitze die Meß- und Regeltechnik für die Wetterstation mit Solaranlage realisiert worden.

Beide Konzepte weisen von der technischen Seite gegenüber konventioneller Meßwerterfassung und -verarbeitung entscheidende Vorteile auf:
Der Arbeits- und Zeitaufwand kann in der Regel erheblich reduziert, Messungen können standardisiert und die Qualität der Meßwertaufnahme und -weiterverarbeitung kann gesteigert werden. Weiterhin führte diese Technik auch zu neuen Aufgabenbereichen.

So konnte ein seit Jahren im Großmaschinenbau zum technischen Standard gehörendes Zustandsüberwachungssystem mit Hilfe eines hardwaremäßig aufgerüsteten Personalcomputers auf kleinere Maschinenanlagen übertragen werden. Bei Dampfturbinen in Kraftwerken wird mittels Schwingungsmessungen kontinuierlich eine Zustandsüberwachung durchgeführt. Eine genaue Analyse des Schwingungssignals ermöglicht eine Früherkennung von Schäden an Lagern und Getriebeteilen.
Durch die Verknüpfung eines aufgerüsteten Apple IIe, konventioneller Schwingungsmeßtechnik und dem „Know-how" aus dem Großmaschinenbau ist ein Sytem entwickelt worden, das auch für kleinere Maschinen zuverlässig und wirtschaftlich arbeitet. Dadurch ist man in der Lage, Instandhaltung und Ausfallkosten für produktionswichtige Maschinen (z.B. Verdichterstationen und Ventilatoren) zu senken.

Ein weiteres Beispiel für ein Meßinstrument auf der Basis des Apple II ist ein digitales Speicheroszilloskop, mit dem analog-digital gewandelte Meßwerte auf dem Monitor dargestellt werden. Es verfügt über alle Funktionen der modernen Oszillographentechnik wie z.B. Pre- und Post-Triggermöglichkeit. Es ersetzt einen Transientenrecorder ebenso wie einen „Signal-Averager".

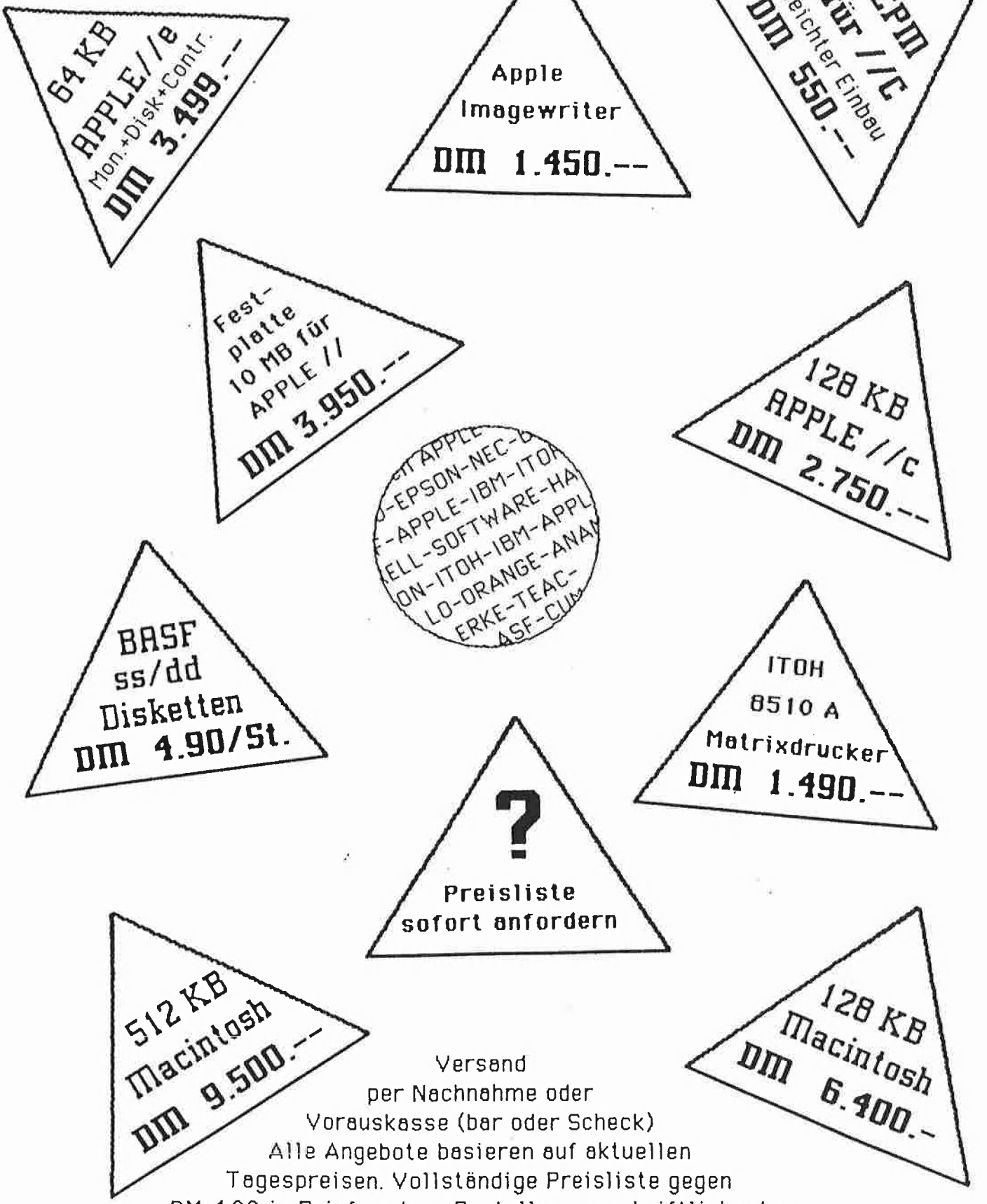
Der entscheidende Vorteil der von Personalcomputern unterstützten Meßtechnik ist das ausgezeichnete Preis-Leistungsverhältnis. So konnte die Stadt Köln in ihrer Funktion als Schulträger den Nachweis erbringen, daß die Kosten für Physiksaalausstattungen für den gymnasialen Oberstufenunterricht sich durch Personalcomputer mit entsprechender Hard- und Software von DM 350.000,- auf ca. DM 200.000,- senken lassen. Man kann also im Bereich Meßtechnik und Signalanalyse nicht nur Zeit für schöpferische Tätigkeiten gewinnen, sondern man spart noch erhebliche Anschaffungskosten für verschiedenartige Meßgeräte.

**Über den Autor**

Der Autor dieses Aufsatzes, Dipl.-Ing. B. Worms VDI aus 5090 Leverkusen, Altstadtstr. 143, arbeitet auf den Gebieten Schwingungs-, Meß- und Regeltechnik. Für den Apple II liegen u.a. folgende Programme (mit entsprechender Hardware) für signaltechnische Aufgaben vor:
- Computer-Oszilloskop
- 1-Kanal-Fouriertransformation
- Langzeit-EEG-Überwachung
- Datenakquisitionspakat für Chromatographie und Spektroskopie

Da wir diese Spezialprogramme nicht im einzelnen im „Peeker" besprechen können, werden interessierte Leser gebeten, sich direkt an den Autor zu wenden.
*Anm. der Red.*


*Bild 5: Transportables, digitales Speicheroszilloskop*




Versand
per Nachnahme oder
Vorauskasse (bar oder Scheck)
Alle Angebote basieren auf aktuellen
Tagespreisen. Vollständige Preisliste gegen
DM 4.00 in Briefmarken. Bestellungen schriftlich oder
telefonisch. Lieferung unfrei, ab Lager innerhalb 24 Stunden.

>>>>>> pyramid versand koslik <<<<<<<<
>>>>>>> 7800 freiburg steinhalde 109 <<<<<<<
>>>>>>>>>>>>>> telefon 0761 / 63675 <<<<<<<<<<<<<<<<<<<<

**Disketten sind heute das wichtigste Speichermedium für Mikrocomputer. Ihre Benutzung ist denkbar einfach und zuverlässig. Um dies zu erreichen, ist ein kompliziertes Zusammenspiel von Hardware und Software erforderlich. Die in diesem Heft beginnende Artikelserie soll eine Einführung in Aufbau und Funktion der Diskettensysteme geben. Dabei wird der Schwerpunkt auf der Erläuterung der Hardware liegen. Die Software wird nur soweit behandelt, wie sie zur unmittelbaren Ansteuerung der Hardware dient.**

# Aufbau und Funktion von Diskettensystemen

**von Dipl.-Ing. Gerhard Berg**

**Bild 1** zeigt den typischen Aufbau eines Diskettensystems. Auf der linken Seite des Diagramms steht das Disketten-Betriebssystem (DOS = Disk Operating System), das vom Benutzer mit Befehlen wie Load, Save, Write, Read usw. direkt von der Tastatur oder per Programm angesprochen wird. Das Betriebssystem steuert Computer, Controller und Laufwerk so, daß die gewünschten Daten oder Programme auf die Diskette geschrieben oder von ihr gelesen werden.
Im vorliegenden Artikel werden Aufbau und Funktion der einzelnen Bausteine allgemein beschrieben. Weitere Artikel sind geplant über Apple-Laufwerk und Apple-Controller mit der zugehörigen Software sowie über den Anschluß von Nicht-Apple-Laufwerken (z.B. BASF) an den Apple. Wenn in den Artikeln von „Apple" gesprochen wird, so ist damit der Apple II, II+ oder IIe und damit kompatible Systeme gemeint, die alle das gleiche Diskettensystem Disk II benutzen.

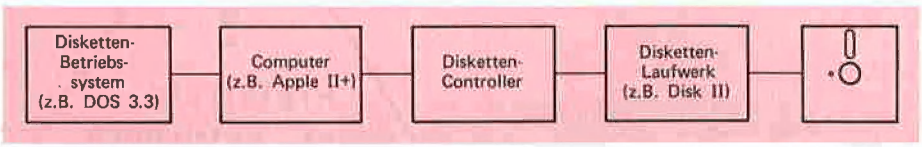



*Bild 1: Übersicht über ein Diskettensystem*

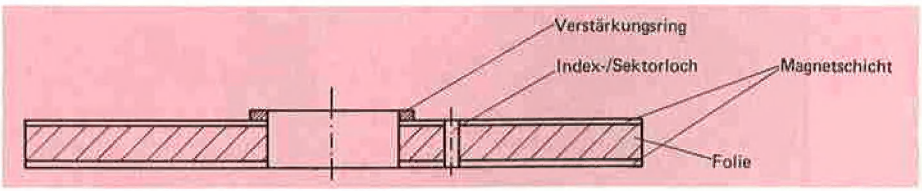



*Bild 2: Schnitt durch eine 5,25-Zoll-Diskette*

## 1. DIE DISKETTE

**Bild 2** zeigt den Schnitt durch eine 5,25-Zoll-Diskette. Sie besteht aus einer 0,085 mm dicken Kunststoffolie, auf die auf beiden Seiten eine nur 0,002 mm dicke Magnetschicht aufgebracht ist, in der die Information gespeichert wird. Die Diskette wird am Innendurchmesser im Laufwerk eingespannt. Zum Schutz vor Beschädigungen ist dieser oft mit einem Verstärkungsring versehen. In die Diskette sind in einem bestimmten Radius ein oder mehrere Löcher gestanzt. Diese werden als Index- bzw. Sektorlöcher bezeichnet und dienen als Referenz bei der Drehung der Diskette.
Zum Schutz steckt die Diskette in einer Hülle, die nur ein paar Öffnungen an den Stellen hat, wo das Laufwerk auf die Diskette zugreifen muß. **Bild 3** zeigt eine 5,25-Zoll-Diskette in der Hülle. In eine Seite der Hülle ist eine Kerbe gestanzt, die

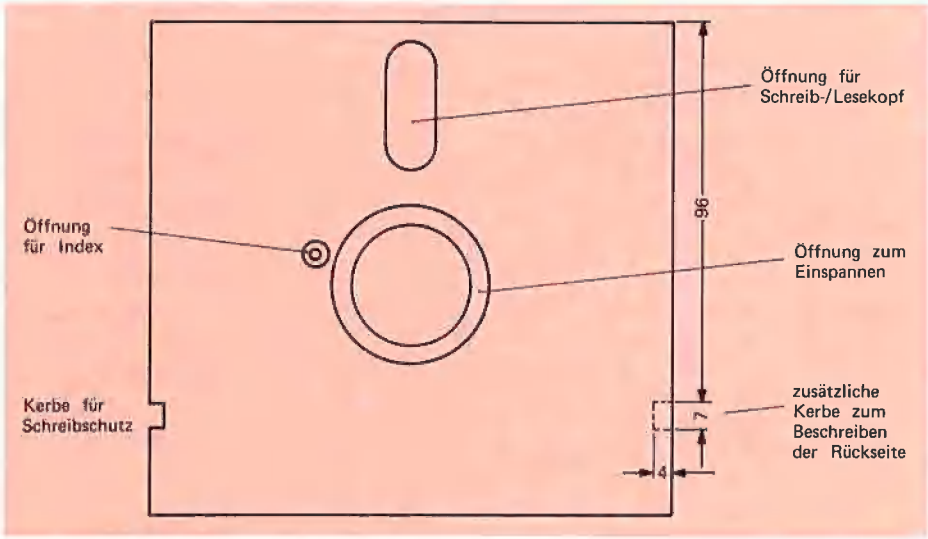



*Bild 3: Ansicht einer 5,25-Zoll-Diskette*

als **Schreibschutz** dient. Wenn die Kerbe offen ist, kann auf die Diskette geschrieben werden. Ist die Kerbe zugeklebt, kann nicht auf die Diskette geschrieben werden. (Bei manchen Laufwerken ist diese Zuordnung auch genau umgekehrt oder kann über eine Brücke umgeschaltet werden.)
Bei der Aufzeichnung wird ein Magnetkopf auf einen bestimmten Radius der Diskette

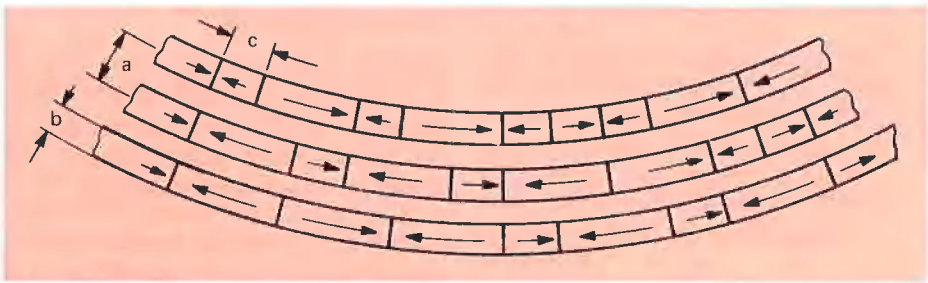



*Bild 4: Aufzeichnung auf der Diskette*

positioniert. Wird durch den Magnetkopf ein Strom geschickt und gleichzeitig die Diskette gedreht, entsteht eine kreisförmige, in sich geschlossene, magnetisierte **Spur**. Durch Positionierung auf unterschiedliche Radien können so mehrere konzentrische Spuren aufgezeichnet werden. Auf physikalischer Ebene ist die Aufzeichnung ganz einfach. Die Magnetschicht wird durch den Magnetkopf entweder voll in die eine oder voll in die andere Richtung magnetisiert. Das ist alles. Der Richtungswechsel der Magnetisierung (des Magnetflusses) wird als **Flußwechsel** bezeichnet.

**Bild 4** zeigt einen Ausschnitt aus der Aufzeichnung. Darin ist:

a – der Spurabstand (von Spurmitte zu Spurmitte)
b – die Spurbreite
c – der Flußwechselabstand

Der Kauf von Disketten fällt oftmals wegen der großen Typenvielfalt nicht ganz leicht. Deshalb sollen die Unterschiede der einzelnen Typen erläutert werden und darauf hingewiesen werden, welches die richtige Sorte für den Apple ist. Ganz pauschal kann man sagen, daß, wenn von einer Marke mehrere Sorten angeboten werden, die billigste für den Apple ausreicht. Glauben Sie nicht, daß Sie Ihrem Apple etwas Gutes tun und teuere Disketten kaufen sollten. Das wäre genau falsch, da teuere Disketten für hohe Anforderungen vorgesehen sind, für den Apple aber nicht erforderlich oder nicht geeignet sind.

### a. Durchmesser

Disketten gibt es mit einem Durchmesser von 8 Zoll (ca. 20 cm), 5,25 Zoll (ca. 13 cm) und 3,5 Zoll (ca. 9 cm). Die 8-Zoll- und 5,25-Zoll-Disketten befinden sich beide in einer flexiblen Hülle und werden deshalb auch FlexyDisk oder Floppy genannt. Die neueren 3,5-Zoll-Disketten sind zum besseren Schutz in eine feste Plastikkassette eingebaut. Der Apple arbeitet mit 5,25-Zoll-Disketten.

### b. Einseitige/doppelseitige Disketten

Disketten werden in einseitig (ss = single sided) und doppelseitig (ds = double sided) unterschieden. Grundsätzlich haben alle Disketten (auch einseitige) auf beiden Seiten eine Magnetschicht. Einseitige Disketten sind jedoch nur zur Aufzeichnung auf einer Seite vorgesehen und deshalb nur auf einer Seite geprüft. Die andere Seite kann Fehler in der Magnetschicht haben, was zu Fehlern bei der Aufzeichnung führen kann.
Im allgemeinen ist die Qualität von Markendisketten so gut, daß man für nichtprofessionelle Zwecke getrost einseitige Disketten beidseitig verwenden kann. Voraussetzung ist, daß das Laufwerk dies – wie beim Apple – zuläßt. Da der Apple weder Index- noch Sektorlöcher braucht, können Disketten einfach durch Herumdrehen auf der Rückseite benutzt werden. Dazu muß lediglich eine zusätzliche Kerbe in die Hülle der Diskette gestanzt werden (s. Bild 3), um das Schreiben auf der Rückseite zu ermöglichen. Mit einem Locher und etwas Übung ist das kein Problem. Die Kerbe sollte nicht kleiner als in Bild 3 gezeigt sein. Es macht aber nichts aus, wenn die Kerbe etwas größer ist.
Natürlich können auch doppelseitige Disketten auf dem Apple verwendet werden. In diese muß aber auch die zusätzliche Kerbe gestanzt werden, wenn sie auf der Rückseite verwendet werden sollen.

### c. Spurdichte

Die Spurdichte wird in **tpi** (= tracks per inch = Spuren pro Zoll) angegeben und ist gleich dem Kehrwert des Spurabstandes.
Bei 8-Zoll-Disketten wird normalerweise mit 48 tpi gearbeitet.
Bei 5,25-Zoll-Disketten sind zwei verschiedene Spurdichten gebräuchlich. Die normale (einfache) Spurdichte (single track density) ist 48 tpi, was einem Spurabstand von ca. 0,5 mm entspricht. Die doppelte Spurdichte (double track density) ist 96 tpi (manchmal auch 100 tpi).
Bei Laufwerken mit doppelter Spurdichte sollen 96-tpi-Disketten verwendet werden. Bei Laufwerken mit einfacher Spurdichte, wie beim Apple, können 48- oder 96-tpi-Disketten verwendet werden, wobei 48-tpi-Disketten meist billiger sind.

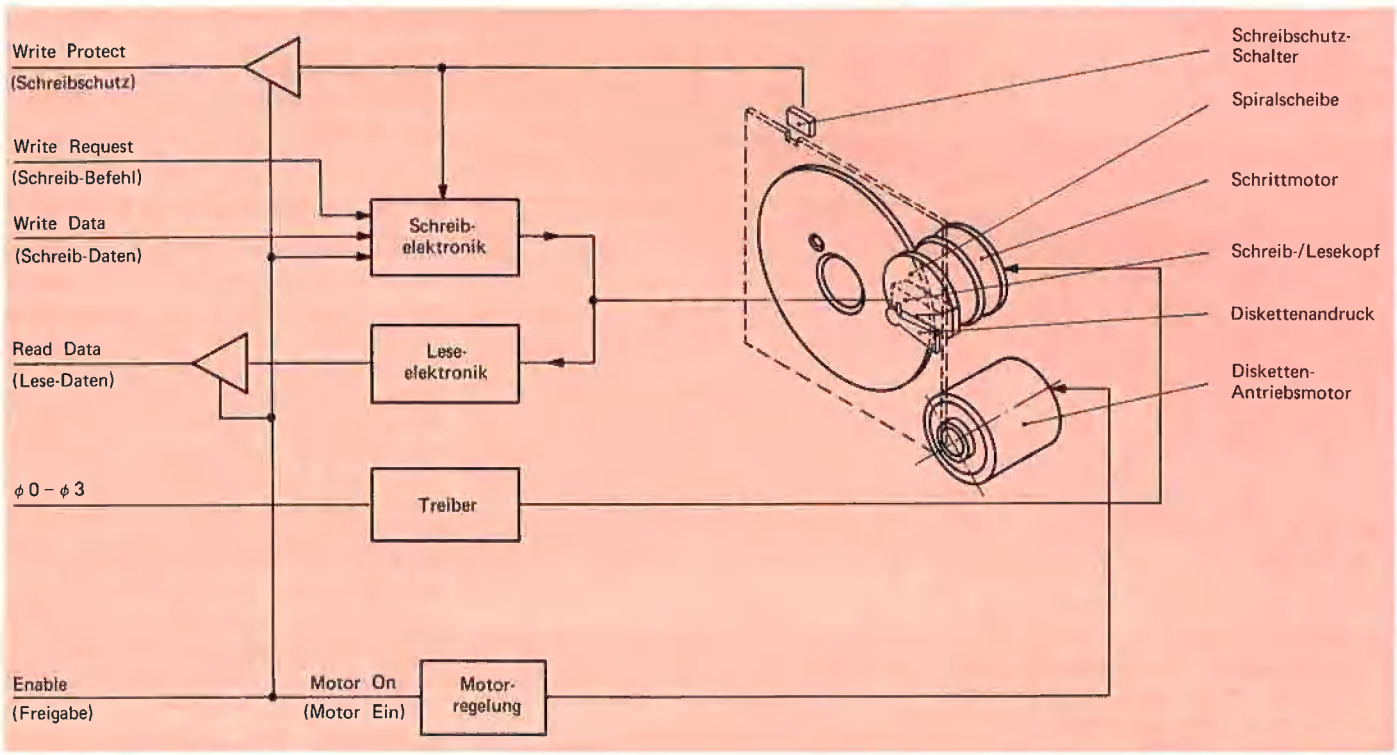



*Bild 5: Aufbau eines Diskettenlaufwerks*

### d. Aufzeichnungsdichte

Die Aufzeichnungsdichte wird in fci oder bpi angegeben.
Die Bezeichnung **fci** (flux changes per inch = Flußwechsel pro Zoll) ist die Einheit der **Flußwechseldichte** (flux density). Die Flußwechseldichte ist gleich dem Kehrwert des kleinsten Abstandes zwischen zwei Flußwechseln auf der innersten Spur. Die übliche Flußwechseldichte ist 6536 fci bei 8-Zoll-Disketten und 5536 fci bei 5,25-Zoll-Disketten. Das entspricht einem minimalen Abstand zwischen zwei Flußwechseln von nur 0,004 mm bzw. 0,005 mm.
Die Bezeichnung **bpi** (bits per inch = Bits pro Zoll) ist die Einheit der Bitdichte (bit density). Die **Bitdichte** ist gleich dem Kehrwert des Abstandes zwischen zwei Bits auf der innersten Spur. Da je nach Aufzeichnungsverfahren mehr oder weniger Flußwechsel zur Aufzeichnung eines Bits benötigt werden, gibt es keine generelle Umrechnung zwischen bpi und fci. Bei dem Aufzeichnungsverfahren mit einfacher Dichte (FM; siehe weiter unten) ist die Bitdichte halb so groß wie die Flußwechseldichte. Bei dem Aufzeichnungsverfahren mit doppelter Dichte (MFM) ist die Bitdichte gleich der Flußwechseldichte.
Entsprechend den zwei verschiedenen Aufzeichnungsverfahren gibt es auch unterschiedliche Disketten für einfache (Aufzeichnungs-)Dichte (sd = single density) und doppelte Dichte (dd = double density). Für den Apple sollten Disketten für einfache Dichte verwendet werden. Disketten für doppelte Dichte sind teurer und für das vom Apple verwendete Aufzeichnungsverfahren nicht erforderlich bzw. nicht geeignet. Die neueren Disketten für sehr hohe Aufzeichnungsdichten (hd = high density) dürfen im Apple auf keinen Fall verwendet werden, da sie zu zahlreichen Fehlern führen.

### e. Soft-/Hardsektorierte Disketten

Softsektorierte Disketten haben ein Indexloch, das es dem Laufwerk auf einfache Weise ermöglicht, den Anfang einer Spur zu erkennen.
Hardsektorierte Disketten haben mehrere Sektorlöcher und ein Indexloch auf dem gleichen Radius. Bei diesen Disketten ist jede Spur in eine feste Anzahl von Abschnitten (Sektoren) eingeteilt. Jedes der im gleichen Abstand befindlichen Sektorlöcher kennzeichnet den Beginn eines Sektors. Das Indexloch liegt zwischen zwei Sektorlöchern und markiert den Anfang der Spur.
Da der Apple weder Index- noch Sektorlöcher benutzt, können sowohl soft- als auch hardsektorierte Disketten mit dem Apple verwendet werden.

### f. Verstärkungsring

Disketten werden mit oder ohne Verstärkungsring (hard hole) angeboten. Die Einspannung der Disketten ist einer der kritischsten Punkte in jedem Laufwerk. Die Diskette muß fest gehalten und exakt zentriert werden, darf aber nicht im geringsten beschädigt werden. Der Verstärkungsring verringert die Gefahr von Beschädigungen und ist deshalb durchaus zu empfehlen.

### g. Formatierung

Für verschiedene Computersysteme muß auf der Diskette ein ganz bestimmtes Format aufgezeichnet sein, bevor die Diskette überhaupt benutzt werden kann. Es gibt deshalb die unterschiedlichsten Diskettentypen, auf die jeweils das Format für einen bestimmten Computer aufgezeichnet ist. Da beim Apple die Formatierung im System selbst erzeugt wird, können für den Apple Disketten ohne Formatierung oder mit Formatierung für jeden beliebigen Computer verwendet werden.

## 2. DAS LAUFWERK

Der prinzipielle Aufbau eines Diskettenlaufwerkes ist in **Bild 5** gezeigt. Die Lauf-

werke der einzelnen Hersteller können sich mehr oder weniger von diesem Aufbau unterscheiden. Alle Disketten-Laufwerke müssen jedoch vier Grundfunktionen erfüllen: Antrieb der Diskette, Positionieren des Schreib-/Lesekopfes, Schreiben auf die Diskette und Lesen von der Diskette.
Die Funktionen des Laufwerks werden über eine Reihe von Signalleitungen gesteuert, die zusammengefaßt als **Schnittstelle** (Interface) bezeichnet werden. Die meisten Disketten-Laufwerke verwenden eine genormte Schnittstelle, die auch als Shugart-Schnittstelle bezeichnet wird, weil die Firma Shugart die ersten 8- und 5,25-Zoll-Disketten-Laufwerke auf den Markt gebracht hat. Die Schnittstellen dieser Geräte haben sich allgemein in der Industrie durchgesetzt und wurden später auch offiziell genormt. Für 8-Zoll-Laufwerke hat die genormte Schnittstelle 50 Leitungen und für 5,25-Zoll-Laufwerke 34 Leitungen. Zu dem Stecker mit den Signalleitungen kommt in beiden Fällen noch ein bzw. zwei Stecker für die Stromversorgung hinzu. Die Laufwerke von Apple bilden eine Ausnahme, da sie nicht die Normschnittstelle verwenden. Bei ihnen werden sowohl Signal- als auch Stromversorgungsleitungen über einen gemeinsamen 20poligen Stecker geführt.

### a. Der Diskettenantrieb

Der Diskettenantrieb muß dafür sorgen, daß sich die Diskette mit konstanter Geschwindigkeit dreht. Bei 8-Zoll-Disketten ist die Solldrehzahl 360 und bei 5,25-Zoll-Disketten 300 Umdrehungen pro Minute (Upm). 300 Upm entspricht 5 Umdrehungen pro Sekunde oder 200 msec pro Umdrehung.
Bei den meisten 8-Zoll-Laufwerken erfolgt der Antrieb mit einem Wechselstrom-Synchronmotor, der direkt aus dem Netz gespeist wird. Bei neueren 8-Zoll-Laufwerken und bei allen 5,25-Zoll-Laufwerken erfolgt der Antrieb mit einem Gleichstrommotor. In diesem Fall sorgt eine elektronische Drehzahlregelung dafür, daß die Diskette mit konstanter Drehzahl läuft. Die Solldrehzahl läßt sich dabei meist mit einem Potentiometer einstellen. Über eine Schnittstellen-Leitung läßt sich der Motor ein- und ausschalten.
Der Antrieb der Diskette erfolgt meist über einen Antriebsriemen. Einige neuere Laufwerke haben einen Direktantrieb, bei dem die Einspannvorrichtung für die Diskette direkt auf der Motorwelle sitzt.

### b. Die Positionierung

Die Aufgabe der Positionierung ist es, den Schreib-/Lesekopf auf den gewünschten Spurradius zu positionieren. Die Positionierung erfolgt bei fast allen Laufwerken mit einem **Schrittmotor**. Der Aufbau eines Schrittmotors ist in stark vereinfachter Form in **Bild 6** gezeigt. In diesem Beispiel besteht der Motor aus zwei feststehenden Statoren, die je eine Spule mit Mittelanzapfung tragen. Die Mittelanzapfungen sind an die positive Versorgungsspannung angeschlossen. Die Enden der Spulen können mit Hilfe der Transistoren T0 bis T3 nach Masse geschaltet werden. Zur Steuerung dienen die Signale ø0 bis ø3 (Phase 0 bis 3). Der Rotor ist ein einfacher permanenter Stabmagnet. In Wirklichkeit haben Stator und Rotor mehrere Pole. Das vereinfachte Modell reicht aber aus, um die Funktion eines Schrittmotors zu erklären. Die **Bilder 6a–e** zeigen den Ablauf einer Positionierung.

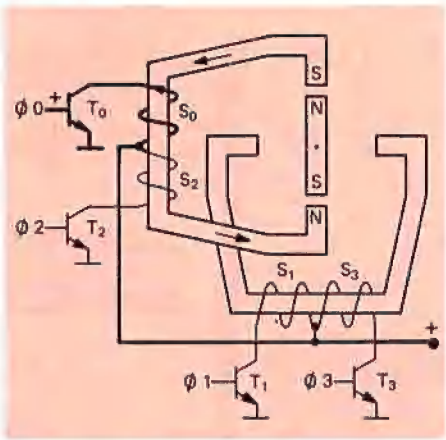



*Bild 6a: Schrittmotor Ø 0 eingeschaltet*

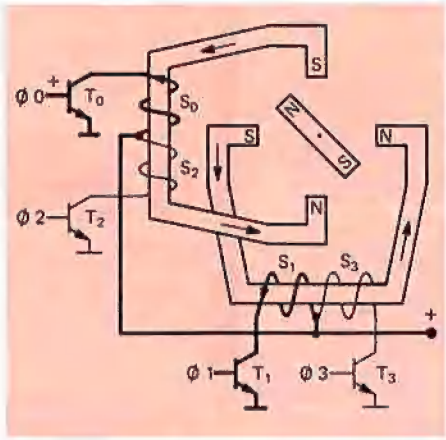



*Bild 6b: Schrittmotor Ø 0 und Ø 1 eingeschaltet*

In **Bild 6a** befindet sich der Kopf auf der Ausgangsspur. Das Signal ø0 ist positiv. Dadurch ist der Transistor T0 eingeschaltet und durch die Spule S0 fließt ein Strom. Durch den Magnetfluß entsteht am oberen Statorpol ein Südpol und am unteren ein Nordpol. Die Süd- und Nordpole ziehen sich gegenseitig an, wodurch der Rotor in der gezeigten Stellung gehalten wird.

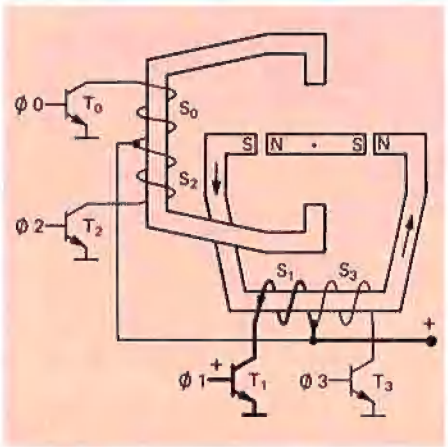



*Bild 6c: Schrittmotor Ø 1 eingeschaltet*

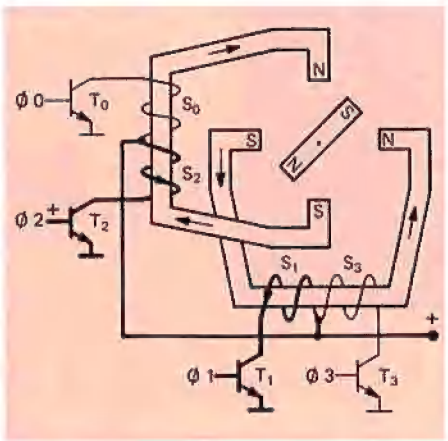



*Bild 6d: Schrittmotor Ø 1 und Ø 2 eingeschaltet*

Hinweis: Unsere Druckerei, HVA, hat kein richtiges Phi. Deshalb wurde das Durchmesserzeichen gesetzt.

In **Bild 6b** wird mit ø1 zusätzlich T1 eingeschaltet, wodurch der linke Pol auch noch ein Südpol wird. Der Nordpol des Rotors wird jetzt von beiden Südpolen gleich stark angezogen, wodurch sich der Rotor um 45 Grad nach links dreht und genau zwischen den beiden Polen stehenbleibt. Im nächsten Schritt in **Bild 6c** wird der Strom durch T0 und S0 ausgeschaltet.

# Mikrocomputer ● Sys

IBS Euro Profi
Bestell-Nr.: B 2945

NEC Monitor
Bestell-Nr.: B 2946

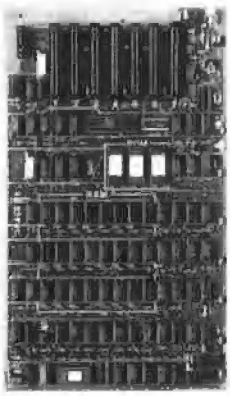
CSC Tastatur
Bestell-Nr.: B 2947

Mit dieser Übersicht möchten wir Ihr Interes
Ihr Computer mit Applebus durch Interfac
Bei mehr als 500 Händlern können Sie Cor
Technologie aus Deutschland für die App
Wir informieren Sie gerne und kostenlos.

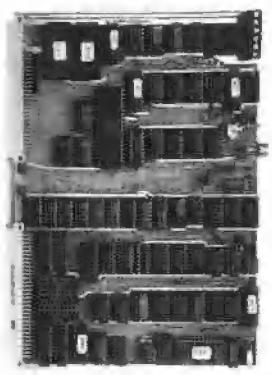
PREH COMMANDER Tastatur
Bestell-Nr.: B 2937

## Prozessorkarten

## SPACE-SINGLEBOARDCOMPUTER



SPACE 84
Das Motherboard in der „Appleklasse" mit integrierter RAMFLOPPY, zweifach EPROM-System mit FORTH und der Sprache Ihrer Wahl. Voll applekompatibel bilden die SPACE-Motherboards die Grundlage für viele tausende von Computersystemen. SPACE 84 ist der vernünftige Kompromiß zwischen Preis und Leistung. Es ist bereits wenige Wochen nach seiner Einführung Mittelpunkt in Mikrocomputersystemen erhöhter Leistungsanforderung. Sie sollten nicht weniger als SPACE 84 für Ihren Computer verlangen.
Bestell-Nr.: A 8400



SPACE 85-1
Eine interessante Variante der Appleklasse stellt das Board SPACE 85-1 dar. Als Einplatinencomputer bietet er alles das, was man von einem Minimalsystem fordert. Seine Leistung zeigt er aber erst so richtig als Systemmaster in einem Doppeleuropakartensystem. Sein Standardbus ist 64-polig, und er kann sieben weitere Interfacekarten kontrollieren. Das Besondere ist aber, daß jede Karte wieder sieben „normale APPLESLOTS" nachbilden kann. Dadurch stehen in diesem System 49 Slots zur Verfügung. Genug also, um auch die kompliziertesten Aufgaben zu lösen. Die Verwendung von indirekten Steckverbindern garantiert höchstes Maß an Sicherheit für Ihre Daten und Steuerungsaufgaben.
Bestell-Nr.: A 8510

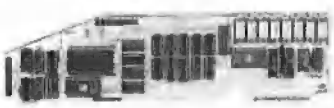
## INTEMEX

CPU und RAM für den APPLEBUS.
Optimal kann man eine CPU nur dann nutzen, wenn sie ohne Einschränkung arbeiten kann. Bereits 1981 entstand die erste INTEMEX-Karte für zusätzliche Prozessorleistung mit eigenem RAM bei IBS. Ein Prinzip, das danach von vielen anderen Anbietern übernommen wurde. Die berühmte „Nasenlänge" bietet Ihnen in IBS einen Partner, dem Sie vertrauen können.

AP 10
INTEMEX mit der 6809 CPU und 64 k-RAM. Mit ihr begann die Geschichte der INTEMEX-Serie. Diese Karte eröffnet Ihnen den Zugang zu dem großen Angebot der FLEX-Software.
Bestell-Nr.: A 1010

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



AP 21
INTEMEX mit 6511 CPU und 64 k-RAM. Diese Karte bietet Ihnen viele Möglichkeiten, I/O-Probleme zu lösen. Ihr Hauptrechner wird entlastet und steht für Ihre Aufgaben schneller zur Verfügung.
Bestell-Nr.: A 1021

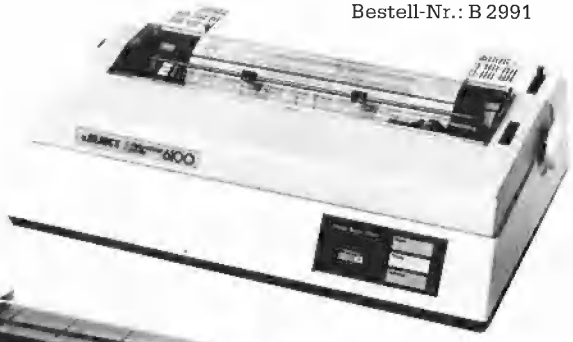
AP 22
INTEMEX mit Z 80 B-CPU und 64 k-RAM. Wenn Sie einmal diese Karte in Aktion gesehen haben, werden Sie auch feststellen: „Geschwindigkeit ist keine Hexerei, man braucht nur die AP 22". Mit dieser Karte wird Ihr APPLE II zum z.Z. schnellsten CP/M-Computer, und in Verbindung mit dem SPACE 84 erhalten Sie Computerleistung, die wirklich einmalig ist. Wir vermitteln gerne eine Vorführung.
Bestell-Nr. A 1022

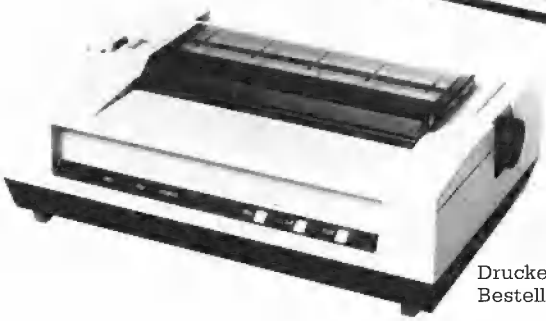
# eme ● Baugruppen ● Zubehör

wecken und Ihnen zeigen, wie leistungsfähig
von IBS heute sein kann.
uterleistung „Made by IBS" kaufen.
Welt.

Intekey Tastatur
Bestell-Nr.: B 2957

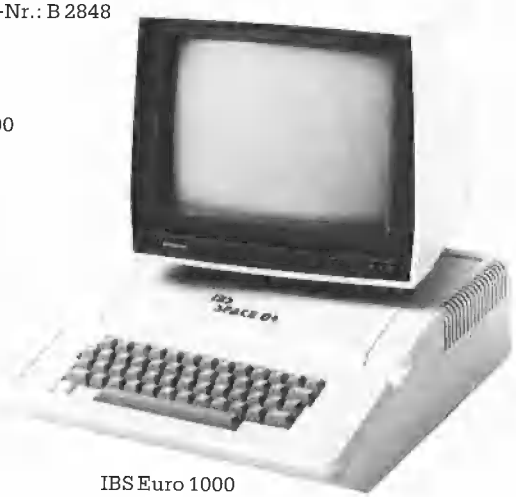
Tpenraddrucker JUKI 6100
Bestell-Nr.: B 2991





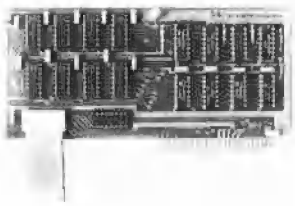
Drucker ITOH 8510 A
Bestell-Nr.: B 2990

SANYO 8112 Monitor
Bestell-Nr.: B 2848

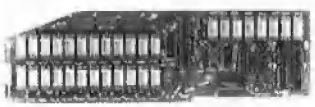


IBS Euro 1000
Bestell-Nr.: B 2845

## RAM-Karten

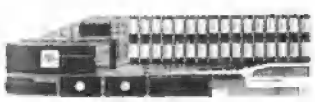


AP 1
16 kB-RAM-Karte zur Erweiterung aller 48 k-Applesysteme auf 64 k-Arbeitsspeicher. Alle Sprachen und Systeme sind einsetzbar.
Bestell-Nr.: A 1001



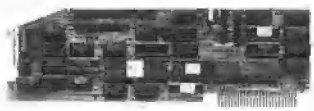
AP 13 und AP 17
RAM-Karten zum Einsatz als Pseudodisk unter CP/M, USCD und APPLE-DOS. Speichergröße von 64 kByte bis 256 kByte.
Bestell-Nr.: A 1013 a-b
A 1017 a-d

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

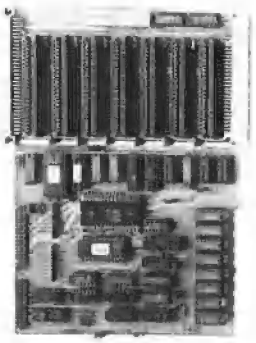


AP 26
Speicherkarte für den Einsatz als Arbeitsspeichererweiterung unserer INTEMEX 68 000 bis 1 MByte.
Bestell-Nr.: A 1026

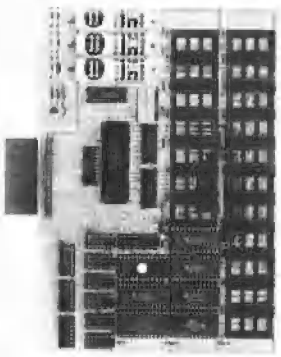
## Spezialentwicklungen



AP 14
Floppy-Controller für alle Anwendungsfälle. 10 Laufwerke können gleichzeitig angeschlossen werden. 4 × 8" DSDD, 4 × 5¼" DSDD und zwei Apple-Standardlaufwerke. Maximal ca. 10MByte im Direktzugriff.
Bestell-Nr.: A 1014



SPACE 85-2
Ermöglicht den Einsatz aller Apple-Interfaces an dem SPACE 85-1. Zusätzlich bietet SPACE 85-2 alle Eigenschaften der AP 21.
Bestell-Nr.: A 8520



48k-CMOS-RAM
Diese Karte ersetzt den dynamischen RAM-Bereich Ihres APPLE durch 48k-CMOS-RAM.
Bestell-Nr. A 8648

Dadurch wird der obere und untere Pol unmagnetisch. Der Nordpol des Rotors wird jetzt nur noch vom Südpol des linken Pols angezogen, wodurch sich der Rotor um die nächsten 45 Grad nach links dreht.
In **Bild 6d** wird mit ø2 der Strom durch T2 und S2 eingeschaltet. Der Strom und damit auch der Magnetfluß fließt durch S2 in umgekehrter Richtung von S0. Dadurch entsteht der Südpol jetzt am unteren Pol. Der Nordpol des Rotors wird jetzt wieder von zwei Südpolen angezogen und dreht sich damit um die nächsten 45 Grad nach links.

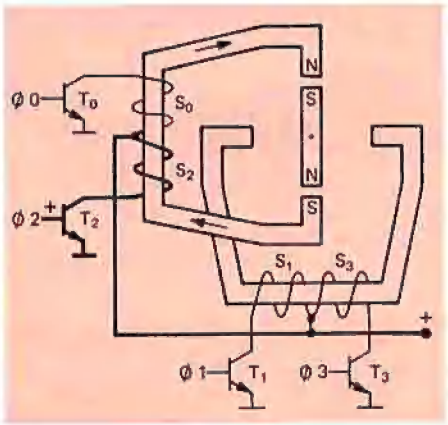



*Bild 6e: Schrittmotor Ø 2 eingeschaltet*

In **Bild 6e** wird der Strom durch T1 und S1 wieder ausgeschaltet, wodurch der Rotor die gezeigte Endposition erreicht.

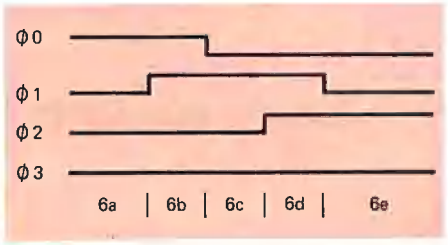



*Bild 7: Zeitdiagramm der Positionierung*

**Bild 7** zeigt ein Zeitdiagramm für den in Bild 6a bis 6e gezeigten Ablauf. Dieser entspricht übrigens beim Apple-Laufwerk genau der Positionierung von einer Spur zur nächsten. Einziger Unterschied ist, daß sich der Schrittmotor wegen der größeren Polzahl nicht um 180 Grad, sondern nur um 15 Grad dreht.
Die Drehbewegung des Schrittmotors muß schließlich noch in eine lineare Bewegung des Kopfträgers umgewandelt werden. Dies kann auf verschiedene Arten erfolgen: mit Hilfe einer Spiralscheibe (wie beim Apple), mit einer Gewindespindel oder mit einem Stahlband, das um die Welle des Schrittmotors gewickelt wird. Die letzte Ausführung ermöglicht das schnellste Positionieren.
Bei Laufwerken mit Normschnittstelle wird das Positionieren mit den Signalen „Step" und „Direction" gesteuert. Jeder Impuls auf dem „Step"-Signal positioniert den Schreib-/Lesekopf um eine Spur weiter. Dabei wird die Richtung des Positionierens mit dem „Direction"-Signal gesteuert. Die Laufwerkselektronik enthält einen Aufwärts-/Abwärtszähler mit nachgeschalteter Dekodierung zur Erzeugung der Phasensignale. Beim Apple-Laufwerk liegen die Phasensignale direkt auf der Schnittstelle.
Bei 8-Zoll-Laufwerken werden normalerweise 77 Spuren aufgezeichnet. Die Spuren werden von außen mit 0 beginnend nach innen bis 76 numeriert.
Bei 5,25-Zoll-Laufwerken werden bei 48 tpi je nach Laufwerk 35 oder 40 Spuren und bei 96 tpi 80 Spuren aufgezeichnet. Haben die Laufwerke 2 Schreib-/Leseköpfe (um beide Seiten einer Diskette ohne Umdrehen benutzen zu können), dann sind die Spuren auf der Rückseite bei 48 tpi um 4 und bei 96 tpi um 8 Spuren nach innen versetzt (in beiden Fällen ist das der gleiche absolute Betrag). Der Radius aller Spuren (bezogen auf die Spurmitte) kann mit folgender Formel angegeben werden:

$$R = (57{,}150 - (n + 4 \cdot s \cdot d) \cdot 25{,}4 : (48 \cdot d)) \text{ mm}$$

In der Formel ist n die Spurnummer. Die Numerierung der Spuren fängt außen mit n=0 an und steigt nach innen bis n=34 bzw. n=39 für 48 tpi und bis n=79 für 96 tpi.
Die Seite ist s=0 bei einseitiger Aufzeichnung oder bei der Vorderseite bei zweiseitiger Aufzeichnung. Für die Rückseite bei Aufzeichnung mit einem Doppelkopflaufwerk ist s=1.
Der Wert d ist die Spurdichte, mit d=1 für 48 tpi und d=2 für 96 tpi.
Aus der Formel kann man sehen, daß die Position aller geraden Spuren (0,2,4...) einer 96-tpi-Diskette mit den Spuren einer 48-tpi-Diskette übereinstimmt. Das gilt sowohl für die Vorder- als auch die Rückseite.

### c. Das Schreiben

Die **Bilder 8a und 8b** zeigen das Prinzip des Schreibvorgangs. Der **Schreibkopf** besteht im wesentlichen aus einem ringförmigen Magnetkern mit einem Luftspalt. Auf den Magnetkern ist eine Spule mit Mittelanzapfung gewickelt. Die beiden Enden der Spule können wahlweise über je einen Transistor nach Masse durchgeschaltet werden. Die Mittelanzapfung liegt über einen Vorwiderstand an der positiven Versorgungsspannung. Die Größe des Vorwiderstandes bestimmt die Höhe des zum Schreiben verwendeten Stromes.

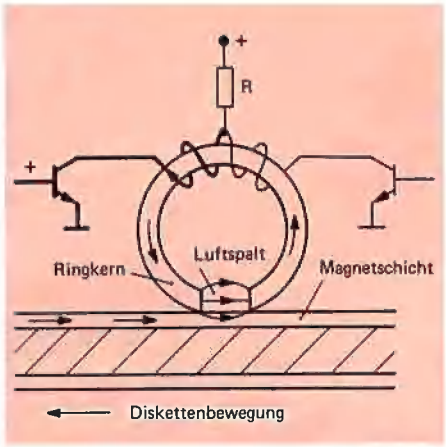



*Bild 8a: Schreibvorgang Magnetisierung nach rechts*

Im **Bild 8a** ist der linke Transistor eingeschaltet, was einen Strom durch die linke Spule zur Folge hat. Aus diesem Strom resultiert ein Magnetfluß durch den Ringkern in Uhrzeigerrichtung. Im Luftspalt des Ringkernes schließt sich ein Teil des Magnetfeldes über die Magnetschicht der Diskette. Wenn sich die Diskette dreht (im Bild nach links), bleibt die Magnetisierung in der Breite des Schreibkopfes in der Magnetschicht der Diskette (Pfeile nach rechts) erhalten (gespeichert).

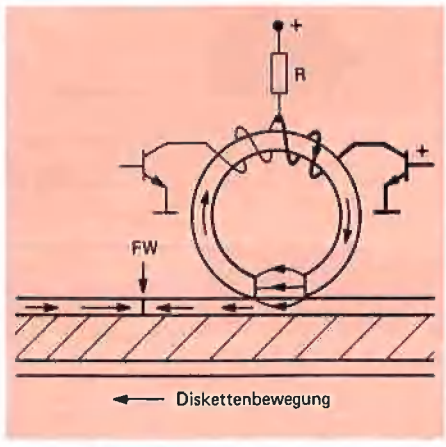



*Bild 8b: Schreibvorgang Magnetisierung nach links*

**Bild 8b** zeigt den Schreibvorgang etwas später. Die nach rechts magnetisierten Bereiche haben sich weiter vom Kopf entfernt. Zu dem Zeitpunkt, als sich der mit „FW" (Flußwechsel) markierte Punkt unter dem Schreibkopf befand, wurde der linke Transistor aus- und der rechte Transistor eingeschaltet. Durch den anderen Windungssinn der rechten Spule wurde die Magnetisierungsrichtung durch den Ringkern und damit auch durch die Magnetschicht der Diskette umgekehrt.
Wenn die Diskette eine volle Umdrehung gemacht hat, ist eine in sich geschlossene magnetisierte Spur entstanden.
Durch wechselweises Einschalten der beiden Transistoren kann die gewünschte Aufzeichnung auf der Diskette erzeugt werden. Soll nicht geschrieben werden, so werden einfach beide Transistoren ausgeschaltet.
Voraussetzung für eine einwandfreie Aufzeichnung ist, daß die Diskette richtig am Schreibkopf anliegt. Dies wird im Laufwerk durch leichtes Andrücken von der Gegenseite (über einen Andruckfilz oder einen zweiten Schreibkopf) erreicht. Das Andrücken wird je nach Laufwerk durch einen besonderen Magneten (Kopflademagnet) oder (wie beim Apple) durch Schließen der Laufwerksklappe erreicht.

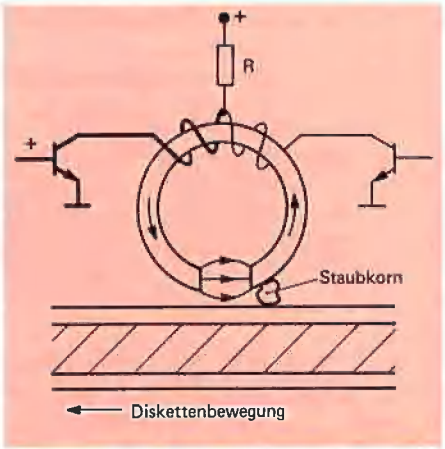



*Bild 8c: Schreibvorgang Störung durch Staubkorn*

Wird, wie in **Bild 8c** gezeigt, die Diskette nur geringfügig vom Kopf abgehoben, so nimmt die Magnetisierung der Diskette ab oder hört ganz auf, was Fehler beim Lesen zur Folge hat. Hierzu reichen schon kleinste Staubkörnchen oder sogar ein Fingerabdruck aus. Es ist deshalb von größter Wichtigkeit, die Disketten sehr sorgfältig zu behandeln, sauber aufzubewahren (immer in der Papierhülle) und niemals auf die Magnetschicht zu fassen.

Die Breite einer geschriebenen Spur ist 0,300 mm bei 48 tpi und 0,146 mm bei 96 tpi. Damit verbleibt zwischen den Spuren ein Sicherheitsabstand von nur ca. 0,23 mm bei 48 tpi und ca. 0,12 mm bei 96 tpi. Die angegebenen Zahlenwerte zeigen, wie genau ein Diskettenlaufwerk arbeiten muß und wie sorgfältig man deshalb mit den Geräten umgehen muß, wenn sie zuverlässig arbeiten sollen.
Neben dem Schreibkopf gibt es noch einen **Löschkopf**, der einen schmalen Bereich zu beiden Seiten des Schreibkopfs löscht. Dies ist erforderlich, um geringe Abweichungen bei der Einspannung der Diskette und der Positionierung auszugleichen. **Bild 9a** zeigt, was ohne Löschkopf passieren könnte. Links im Bild ist eine erste Aufzeichnung gezeigt. Rechts im Bild ist die erste Aufzeichnung durch eine zweite Aufzeichnung überschrieben. Dabei ist die Spurlage, z.B. auf einem anderen Gerät, geringfügig gegenüber der ersten Aufzeichnung verschoben. Dadurch

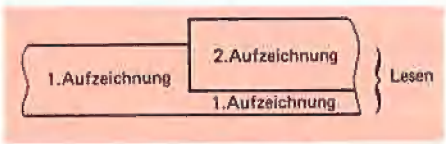



*Bild 9a: Überschreiben ohne Löschkopf*

bleibt ein Rest der ersten Aufzeichnung stehen und wird nicht überschrieben. Würde nun diese Aufzeichnung auf dem ersten Gerät mit der ursprünglichen Spurlage wieder gelesen, so würde der Lesekopf eine Mischung aus erster und zweiter Aufzeichnung lesen, was zu Fehlern führen würde. **Bild 9b** zeigt das Verfahren mit zusätzlichem Löschkopf, bei dem der Rest der ersten Aufzeichnung gelöscht wird (schraffierte Flächen), so daß es zu keinen Problemen kommt.

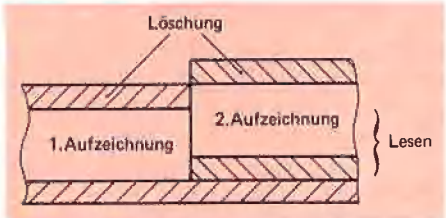



*Bild 9b: Überschreiben mit Löschkopf*

### d. Das Lesen

Beim Lesen soll die aufgezeichnete Information wieder von der Diskette gelesen werden. Es ist jedoch nicht auf einfache Weise möglich, die Richtung der gespeicherten Magnetisierung zu ermitteln. Statt dessen kann aber eine Richtungsänderung der Magnetisierung (Flußwechsel) sehr leicht festgestellt werden. Dazu wird das Induktionsgesetz angewandt, nach dem jede Änderung eines Magnetfeldes in einer Spule eine elektrische Spannung erzeugt.

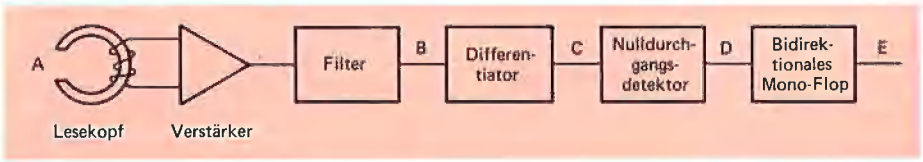



*Bild 10: Blockschaltbild des Lesekreises*

Die Leseelektronik ist bei allen Disketten-Laufwerken nach dem gleichen Prinzip aufgebaut. In vielen Fällen wird sogar der gleiche IC verwendet (MC 3470 von Motorola). Ein Blockschaltbild der Leseelektronik ist in **Bild 10** gezeigt und ein Zeitdiagramm, das den Signalverlauf an verschiedenen Stellen des Lesekreises zeigt, in **Bild 11**.

Zum Lesen wird der gleiche Kopf wie beim Schreiben verwendet, der deshalb auch Schreib-/Lesekopf genannt wird. Wird die Diskette am Kopf vorbei bewegt, dann induziert jede Änderung der in der Diskette gespeicherten Magnetisierung eine Spannung in der Spule des **Lesekopfes**. Die zunächst sehr kleine Lesespannung wird über einen **Verstärker** auf eine größere Amplitude verstärkt. Dann wird über ein **Filter** der gewünschte Frequenzbereich ausgefiltert (z.B. werden hochfrequente Störungen unterdrückt). In Bild 11 ist in Kurve A die in der Diskette gespeicherte Magnetisierung gezeigt. Die Magnetschicht ist immer voll in die eine oder voll in die andere Richtung magnetisiert. Kurve B zeigt das verstärkte und gefilterte Lesesignal für verschiedene typische Flußwechselabstände. Jedesmal, wenn die Magnetisierungsrichtung von links nach rechts wechselt, wird ein positiver Spannungsimpuls (in Kurve B) erzeugt. Ent-

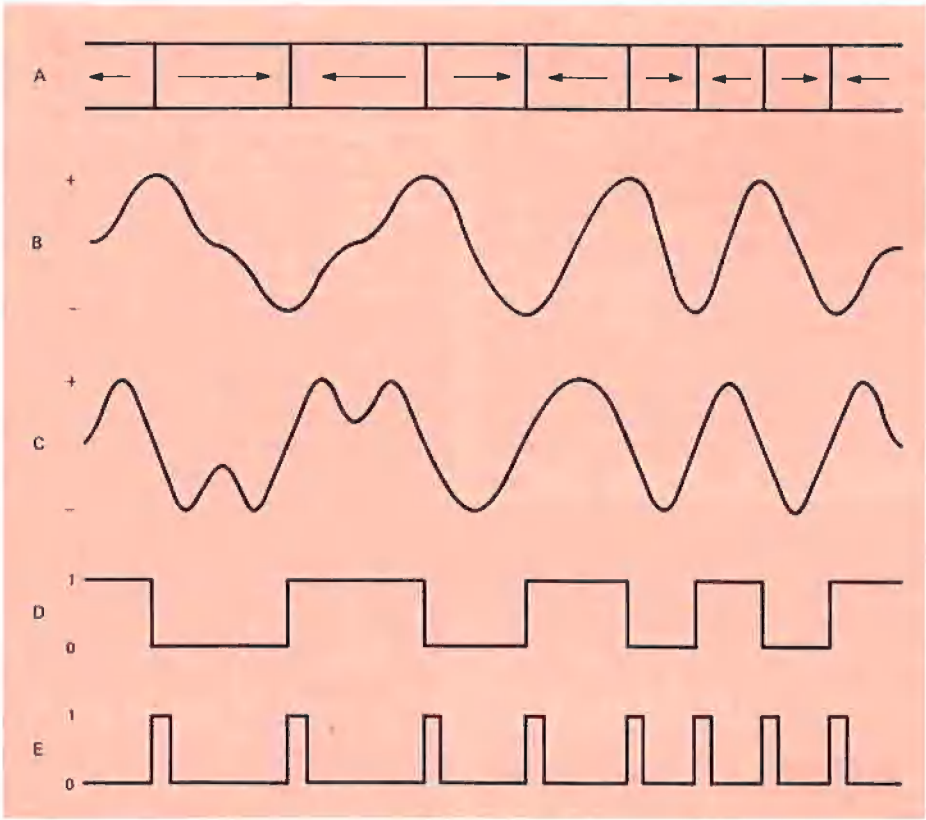



*Bild 11: Signalverlauf im Lesekreis*

sprechend wird bei einem umgekehrten Wechsel ein negativer Spannungsimpuls erzeugt. Je dichter die Flußwechsel zusammenrücken, um so mehr verschmelzen die Spannungsimpulse ineinander. Würden die Flußwechsel noch weiter zusammenrücken, dann würde die Amplitude des Lesesignals abnehmen.
Wie man sieht, hat jeder Spannungsimpuls seinen größten Wert genau zu dem Zeitpunkt, zu dem der Wechsel der Magnetisierungsrichtung stattfindet. Die Aufgabe der weiteren Schaltung ist es, zu genau diesem Zeitpunkt einen kurzen digitalen Impuls abzugeben. Da die Amplitude der Lesesignale sehr unterschiedlich sein kann (z.B. je nach Spurradius), kann man nicht einfach mit einem Komparator arbeiten, der einen Impuls abgibt, wenn ein bestimmter Spannungswert überschritten wird. Statt dessen arbeitet man mit einem sog. **Differentiator**. Das Ausgangssignal des Differentiators ist in Kurve C gezeigt. Der Differentiator hat die Eigenschaft, daß sein Ausgangsignal um so größer (positiver) ist, je steiler sein Eingangssignal ansteigt. Entsprechend wird das Ausgangssignal um so negativer, je steiler das Eingangssignal abfällt. Verläuft das Eingangssignal waagerecht, dann ist das Ausgangssignal genau null. Jedesmal, wenn das Lesesignal (Kurve B) seinen höchsten positiven oder negativen Wert erreicht, läuft es für einen ganz kurzen Moment waagerecht. Genau zu diesem Zeitpunkt geht entsprechend das differenzierte Signal (Kurve C) durch null.
Jetzt braucht man nur noch eine Schaltung, die diese Nulldurchgänge erkennt und zu jedem Nulldurchgang einen Ausgangsimpuls liefert. Als ein solcher **Nulldurchgangsdetektor** eignet sich ein Komparator, bei dem der Vergleichseingang auf null gelegt wird. Bei dem Komparator ist der Ausgang immer logisch 1, solange der Signaleingang positiver als der Vergleichseingang ist. Ist der Signaleingang negativer als der Vergleichseingang, dann ist der Ausgang logisch 0. Damit wechselt der Ausgang des Komparators (Kurve D) jedesmal von 0 nach 1, wenn der Signaleingang (Kurve C) die Nullinie von negativen Werten zu positiven Werten überschreitet. Umgekehrt wechselt der Ausgang des Komparators von 1 nach 0, wenn der Signaleingang die Nullinie von positiven nach negativen Werten überschreitet.
Als letzter Baustein folgt jetzt nur noch ein **bidirektionales Mono-Flop**. Dieses Mono-Flop erzeugt an seinem Ausgang (Kurve E) jedesmal dann einen Impuls, wenn an seinem Eingang eine Flanke auftritt. Im Gegensatz zu normalen Mono-Flops wird jedoch der Ausgangsimpuls sowohl bei positiven (ansteigenden) als auch bei negativen (abfallenden) Flanken erzeugt.
Mit dem Ausgangssignal des Mono-Flops (Kurve E) ist schließlich das gewünschte Signal erzeugt worden. Wenn man die Kurven A und E vergleicht, dann sieht man, daß genau zu jedem Flußwechsel ein Ausgangsimpuls erzeugt wird.

### e. Sonstiges

Bei den meisten Systemen können mehrere Diskettenlaufwerke an einen Controller angeschlossen werden (beim Apple bis zu zwei). Die Auswahl des gewünschten Laufwerks erfolgt durch die Leitung „Select" bzw. „Enable".
Bei Geräten mit zwei Schreib-/Leseköpfen (Doppelkopf-Laufwerken) wird über die Leitung „Head Select" der gewünschte Kopf ausgewählt. Das Apple-Laufwerk hat nur einen Schreib-/Lesekopf.
Zur Abfrage, ob eine Diskette schreibgeschützt ist oder nicht, verfügen die Laufwerke über eine Lichtschranke oder einen Schalter. Das Apple-Laufwerk verwendet einen Schalter. Bei Geräten mit Lichtschranke muß die Kerbe in der Diskettenhülle natürlich mit einem lichtundurchlässigen Material zugeklebt werden.
Die Diskettenlaufwerke mit Normschnittstelle verfügen über einen Schalter oder eine Lichtschranke zur Erkennung, ob der Schreib-/Lesekopf auf Spur 0 steht. Das „Track 0"-Signal (Spur 0-Signal) wird nur aktiviert, wenn der Schreib-/Lesekopf auf Spur 0 steht. Auf allen anderen Spuren ist das „Track 0"-Signal inaktiv. Die Apple-Laufwerke haben keine Erkennung für Spur 0.
Geräte mit Normschnittstelle haben schließlich noch eine Lichtschranke zur Erkennung der Index- und Sektorlöcher. Jedesmal, wenn ein Index- oder Sektorloch die Lichtschranke passiert, wird der Phototransistor der Lichtschranke leitend und ein Impuls auf der „Index"-Leitung erzeugt. Die Apple-Laufwerke haben keine Lichtschranke zur Index-/Sektor-Erkennung.

## 3. DER DISKETTEN-CONTROLLER

Der Controller stellt die Verbindung zwischen Disketten-Laufwerk und Computer her. Aufbau und Funktion des Controllers ist von System zu System sehr unterschiedlich. Zahlreiche Controller benutzen hochintegrierte IC's zur Steuerung der

meisten Funktionen. Beim Apple ist der Controller hardwaremäßig relativ einfach aufgebaut, was dadurch erreicht wird, daß die meisten Funktionen per Software ausgeführt werden. Die erforderlichen Routinen sind im Betriebssystem enthalten. Beim DOS 3.3 sind diese Routinen in der sog. RWTS-Routine (= Read/Write Track/Sector = Spur/Sektor Lesen/Schreiben) zusammengefaßt. Wenn nachfolgend die Funktionen eines Controllers allgemein beschrieben werden, so bezieht sich das beim Apple auf Controller-Hardware und RWTS-Software zusammen.

### a. Das Selektieren

Unter Selektieren versteht man die Auswahl eines Laufwerks, wenn mehrere Laufwerke an den Controller angeschlossen sind. Dazu wird dem Controller die Laufwerksnummer übergeben. Der Controller aktiviert dann das zugehörige „Select"- bzw. „Enable"-Signal, wodurch das gewünschte Laufwerk eingeschaltet wird. Das selektierte Laufwerk führt die über die Schnittstelle übertragenen Befehle aus und sendet die Ergebnisse über die Schnittstelle an den Controller zurück.
Sind an den Controller Laufwerke mit zwei Schreib-/Leseköpfen angeschlossen, wird zusätzlich über die Leitung „Head Select" der gewünschte Kopf ausgewählt.

### b. Das Positionieren

Zum Positionieren wird dem Controller die gewünschte Spurnummer übergeben. Der Controller erzeugt daraus die erforderlichen Schnittstellensignale (bei Geräten mit Normschnittstelle „Step" und „Direction" oder beim Apple direkt die vier Phasensignale).

### c. Das Positionieren auf Spur 0

Das Positionieren auf Spur 0 (Rezero, Recalibrate, Restore) ist eine besondere Positionierroutine, die benutzt wird, wenn das System nicht weiß, auf welcher Spur der Schreib-/Lesekopf steht, wie z.B. beim Einschalten oder Urladevorgang (Booten).
Beim Positionieren auf Spur 0 wird solange nach außen positioniert, bis die Spur 0 erreicht ist. Bei Geräten mit Normschnittstelle ist das der Fall, wenn das „Track 0"-Signal aktiv wird. Von da an hat der Schreib-/Lesekopf eine bekannte Position, und der Controller kann jede gewünschte Spur gezielt anfahren.
Das bekannte „Rappeln" beim Einschalten des Apple rührt daher, daß das Apple-Laufwerk keine Erkennung für die Spur 0 hat. Der Apple macht deshalb soviele Schritte nach außen, daß auch im ungünstigsten Fall die Spur 0 erreicht wird. Wenn der Kopf nicht ganz innen stand, wird die Spur 0 schon eher erreicht und der Positioniermechanismus stößt solange gegen einen mechanischen Anschlag, bis die vorgegebene Anzahl von Schritten ausgeführt ist.

### d. Das Formatieren

Aus Zeit- und Speicherplatzgründen ist es unzweckmäßig, immer eine ganze Spur einer Diskette auf einmal zu schreiben oder zu lesen. Man unterteilt deshalb jede Spur in mehrere Abschnitte, die **Sektoren** genannt werden.
Bei hardsektorierten Disketten ist die Lage und Länge der einzelnen Sektoren durch die Sektorlöcher festgelegt.
Bei softsektorierten Disketten wird die Unterteilung in Sektoren durch die Formatierung vorgenommen. Jeder Sektor besteht aus zwei Teilen, dem **Adreßfeld** (address field, header, ID record) und dem **Datenfeld** (data field, data field record). Die Adreßfelder enthalten u.a. die Spur- und Sektornummer des betreffenden Sektors und eine Prüfsumme. Die Datenfelder enthalten u.a. die Benutzerdaten (die eigentliche Nutzinformation) und ebenfalls eine Prüfsumme. Die **Prüfsumme** im Adreß- und Datenfeld wird bei der Aufzeichnung nach einem bestimmten Verfahren aus den Daten errechnet und unmittelbar nach den Daten aufgezeichnet. Beim Lesen wird erneut die Prüfsumme aus den gelesenen Daten errechnet und mit der aufgezeichneten Prüfsumme verglichen. Stimmen beide Werte überein, so ist das betreffende Adreß- bzw. Datenfeld mit sehr großer Wahrscheinlichkeit fehlerfrei gelesen worden.
Das genaue Format der Adreß- und Datenfelder ist für das FM- und MFM-Aufzeichnungsverfahren ausführlich in dem Aufsatz von Klein, R. D.: „Floppy-Disk-Aufzeichnungsverfahren", mc 7/1984, S. 38ff., beschrieben. Für das beim Apple verwendete Aufzeichnungsverfahren folgt die Beschreibung in einem späteren Teil dieser Artikelserie.
Während der **Formatierung** werden die Adreßfelder für alle Sektoren auf die Diskette geschrieben. Zwischen den Adreßfeldern wird der erforderliche Platz für die Datenfelder freigehalten. Bei der späteren Benutzung der Diskette werden die Adreßfelder nicht wieder überschrieben, sondern nur die Datenfelder. Beim Apple DOS 3.3 ist das Formatieren eine Teilfunktion des Initialisierens (INIT-Befehl).

### e. Das Aufzeichnungsverfahren

Unter Aufzeichnunsverfahren versteht man die Umsetzung (Codierung) der Daten in die Aufzeichnung auf der Diskette und zurück. Vom System her werden die Daten parallel in 8-Bit-Bytes oder 16-Bit-Worten übertragen. Vom Controller werden sie zunächst in eine serielle Bitfolge umgewandelt. Dabei wird meist das höchstwertige Bit (MSB = most significant bit) zuerst übertragen. Anschließend werden die Datenbits über die Hardware des Controllers und des Laufwerks in eine Folge von Flußwechseln umgesetzt.

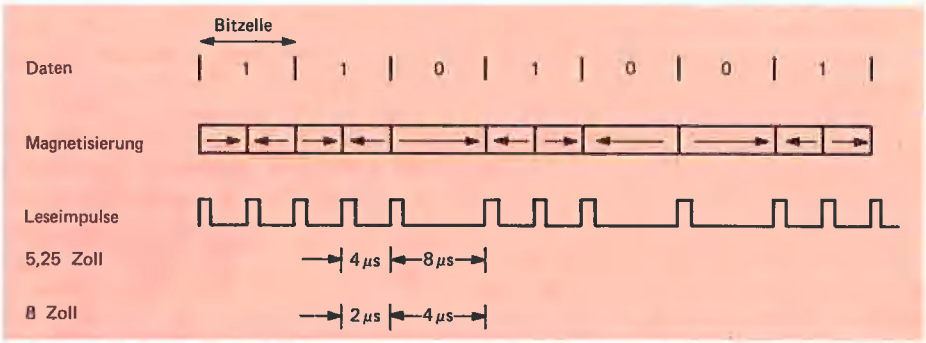



*Bild 12: FM Aufzeichnungsverfahren*

Die gebräuchlisten Aufzeichnungsverfahren sind **FM** (frequency modulation = Frequenzmodulation) für einfache Dichte und **MFM** (modified frequency modulation = modifizierte Frequenzmodulation) für doppelte Dichte.
**Bild 12** zeigt die Codierung bei FM. Die erste Zeile enthält die zu codierenden Daten. Jedem Bit ist eine feste Zeit, die sog. **Bitzelle**, zugeordnet. Bei 5,25-Zoll-Disketten und FM ist jede Bitzelle nominal 8$\mu$sec lang. Die zweite Zeile zeigt die Ma-

gnetisierung der Diskette und die dritte Zeile die Lesesimpulse.
In der Mitte jeder Bitzelle erfolgt ein Flußwechsel, wenn das betreffende Datenbit eine 1 ist. Wenn das Datenbit eine 0 ist, so ist in der Mitte der Bitzelle kein Flußwechsel. Zusätzlich erfolgt an jeder Grenze zwischen zwei Bitzellen ein Flußwechsel. Die Flußwechsel in der Mitte der Bitzellen werden auch als **Datenflußwechsel** bezeichnet, da sie direkt den Daten zugeordnet sind. Die Flußwechsel an den Grenzen der Bitzellen werden als **Taktflußwechsel** bezeichnet, da sie zur Erzeugung eines Taktsignales benutzt werden. Wie man aus der Zeichnung sieht, kommen bei FM Flußwechselabstände von 4$\mu$sec und 8$\mu$sec vor.

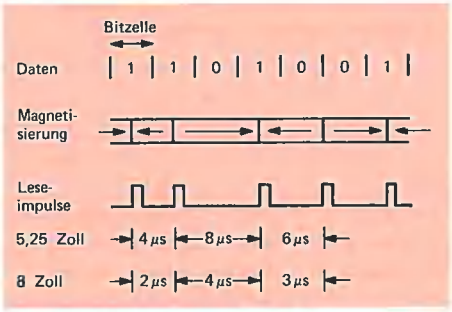



*Bild 13: MFM Aufzeichnungsverfahren (doppelte Dichte)*

**Bild 13** zeigt die Codierung für **MFM** bei gleichen Daten. Die Definition der Datenflußwechsel ist die gleiche wie bei FM. Taktflußwechsel erfolgen bei MFM jedoch nur an den Grenzen zwischen zwei aufeinanderfolgenden 0 Bits. Bei 5,25-Zoll-Disketten und MFM ist jede Bitzelle nur 4$\mu$sec lang (deshalb auch doppelte Dichte). Damit ergeben sich, wie Bild 13 zeigt, mögliche Flußwechselabstände von 4$\mu$sec, 6$\mu$sec oder 8$\mu$sec. Wichtig ist, daß bei MFM der gleiche Bereich (4$\mu$sec bis 8$\mu$sec) wie bei FM benutzt wird, d.h. bei gleicher Flußwechseldichte hat MFM die doppelte Bitdichte.
Bei 8-Zoll-Laufwerken sind alle Zeiten bei der Aufzeichnung genau halb so lang wie bei 5,25-Zoll-Laufwerken.
Beim Apple wird weder FM noch MFM benutzt, sondern ein eigenes Aufzeichnungsverfahren, das ausführlich in einem späteren Teil der Artikelserie beschrieben wird.

### f. Das Schreiben und Lesen

Das Schreiben auf die Diskette ist relativ einfach. Dazu brauchen nur die Datenbits – je nach Aufzeichnungsverfahren – in die entsprechenden Flußwechsel umcodiert und dann die Schreibelektronik des Laufwerkes entsprechend angesteuert zu werden.
Das Lesen ist sehr viel komplizierter. Das Laufwerk liefert dem Controller lediglich eine endlose Folge von Leseimpulsen mit unterschiedlichem Abstand, aus denen der Controller die ursprüngliche Information zurückgewinnen muß. Die Aufbereitung erfolgt in mehreren Stufen. Bei FM und MFM muß zunächst unterschieden werden, welche Impulse von Taktflußwechseln und welche Impulse von Datenflußwechseln herrühren. Wenn die Unterscheidung festliegt, werden die Taktimpulse unterdrückt und die Datenimpulse weiter ausgewertet. Die Trennung von Taktimpulsen und Datenimpulsen wird auch als **Datenseparierung** (data separation) bezeichnet. Die verwendete Methode und Schaltung bestimmt weitgehend die Zuverlässigkeit eines Disketten-Systems. Nachdem Takt und Daten voneinander getrennt sind, muß als nächstes gefunden werden, wo innerhalb des seriellen Datenstroms jeweils ein Byte beginnt. Als letztes muß schließlich noch der Anfang der Adreß- und Datenfelder gefunden werden. Die Erkennung von Byte-, Adreßfeld- und Datenfeldanfang erfolgt über Flußwechselmuster, die sonst nirgendwo in der Aufzeichnung vorkommen.

Auf der Ebene des Controllers erfolgen Schreiben und Lesen sektorweise. Um einen bestimmten Sektor zu schreiben oder zu lesen, muß der Schreib-/Lesekopf zunächst auf die gewünschte Spur positioniert werden.
Danach muß der Controller den gewünschten Sektor finden. Bei hardsektorierten Disketten geschieht das einfach durch Abzählen der Sektorlöcher. Bei softsektorierten Disketten muß der Controller die Adreßfelder der am Schreib-/Lesekopf vorbeikommenden Sektoren lesen. Stimmen Spur- und Sektornummer mit den vorgegebenen Werten überein, ist der richtige Sektor gefunden.
Beim Schreiben wird das Datenfeld unmittelbar hinter dem gefundenen Adreßfeld auf die Diskette geschrieben, wobei das Datenfeld, das vorher an diesem Platz stand, überschrieben wird.
Beim Lesen wird das Datenfeld, das unmittelbar hinter dem gefundenen Adreßfeld steht, gelesen und zum System übertragen.

### g. Die Zugriffszeit

Die Leistungsfähigkeit von Computersystemen hängt sehr stark davon ab, wie lange es dauert, Daten von externen Speichermedien (z.B. Disketten) zu lesen oder darauf zu schreiben. Als **Zugriffszeit** (access time) wird die Zeit bezeichnet, die vom Erteilen des Befehls zum Schreiben oder Lesen eines Sektors benötigt wird, bis der Befehl ausgeführt ist.
Die Zugriffszeit setzt sich aus folgenden, durch die Mechanik bedingten Zeitabschnitten zusammen:
– Die Zeit vom Start des Disketten-Antriebsmotors bis zum Erreichen der vollen Drehzahl.
– Bei zahlreichen Laufwerken wird der Magnet zum Andrücken der Diskette an den Schreib-/Lesekopf eingeschaltet, nachdem der Motor seine volle Drehzahl erreicht hat. Von da an wird die **Kopf-Lade-Zeit** (head load time) benötigt, bis die Diskette richtig angedrückt wird.
– Die Zeit zum Positionieren des Schreib-/Lesekopfes auf die gewünschte Spur (positioning time). Um Zeit zu sparen, kann die Positionierung gleichzeitig mit dem Starten des Antriebsmotors erfolgen. Für die Zugriffszeit maßgebend ist dann die längere der beiden Zeiten.
– Nach dem letzten Schritt des Positionierens schwingt das Positioniersystem auf Grund seiner Masse noch etwas weiter. Die Zeit, die gewartet werden muß, bis das Positioniersystem zur Ruhe gekommen ist, wird als **Beruhigungszeit** (settling time) bezeichnet.
– Erst jetzt kann der Controller anfangen, den gewünschten Sektor zu suchen. Die Zeit, die vergeht, bis der Sektor den Kopf erreicht, wird als **Latenzzeit** (latency time) bezeichnet. Im ungünstigsten Fall ist sie gleich der Umdrehungszeit der Diskette.

Werden mehrere Sektoren kurz hintereinander gelesen oder geschrieben, so kann sich die Zugriffszeit für die nachfolgenden Sektoren wesentlich verkürzen:
– Nach dem Schreiben bzw. Lesen eines Sektors wird der Disketten-Antriebsmotor meist nicht sofort abgeschaltet. Wenn der Motor beim nächsten Sektor also noch läuft, entfällt die Wartezeit für das Starten des Motors und das Andrücken der Diskette.
– Die Positionierzeit und Beruhigungszeit kann entfallen, wenn der nachfolgende Sektor auf der gleichen Spur wie der vorhergehende steht. Im günstigsten Falle werden alle Sektoren einer Spur ohne dazwischenliegende Positionierung nacheinander gelesen. Weiterhin kann die Positionierzeit verringert werden, indem die erforderlichen Positionierwege so kurz wie möglich gehalten werden.

– Die Latenzzeit kann durch die **Sektor-Schachtelung** (interleaving, interlacing, skewing) verringert werden. Dazu werden die Sektoren nicht in der Reihenfolge ihrer Nummern auf die Diskette geschrieben, sondern so, daß der nächste Sektor gerade dann den Schreib-/Lesekopf erreicht, wenn der Controller ihn lesen kann.

Zu den bisher erläuterten, durch die Mechanik bedingten Zeiten, kommen noch weitere Zeiten hinzu:
– Die Zeit zur Übertragung des Sektors. Sie hängt von der Sektorlänge und der Zeit pro Bitzelle ab.
– Die Verarbeitungszeit des Betriebssystems und des Benutzerprogramms.
– Nach dem Schreiben erfolgt meist noch ein Prüflesen, das eine bestimmte Zeit erfordert. Zum Prüflesen werden die geschriebenen Sektoren nochmals gelesen und die gelesenen Daten mit denen im Speicher verglichen oder nur die Prüfsumme kontrolliert.
– Im Falle von Schreib- oder Lesefehlern wird der betreffende Vorgang mehrfach wiederholt, wobei ggf. auf die gewünschte Spur neu positioniert wird.

Oftmals wird als Kenngröße die **Übertragungsrate** (transfer rate) angegeben. Dabei muß man zwischen zwei verschiedenen Werten unterscheiden.
Die **Block-Übertragungsrate** (burst transfer rate) ist die Geschwindigkeit, mit der die Daten während des Schreib- bzw. Lesevorgangs übertragen werden. Sie ist gleich dem Kehrwert der Zeitdauer einer Bitzelle. Bei 5,25-Zoll-Disketten ist die Block-Übertragungsrate bei FM 1/(8$\mu$sec) = 125 kBit/sec und bei MFM 1/(4$\mu$sec) = 250 kBit/sec. Bei 8-Zoll-Disketten ist die Block-Übertragungsrate bei FM 1/(4$\mu$sec) = 250 kBit/sec und bei MFM 1/(2$\mu$sec) = 500 kBit/sec.
Die **mittlere Übertragungsrate** (average transfer rate) ist der Mittelwert der Übertragungsrate über einen längeren Zeitraum einschließlich Positionierung und aller Wartezeiten. Er wird errechnet als Quotient aus der Summe der übertragenen Daten, geteilt durch die dafür benötigte Zeit.

### h. Die Kapazität einer Diskette

Bei der Angabe der Kapazität einer Diskette wird zwischen unformatiert und formatiert unterschieden.
Die **unformatierte Kapazität** gibt an, wieviele Bytes insgesamt (ohne Unterteilung in Sektoren) auf eine Diskette geschrieben werden könnten. Sie errechnet sich aus folgender Formel:

$$Ku = U \cdot N \cdot S : (B \cdot 8)$$

In der Formel ist Ku die unformatierte Kapazität in Bytes, U die Umdrehungsdauer der Diskette, N die Anzahl der Spuren, S die Zahl der Seiten und B die Zeitdauer einer Bitzelle.
Für eine einseitige 8-Zoll-Diskette mit 77 Spuren und einfacher Dichte beträgt die unformatierte Kapazität:

$$Ku = 166{,}7\ \text{msec} \cdot 77 \cdot 1 : (4\mu\text{sec} \cdot 8) = 166667\mu\text{sec} \cdot 77 : 32\mu\text{sec} = 401042$$

Die unformatierte Kapazität beträgt also ca. 400 kBytes. Bei zweiseitiger Aufzeichnung und doppelter Dichte ist sie viermal so hoch, nämlich ca. 1,6 MBytes.
Für eine einseitige 5,25-Zoll-Diskette mit 35 Spuren und einfacher Dichte (FM) errechnet sich die unformatierte Kapazität zu:

$$Ku = 200\ \text{msec} \cdot 35 \cdot 1 : (8\mu\text{sec} \cdot 8) = 200000\mu\text{sec} \cdot 35 : 64\mu\text{sec} = 109375$$

Die unformatierte Kapazität beträgt also ca. 109 kBytes.
Nach dem gleichen Verfahren kann man die unformatierte Kapazität für alle anderen Disketten ausrechnen. So ist beispielsweise die maximale unformatierte Kapazität einer 5,25-Zoll-Diskette bei doppelseitiger Aufzeichnung mit 80 Spuren und doppelter Dichte (MFM) genau 1000000 Bytes bzw. 1 MByte.

Die **formatierte Kapazität** gibt an, wieviele Benutzerdaten auf einer Diskette gespeichert werden können. Sie errechnet sich nach der Formel:

$$Kf = B \cdot M \cdot N \cdot S$$

In der Formel ist Kf die formatierte Kapazität in Bytes, B die Anzahl der Bytes pro Sektor, M die Anzahl der Sektoren pro Spur, N die Anzahl der Spuren und S die Zahl der Seiten.
8-Zoll-Disketten werden meist mit dem sogenannten IBM-Format benutzt. Dieses Format benutzt 26 Sektoren pro Spur. Bei einfacher Dichte (FM) enthält ein Sektor 128 und bei doppelter Dichte (MFM) 256 Datenbytes. Damit ist z.B. die formatierte Kapazität einer einseitigen 8-Zoll-Diskette mit einfacher Dichte:

$$Kf = 128 \cdot 26 \cdot 77 \cdot 1 = 256256$$

Die formatierte Kapazität beträgt also rund 256 kBytes. Die Ausnutzung der Diskette – das Verhältnis zwischen formatierter und unformatierter Kapazität – ist:

$$Kf : Ku = 256256 : 401042 = 0{,}639 = 63{,}9\%$$

Die restlichen 36,1% der Diskettenkapazität sind für die Formatierung (Adreßfelder usw.) benötigt worden. Die Ausnutzung ist um so besser, je länger die Sektoren sind, da entsprechend weniger Adreßfelder usw. benötigt werden.
Übliche Werte für B und M bei 5,25-Zoll-Disketten und FM sind 16 Sektoren zu 128 Bytes oder 9 Sektoren zu 256 Bytes. Bei MFM sind 16 Sektoren zu 256 Bytes oder 9 Sektoren zu 512 Bytes üblich. Manchmal werden statt 9 auch 10 Sektoren auf einer Spur untergebracht.
Bei einer einseitigen 5,25-Zoll-Diskette mit 35 Spuren und 16 Sektoren zu 128 Bytes beträgt die formatierte Kapazität:

$$Kf = 128 \cdot 16 \cdot 35 \cdot 1 = 71680$$

Die formatierte Kapazität beträgt also nur ca. 72 kBytes. Die Ausnutzung der Diskette ist:

$$Kf : Ku = 71680 : 109375 = 0{,}655 = 65{,}5\%$$

Für eine doppelseitige Diskette mit 80 Spuren und 9 Sektoren zu 512 Bytes beträgt die formatierte Kapazität:

$$Kf = 512 \cdot 9 \cdot 80 \cdot 2 = 737280$$

In diesem Fall ist die formatierte Kapazität also ca. 737 kBytes bei einer Ausnutzung von 73,7%.

Beim Apple werden normalerweise 35 Spuren mit 16 Sektoren zu 256 Bytes benutzt. Das ergibt bei einseitiger Aufzeichnung eine formatierte Kapazität von:

$$Kf = 256 \cdot 16 \cdot 35 \cdot 1 = 143360$$

Die Apple Disketten haben somit eine formatierte Kapazität von ca. 140 kBytes bei einseitiger Benutzung und ca. 280 kBytes bei doppelseitiger Benutzung.

**Da unbestritten ist, daß in der Legastheniethérapie dem Einsatz von Schreibmaschinentastaturen eine nicht zu unterschätzende Bedeutung zukommt, sollte auch hier der Versuch eines Einsatzes von Computern (Apple IIe) gemacht werden. Da es bislang keine für unsere Arbeit brauchbaren Programme gibt, mußten eigene geschrieben werden; eines davon sei hier vorgestellt.**

# Testgenerator für Legastheniker

von Dr. Wolfgang Kersken

## 1. Ausgangssituation

Grundsätzliche Überlegungen gingen davon aus, daß häufig beklagte Mängel hinsichtlich der Möglichkeiten des Computereinsatzes zu berücksichtigen und zu vermeiden seien:

1. Das Programm soll möglichst flexibel gehalten werden, damit ein Schüler individuell und hinsichtlich seiner besonderen Schwäche damit arbeiten kann.
2. Die Bedienung muß sowohl für den Lehrer als auch besonders für den Schüler einfach und unproblematisch sein. Es dürfen bei beiden keine Kenntnisse in Programmierung vorausgesetzt werden. An jeder Stelle muß transparent bleiben, was der Anwender tun soll.
3. Das Programm soll nicht den Lehrer ersetzen, sondern ihn für individuelle Betreuung freisetzen, obwohl in einer *Gruppe* gearbeitet wird.
4. Das Programm muß so gestaltet sein, daß es Bedienungsfehler nicht übelnimmt und aussteigt.
5. Es soll zumindest ein wenig gesichert sein, so daß allzu problemloses Manipulieren des Programms und der Übungstexte (= Textfiles) durch „böse Buben“ nicht einfach ist.
6. Der Computer darf nur Hilfsmittel, allenfalls Stimulanz sein, kein modernistischer Selbstzweck. Der Schüler darf sich nicht

alleingelassen oder abgeschoben vorkommen (vgl. „elektronische Großmutter" Fernsehen).
7. Das Programm und der Computer dürfen nicht ablenken von dem, was eigentlich zu geschehen hat.

Für uns im Landerziehungsheim Schule Marienau (östlich von Lüneburg) ist *Ausgangssituation*: In Gruppen von etwa fünf Schülern werden unsere Legastheniker betreut. Zwar werden diese Gruppen sorgfältig zusammengestellt, doch kann es nicht ausbleiben, daß ein einzelner in dieser Gruppe eine besondere Problematik aufweist, die gemeinsam mit allen nicht angegangen werden kann, da die anderen sie nicht haben. So ist dann der Übungsteil der Therapie weitgehend orientiert an einem Durchschnittswert, der die ganze Gruppe betrifft. Das Programm TESTGENERATOR schafft hier insofern Abhilfe, als es jetzt möglich ist, in der Übungsphase statt der Gruppenarbeit Einzelbetreuung vorzunehmen, da jeder Schüler „seine", das heißt ihm angemessene Übungsaufgaben erhält und bearbeiten kann.

## 2. Praktischer Teil

Der **Schüler** legt „seine" Diskette ins Laufwerk ein und startet das System. Ein kurzer Hinweistext gibt die wichtigen Erläuterungen; er wird später von den meisten Schülern schnell übergangen. Nun weist das Programm alle Textfiles aus, die zur Verfügung stehen und in diesem Arbeitsabschnitt behandelt werden sollen. Der Schüler gibt den Namen des Files ein, den er sich vornehmen will. Hierbei ist wichtig, daß der Schüler sich bei der Eingabe exakt an die Vorlage hält; tut er dies nicht, erhält er einen Hinweis auf seinen Fehler, und er kann seine Eingabe korrigieren. Der Übungstext wird von der Diskette geladen. In den Sätzen des Textes findet der Schüler Fehler, die er korrigieren soll; die Anzahl der Fehler wird mitgeteilt. Die Reihenfolge, in der der Schüler die Fehler – jetzt korrekt geschrieben – eingibt, ist beliebig. Das Programm vergleicht und gibt dem Schüler an, wieviele Fehler er entdeckt hat, und gibt ihm die richtige Schreibweise vor. Zu weiteren Übungszwecken oder aber auch zur Dokumentation seines Erfolges kann der Schüler die gerade gemachte Übung auch ausdrucken lassen. Hier erhält er ebenfalls die Textvorgabe, seine eigene Fehlerkorrektur, die korrekte Schreibweise und die Anzahl der Fehler, die er gefunden hat. Schließlich wird er gefragt, ob er noch eine Übung machen oder lieber aufhören möchte. Drei bis vier Übungen im Zusammenhang haben sich als durchaus sinnvoll erwiesen. Während dieser Zeit hat der Lehrer Gelegenheit, mit dem einzelnen zu arbeiten, und gleichzeitig hat jeder in der Gruppe „seine" speziellen Aufgaben.

Der **Lehrer** kann nach Eingabe des von ihm selbst bestimmten Passwortes in einen Eingabemodus gelangen. Hier gibt er seine Übungssätze mit den Fehlern ein, die korrekte Schreibweise und die Anzahl der Fehler; anschließend wird sein Textfile auf der Diskette gespeichert und zur Kontrolle ausgegeben. Es ist nicht erforderlich, für jeden Schüler eine spezielle Diskette anzulegen. Jede Diskette kann alle Textfiles enthalten. Sie müssen lediglich im Einzelfall ungeschützt (UNLOCK) werden, um nicht im CATALOG zu erscheinen. So sind dann zwar die Disketten für alle Schüler identisch (leichteres Kopieren!), jeder Schüler hat aber lediglich „sein" Inhaltsverzeichnis.

## 3. Allgemeine Programmbeschreibung

Das Programm TESTGENERATOR hat zwei Aufgaben:
1. Einmal soll es in einem kombinierten Ein- und Ausgabeteil möglich sein, Textfiles auf die Übungsdiskette zu schreiben, alte zu überarbeiten oder zu löschen. Dieser Teil ist durch ein Passwort geschützt. Es ist einzugeben, wenn auf dem Bildschirm die Frage „Alles klar (J/N)" erscheint. Jede andere Eingabe außer dem Passwort führt in den Ausgabemodus für den Schüler.
2. Ein zweiter Teil dient der Ausgabe und Bearbeitung von Textfiles durch den Schüler. Aus dem CATALOG, der nur Textfiles anzeigt und keine Programme, wählt er den File aus, den er bearbeiten möchte oder gibt den an, den er bearbeiten sollte.

Jeder einzelne Textfile kann aus bis zu zehn Aufgaben bestehen. Größere Aufgabenpartien haben sich als ungünstig erwiesen; sinnvoll sind drei oder vier Aufgaben pro Textfile.
Der Vorteil gegenüber den landläufigen Übungsdiktaten oder Schreibroutineübungen liegt darin, daß auf einen bestimmten Schüler und seine besondere Problematik hin Übungsaufgaben erstellt werden können, daß gleichzeitig mehrere Schüler so gezielter arbeiten können und der Lehrer dadurch freigesetzt wird, sich individueller um den einzelnen kümmern zu können. Daß gerade Legastheniker, besonders auch im gymnasialen Bereich, durch die Bedienung einer Schreibmaschinentastatur zusätzlich üben, ist klar; und auch die Attraktivität des „Computers" ist nicht zu unterschätzen und sollte durchaus als Stimulanz in den LRS-Unterricht eingebracht werden. In der Möglichkeit, einen einzelnen Schüler und seine besondere Problematik ansprechen zu können, sehe ich die Leistungsfähigkeit des TESTGENERATORs.
Dabei ist das Programm so gegliedert, daß es durchaus über den LRS-Unterricht hinaus auch in anderen Fächern eingesetzt werden kann. Dazu müssen vor dem Compilieren einzelne Zeilen der neuen Aufgabe angepaßt werden:

Die Programmdefinitionen von Zeile 1940 bis 1980 sind zu ändern:
**1940 Titel:** Hier ist der gewünschte Titel einzugeben, z.B. Test Geographie.
**1950 Satz/Fehler:** Statt dieser Paarung kann Frage/Antwort oder Aufgabe/Lösung treffender und damit sinnvoller sein. Analog dazu sollten Sie in 1370, 1440 und 1450 Anpassungen vornehmen, wenn es sinnvoll scheint.
**1960 Fehlerzähler:** Diese Routine gibt an, wieviele Fehler der Schüler erkannt hat. Es kann in anderem Zusammenhang durchaus richtig sein, hier die Höchstzahl zu hinterlegen. Hinsichtlich der Motivationsverstärkung scheint die gewählte Form für den LRS-Unterricht sinnvoll. Sie dürfen bei einer Änderung nicht vergessen, auch Zeile 1630 anzupassen, denn jetzt wäre ja zu vermindern, also: 1630 Z = Z - 1: RETURN.
**1970 Leerdatei:** Programminterner Name zum Schutz vor versehentlichen Löschungen; sollte nicht geändert werden.
**1980 Passwort:** Derzeit gilt als Passwort „LEPGM" (= LEhrerProGramM); bitte ändern. Dieses Passwort schützt Sie zumindest ein wenig davor, daß ein Unbefugter in Ihren ausgearbeiteten Textfiles herumhantiert. Ihr Passwort kann beliebig lang sein und aus Groß- und Kleinbuchstaben bestehen (Bitte nicht zu kompliziert; Sie vergessen's sonst selbst).

Folgende Modifikationen sollten Sie **nicht** vornehmen: Erhöhen Sie nicht die Aufgabenzahl pro Textfile; ich arbeite im Regelfall mit 3 bis 4 Aufgaben pro File. Ein Versuch, das Programm zu beschleunigen, bringt nichts ein, da der Schüler die Pausen zwischendurch, besonders beim Einlesen von der Diskette, braucht. Die 80-Zeichen-Karte ist nicht aktiv: Legastheniker haben ohnehin schon genug Schwierigkeiten beim Lesen. „Beladen" Sie für die Anfangszeit den CATALOG nicht mit mehr als 20 sichtbaren Einträgen, damit der Schüler wenigstens in der ersten Zeit das Wort, das er eingeben soll, noch sieht (und daß Sie ihn nicht mit Ctrl-Tasten ärgern, versteht sich von selbst!).

## 4. Erstellen der Übungsdisketten

1. Nehmen Sie als erstes alle Änderungen am Programm vor, die Sie als für Ihre Aufgabenstellung nötig erachten.
2. Compilieren Sie das Programm. Heben Sie sich auf jeden Fall das Applesoft-Programm auf, falls Sie noch andere Modifikationen vornehmen wollen.
3. Kopieren Sie auf eine Übungsdiskette die Library des TASC-Compilers namens RUNTIME und das compilierte Programm; sprechen Sie beide durch z.B. folgendes HELLO-Programm an:

```
10 VTAB (13): HTAB(10): PRINT "LRS-TESTS"
20 PRINT CHR$ (4) + "BLOAD RUN-TIME" + CHR$ (13) + CHR$ (4) + "BRUN TESTGENERATOR"
```

4. Geben Sie danach die Textfiles auf die Disketten, die Sie für bestimmte Schüler vorgesehen haben. Sie können auch bei identischen Disketten für Schüler mit LOCK und UNLOCK arbeiten. Die Textfiles auf einer Diskette können unterschiedlich lang sein, da die Anzahl der Aufgaben mit abgespeichert wird. Im Laufe der Zeit werden Sie eine Bibliothek von Files erhalten, aus der Sie auswählen können. HELLO, RUNTIME und TESTGENERATOR dürfen nicht geLOCKt sein; hingegen müssen die Textfiles geLOCKed werden (wichtig für den CATALOG-Ausdruck). Geben Sie die Textfiles im Eingabemodus vom TESTGENERATOR ein, so werden diese automatisch geLOCKt.
5. Kopieren Sie diese Diskette für jeden Schüler.

Ich wünsche Ihnen viel Erfolg.


Wollen Sie sich das Eintippen des folgenden Listings ersparen, so können Sie von uns eine Diskette mit dem Quell-Code, dem compilierten Programm und Beispielfiles aus verschiedenen Unterrichtsfächern beziehen. (Die Peeker-Sammeldiskette enthält die nicht-compilierte Applesoft-Version sowie 5 Beispielfiles „Bahnfahrt", „Zu", „Tun und sollen", „Irgend" und „Sätze". Anm. der Red.).

## 5. Detaillierte Programmbeschreibung

**0 ff.:** Der *Listschutz* durch POKE 214,130 soll unnötiges „Spielen" und unerwünschte Manipulation an den Tests erschweren; dem dient auch das Passwort und das „Verbergen" von Programmen im CATALOG.
**140 ff.:** Wollen Sie in den (geschützten) Eingabemodus, geben Sie hier Ihr **Passwort** ein.
**250 ff.:** Die *Programmverzweigung* in Zeile 250 trennt zwischen einem reinen Ausgabeteil für den Schüler und einem kombinierten Ein- und Ausgabeteil für den Lehrer. Dazu ist ein Passwort einzugeben, das in Zeile 1980 festgelegt werden kann; derzeit gilt „LEPGM". Während der Eingabe werden für jedes Zeichen Nullen ausgedruckt (Zeile 260).
**350 ff.:** Bei Modifikationen hinsichtlich des Einsatzes ist „Aufgaben" in Zeile 510 und „Antworten" in 660 möglicherweise anzupassen und zu ändern, vgl. auch weiter unten. Die Anzahl der Aufgaben ist identisch mit der Anzahl der Fehler.
**520 ff.:** Haben Sie bei Ihrer Eingabe einen *Fehler* entdeckt, nachdem Sie RETURN gedrückt haben, können Sie Ihre Eingabe durch ESCAPE löschen.
**560 ff.:** Hinterlegen Sie an dieser Stelle die korrekte Schreibweise bzw. die Antwort, die der Schüler zu geben hat. Beachten Sie, daß der Computer zwischen „Köln", „KÖLN" und „Koeln" unterscheidet; geben Sie „KÖLN" als richtig ein, so liegt der Schüler mit Köln falsch!
**670 ff.:** Soll oder kann kein *Drucker* verwendet werden, sind die Zeilen 1070 bis 1090 zu löschen.
**1140 ff.:** POKE 44513, 67 zeigt im CATALOG nur geLOCKte Files an.
**1370 ff.:** „Fehler" in Zeile 1370, „Antworten" in 1440 und „Lösungen" in 1450 sind gegebenenfalls der Aufgabenstellung des Programms anzupassen.
**1380 ff. und 1540 ff.:** Diese Subroutine soll verhindern, daß das Programm bei den am häufigsten auftretenden Fehlbedienungen aussteigt.
**1940 ff.:** Der Titel in Zeile 1940 ist eigenen Bedürfnissen anzupassen. „Satz" und „Fehler" in Zeile 1950 können ersetzt werden durch z.B. „Frage"/„Antwort"; „Aufgabe"/„Lösung", je nach Aufgabenstellung. Das Passwort in Zeile 1980 sollte vor einer Compilierung neu bestimmt werden.
**2000 ff.:** Dieses Unterprogramm ab Zeile 2000 kann entfallen, wenn ein Drucker nicht eingesetzt werden soll oder kann; auch Zeile 1720 ist dann zu löschen.

### Beispiel eines Textfiles

Ein stückeschreibender Autor wollte einen feuerspeienden Berg besuchen. Eine Schmerz lindernde und Blut stillende Arznei gab man nach dem Unfall dem Hecke schneidenden Gärtner. Ein Fleisch fressender Hund traf eine fleischfressende Pflanze. (Haben Sie die drei Schreibfehler gefunden?)


# APPLE
# MASCHINEN SPRACHE

## Der Einstieg über BASIC!

**Die Idee:** Maschinenbefehle des Mikroprozessors 6502 über die BASIC-Befehle POKE, CALL, PEEK im Apple abspeichern, ablaufen lassen und Ergebnisse in das BASIC-Programm zurücklesen. Grundkenntnisse in Basic sind erforderlich. Anschauliche Darstellung der 6502-Funktionen, der Speicher im Apple und der Wirkungen von Maschinenbefehlen – verbunden mit Experimentierprogrammen zur Grafik, Akustik, Arithmetik, Textdarstellung, Verbindung von Bild und Ton sowie zum Aufruf der zahlreichen, bereits vorhandenen Maschinenprogramme im Apple.
**Der Höhepunkt:** Eine schrittweise Einführung in die professionelle Entwicklung von Maschinenprogrammen mit APPLE SYSTEM-MONITOR und APPLE MINI ASSEMBLER.
**Von D. Inman/K. Inman, 224 Seiten, Softcover, DM 49,–**

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

te·wi

# Weiterführende Literatur...


**APPLE II –**
**Anwenderhandbuch**
(L. Poole)
Erst mit Hilfe dieses Leitfadens werden Sie Ihren Apple II erfolgreich einsetzen, denn Text und Bildmaterial gehen weit über das hinaus, was herstellerseitig an Literatur angeboten wird.
416 Seiten, Softcover, DM 56,–


**LOGO –**
**Jeder kann programmieren**
(Daniel Watt)
Buch des Jahres in den USA. Für die Computer C64, ATARI, APPLE II, IBM-PC und TI-99.
Hochwertiges Textbuch für Logo-Kurse für zu Hause und im Lehrbereich.
A4, DM 59,– (4. Quartal 84)


**APPLE II PASCAL –**
**Eine praktische Anleitung**
(A. Luehrmann, H. Peckham)
Unentbehrlich für alle, die die Programmiersprache PASCAL lernen wollen und Zugang zu einem Apple-Computer haben.
544 Seiten, Softcover, DM 59,–


**APPLE II – Bewegte 3D-Graphik**
(Phil Cohen)
Selbstentworfene Graphiken und Diagramme – animiert oder als Standbilder – eben oder räumlich: alle erforderlichen BASIC-Programme mit Erklärung finden Sie in diesem Buch.
ca. 190 Seiten, Softcover, DM 49,–
(2. Quartal 85)


**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.
A4 quer, Fadenheftung, DM 29,80


**Macintosh Programmier-Handbuch**
(David Lien)
Macintosh – Pionier des graphischen Computerdialogs – benutzt MICROSOFT BASIC. D. Lien's Buch ist eine souveräne, zuverlässige Einführung und Referenz zu Macintosh's Basic.
450 Seiten, DM 59,– (4. Quartal 84)

tm 4104

**Noch im Programm: 6502 – Programmieren in Assembler DM 59,–**
**VisiCalc, 50 Programme auf Diskette, DM 79,–**

## TESTGENERATOR

```
  0 POKE 214,130: REM Listschutz
100 REM ************************************
110 REM *  Testgenerator, Vers. LRS         *
120 REM *  Wolfgang Kersken, Marienau 1984 *
125 REM *  getestet Marquard, Hamburg      *
130 REM ************************************
140 GOSUB 1920: REM Definitionen
150 DIM S$(300),R$(300)
160 ONERR GOTO 1820
169 :
170 REM Titel Bildschirm
180 TEXT : HOME
190 INVERSE : VTAB 13: HTAB ((39 - LEN (TE$)) / 2):
    PRINT TE$: NORMAL
200 PRINT : PRINT "   .          WolKe 1984"
210 PRINT : PRINT "Schule Marienau, 2121 Dahlem -
    Marienau"
220 GOSUB 2180: REM Anlage Leerdatei
229 :
230 REM Programmverzweigung
240 VTAB 22: PRINT "         Alles klar (J/N) ?";
250 GET JN$
260 PRINT JN;:PN$ = PN$ + JN$
270 IF JN$ = CHR$ (13) THEN 290
280 GOTO 250
290 IF PN$ = CODE$ + CHR$ (13) THEN 350
300 IF PN$ = "N" + CHR$ (13) THEN 330
310 IF PN$ = "J" + CHR$ (13) OR PN$ = "" + CHR$ (13)
    THEN 930
320 GOTO 930
330 HOME : VTAB 12: HTAB 14: PRINT "Bitte melden!": END
339 :
340 REM Hauptprogramm
349 :
350 REM Ein- Ausgabe
360 HOME : INVERSE : PRINT "LEHRER-PROGRAMM": NORMAL
370 VTAB 8: PRINT "      Möchten Sie"
380 PRINT : PRINT "1 ==> Neue Tests eingeben oder"
390 PRINT : PRINT "2 ==> Tests ausgeben (ohne Hinweis)"
400 PRINT : PRINT "3 ==>                (mit Hinweis)"
410 PRINT : PRINT "4 ==> Textfiles löschen"
420 PRINT : PRINT "      (1 - 4) ";
430 GET F$
440 IF F$ = "1" OR F$ = "2" OR F$ = "3" OR F$ = "4" THEN
    460
450 GOTO 430
460 F = VAL (F$)
470 ON F GOTO 480,1120,930,2250
479 :
480 REM Eingabe von Textfiles
490 ONERR GOTO 1820
500 HOME : PRINT "Maximal 10 Sätze !": PRINT
510 INPUT "Wie viele Aufgaben ? ";AF
520 IF AF > 10 THEN AF = 10
530 GOSUB 1920: REM Definitionen
540 HOME
550 PRINT "<ESC> = Neueintrag"
560 FOR I = 1 TO AF
570 PRINT : PRINT I;". ";TW$;":"
580 PRINT
590 GET A$
600 IF A$ = CHR$ (27) THEN 2230
610 IF A$ = CHR$ (13) THEN 640
620 PRINT A$;:S$(I) = S$(I) + A$
630 GOTO 590
640 PRINT : NEXT I
650 PRINT
660 PRINT : PRINT "Welche Antworten erwarten Sie? ":
    PRINT
670 FOR I = 1 TO AF
680 PRINT I;". : "
690 GET B$
700 IF B$ = CHR$ (13) THEN 730
710 PRINT B$;:R$(I) = R$(I) + B$
720 GOTO 690
730 PRINT : NEXT I
740 PRINT : INPUT "Test-Name: ";NN$
750 GOTO 1650: REM Kontrolle
760 PRINT D$;"OPEN";NN$
770 PRINT D$;"WRITE";NN$
780 PRINT AF
790 FOR I = 1 TO AF
800 PRINT S$(I)
810 NEXT I
820 FOR I = 1 TO AF
830 PRINT R$(I)
840 NEXT I
850 PRINT
860 PRINT D$;"CLOSE";NN$
870 PRINT D$;"LOCK";NN$
880 HOME
890 INPUT "Weitere Eingaben J/N?";FF$
900 IF FF$ = "J" OR FF$ = "j" THEN RUN 480
910 IF FF$ = "N" OR FF$ = "n" THEN 1290: REM Probelauf
920 GOTO 890
929 :
930 REM Hinweise für Schüler
940 HOME : INVERSE : PRINT "SCHÜLER-PROGRAMM"
950 PRINT : PRINT "HINWEISE:": NORMAL : PRINT
960 PRINT "Hier kannst du deine Rechtschreibkennt-"
970 PRINT "nisse überprüfen. Suche dazu gleich aus"
980 PRINT "der Auswahl den kurzen Text, den du be-"
990 PRINT "arbeiten willst. In jedem Text sind"
1000 PRINT "Fehler, die du finden sollst. Gib die"
1010 PRINT "richtige Schreibweise ein und beende"
1020 PRINT "jede Antwort mit <RETURN>. Die Reihen-"
1030 PRINT "folge deiner Antworten ist beliebig."
1040 PRINT : PRINT "Am Ende sage ich dir, welche
     Antworten"
1050 PRINT "richtig waren, und zeige dir die Lösun-"
1060 PRINT "gen."
1070 PRINT : PRINT "Wenn ein Drucker angeschlossen ist, "
1080 PRINT "kannst du auch einen Ausdruck zur wei-"
1090 PRINT "teren Übung erhalten."
1100 PRINT : PRINT "Viel Erfolg und Spaß!"
1110 GOSUB 1560
1119 :
1120 REM Ausgabe
1130 HOME : PRINT
1140 POKE 44513,67: PRINT CHR$ (4);"CATALOG"
1150 PRINT : INPUT "Titel: ";NN$
1160 PRINT D$;"OPEN";NN$
1170 PRINT D$;"READ";NN$
1180 INPUT AF
1190 FOR I = 1 TO AF
1200 GET A$:S$(I) = S$(I) + A$: IF A$ = CHR$ (13) THEN
     1230
1210 GOTO 1200
1220 INPUT S$(I)
1230 NEXT I
1240 PRINT
1250 FOR I = 1 TO AF
1260 INPUT R$(I)
1270 NEXT I
1280 PRINT D$;"CLOSE";NN$
1289 :
1290 REM Textausdruck Bildschirm
1300 HOME
1310 PRINT "Bearbeite jetzt bitte '";NN$;"'!"
1320 PRINT "----------------------------------------"
1330 PRINT
1340 FOR I = 1 TO AF
1350 PRINT S$(I)
1360 NEXT I
1370 PRINT : PRINT "Im Text sind ";AF;" Fehler: "
1380 PRINT
1390 FOR J = 1 TO AF
1400 PRINT J;". ==> ";: INPUT AN$(J)
1410 X$ = AN$(J): GOSUB 1590: REM Zählung
1420 NEXT J
1430 PRINT
1440 PRINT "Du hast ";Z;" Antworten gewußt!"
1450 PRINT "Die richtigen Lösungen heißen: "
1460 FOR I = 1 TO AF
1470 PRINT R$(I)
1480 NEXT I
1490 GOSUB 1560
1500 GOSUB 1690
1510 CLEAR : GOSUB 1920
1520 GOTO 1130
1529 :
1530 REM ************************************
1540 REM *          Subroutinen             *
1550 REM ************************************
1559 :
1560 REM Fortsetzung
1570 INVERSE : VTAB 24: INPUT "WEITER MIT <RETURN> ";AA$:
     NORMAL
1580 HOME : RETURN
1589 :
```

## Für Ihre Unterlagen

Abonnement bestellt

**am:** _____

**Vertrauensgarantie:**
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Verlag innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs.

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

# peeker
MAGAZIN FÜR APPLE-COMPUTER

## Abo-Karte

**Ja,** ich möchte »**peeker**« abonnieren.

Liefern Sie mir »**peeker**« ab der nächsten Ausgabe zum **Einführungspreis** von nur **DM 58,–**. 1985 erscheinen 11 Ausgaben (1 Doppelheft). Ich spare dabei gegenüber dem Einzelkauf genau DM 13,50. Es entstehen mir keine weiteren Kosten. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Einführungspreis für das Ausland beträgt DM 58,– zuzüglich Versandspesen.

Ich wünsche jährliche Berechnung

☐ durch Verlagsrechnung oder ☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank/PschA _____

Bankleitzahl _____ Kto.-Nr. _____

Datum _____ Unterschrift _____

## Für Ihre Unterlagen

Folgende Bücher bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

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

peeker
MAGAZIN FÜR APPLE-COMPUTER

## Buch-Shop

Bitte senden Sie mir gegen Rechnung folgende Bücher:

☐ Apple Assembler, DM 34,–
☐ Apple II Basic Handbuch, DM 32,–
☐ Apple DOS 3.3, DM 28,–
☐ Apple II leicht gemacht, DM 28,–
☐ Apple Maschinensprache, DM 49,–
☐ Apple ProDOS, Band 1, DM 28,–
☐ Arbeiten mit dem Macintosh, DM 54,–
☐ BASIC-Übungen für den Apple, DM 38,–

Datum _____ Unterschrift _____

## Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

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

peeker
MAGAZIN FÜR APPLE-COMPUTER

## Software-Karte

Bitte senden Sie mir
gegen Rechnung folgende Apple-Programme:

☐ Peeker-Sammeldiskette, Heft 1–2, 1984, DM 28,–
☐ Peeker-Sammeldiskette, Heft 1–2, 1985, DM 28,–
☐ Peeker-Sammeldiskette, Heft 1–2, 1985, Steuererklärung 1984, CP/M, DM 28,–
☐ Apple DOS 3.3, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 1, Begleitdiskette, DM 28,–
☐ Apple Assembler, Begleitdiskette, DM 28,–
☐ ProDOS-Editor 1.0, Programm, DM 98,–
☐ MMU 2.0, Programm, DM 98,–
☐ INPUT 2.0, Programm, DM 98,–
☐ Softbreaker 1.0, Programm, DM 48,–
☐ DB-Meister, Programm, DM 290,–
☐ Superplot, Programm, DM 48,–

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 Verlag innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs.

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.

POSTKARTE

**peeker**
Leserservice

Postfach 10 28 69

6900 Heidelberg 1

## Buch-Shop

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

**peeker**
Versandbuchhandlung

Postfach 10 28 69

6900 Heidelberg 1

## Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

**peeker**
Softwareabteilung

Postfach 10 28 69

6900 Heidelberg 1

```
1590 REM Fehlerzählung
1600 FOR I = 1 TO AF
1610 IF X$ = R$(I) THEN 1630
1620 NEXT I: RETURN
1630 Z = Z + 1: RETURN
1640 REM Frage Prüfung
1650 PRINT : PRINT "Alles korrekt (J/N) ";: INPUT WF$
1660 IF WF$ = "J" OR WF$ = "j" THEN HOME : GOTO 760
1670 IF WF$ = "N" OR WF$ = "n" THEN HOME : GOTO 2240
1680 GOTO 1650
1690 REM Weiter/Ende
1700 HOME
1710 VTAB 11: PRINT "     W = neuer Text"
1720 PRINT : PRINT "     D = Ausdruck des letzten Textes"
1730 PRINT : PRINT "     E = Ende": PRINT
1740 PRINT "          ";
1750 GET WE$: IF WE$ = "W" OR WE$ = "w" THEN HOME :
     RETURN
1760 IF WE$ = "D" OR WE$ = "d" THEN 2000
1770 IF WE$ = "E" OR WE$ = "e" THEN 1790
1780 GOTO 1690
1790 HOME : VTAB 15: HTAB 14: PRINT "Tschüß ... ": FOR I
     = 1 TO 1500: NEXT
1800 HOME
1810 END
1820 REM Fehler
1830 FE = PEEK (222)
1840 IF FE = 255 THEN HOME : END
1850 IF FE = 5 THEN FE$ = "BITTE RICHTIG SCHREIBEN!":
     GOTO 1880
1860 IF FE = 11 THEN GOTO 1120
1870 FE$ = "BITTE AUFPASSEN!"
1880 PRINT CHR$ (7): HOME : VTAB 15: FLASH
1890 PRINT FE$
1900 FOR I = 1 TO 3000: NEXT : NORMAL
1910 PRINT D$;"DELETE";NN$: GOSUB 1920: GOTO 1130
1920 REM Definitionen
1930 D$ = CHR$ (4)
1940 TE$ = "LEGASTHENIKERTESTS": REM Programmtitel
1950 TW$ = " SATZ":TX$ = " FEHLER"
1960 Z = 0: REM Fehlerzähler
1970 NN$ = "Leerdatei"
1980 CODE$ = "LEPGM": REM Passwort
1990 RETURN
2000 REM Ausgabe Drucker
2010 PR# 1
2020 PRINT "TESTNAME: ";NN$
2030 PRINT : PRINT "DIE VORGABEN:": PRINT
2040 FOR I = 1 TO AF
2050 PRINT I;". Satz:"
2060 PRINT S$(I)
2070 NEXT I
2080 PRINT : PRINT "RICHTIG WAR: ": PRINT
2090 FOR I = 1 TO AF
2100 PRINT I;". ";R$(I)
2110 NEXT I
2120 PRINT : PRINT "DEINE ANGABE:": PRINT
2130 FOR I = 1 TO AF
2140 PRINT I;". ";AN$(I)
2150 NEXT I
2160 PRINT : PRINT "RICHTIG:     ";Z
2170 PR# 0: HOME : GOTO 1500
2180 REM Anlage Prüfdatei
2190 PRINT D$;"OPEN";NN$
2200 PRINT D$;"WRITE";NN$
2210 PRINT D$;"CLOSE";NN$
2220 RETURN
2230 REM Fehlerkorrektur Eingabe
2240 CLEAR : GOSUB 1920: GOTO 480
2250 REM Textfiles löschen
2260 HOME
2270 PRINT
2280 D$ = CHR$(4): REM CTRL-D
2290 POKE 44513,67: PRINT
2300 PRINT D$;"CATALOG"
2310 PRINT
2320 VTAB 23: INPUT "Welches File löschen ? ";NM$
2330 HOME : VTAB 12: PRINT "Sie löschen jetzt '";NM$;"'
     (J/N) ";
2340 INPUT AS$
2350 IF AS$ = "J" OR AS$ = "j" THEN 2370
2360 IF AS$ < > "J" OR AS$ < > "j" THEN 350
2370 PRINT CHR$ (4);"UNLOCK";NM$
2380 PRINT CHR$ (4);"DELETE";NM$
2390 GOTO 350
```

# DB-MEISTER

**Adreß- und Schemabriefprogramm**

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

*Der DB-Meister dient zum Anlegen, Pflegen, Sortieren, Selektieren und Ausdrucken von Dateien aller Art. Als Apple-Benutzer wissen Sie, wie langsam viele Programme dieser Art sind. Nicht so der DB-Meister!*

Drei Beispiele:

- Jeder beliebige von 560–999 Records wird nach Indexfeldern in 0,2 Sekunden gefunden.
- Eine komplette Datendiskette mit z. B. 600 Records läßt sich in 1 Minute nach 3 Feldern sortieren und untersortieren. Dabei ist die Zeit für Diskettenzugriff bereits mitgerechnet.
- Das Einlesen eines 50 Sektoren langen Programm-Moduls dauert nur 3,5 Sekunden.

*Technische Daten des DB-Meisters*

- Recordlänge bis zu 230 Zeichen
- 560 bis 1000 Records pro Datendiskette
- Maximal 25 Felder pro Record
- 4 Datentypen (String, Integer, Dezimalzahl, Real)
- Suche nach 3 Indexfeldern – je 4 Zeichen lang – mit Wildcard-Funktion
- Sortieren und Filtern (kumuliertes Selektieren) geschieht nach den Index-Feldern
- Ausdruck der Dateien als Etiketten, Listen und Schemabriefe (mit Felder- und Tastatureinschüben an beliebigen Stellen des Formbriefes)
- normal kopierbare Programmdiskette, unterteilt in Hauptprogramme und diverse Hilfsprogramme
- einsatzfähig auf Apple IIe oder IIc. (Achtung: Brief-Modul läuft nicht mit Videx-Karte!)
- 256K RAM-Disks verwendbar

**Gesamtpreis 290,– (2 Disketten + gedrucktes Manual)**
**U. Stiehl**

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

**Applesoft hat nur eine 9stellige Mantisse. Wie man trotzdem eine höhere Präzision bei den Grundrechenarten in Basic erreichen kann, zeigt der nachfolgende Aufsatz, der speziell für den Informatikunterricht in der Schule gedacht ist.**

# Höhere Präzision bei den vier Grundrechenarten

**von Konrad Steinmeier**

Applesoft speichert reelle Zahlen mittels eines einigermaßen komplizierten Verfahrens, auf das in diesem Aufsatz nicht eingegangen werden soll. Nach erfolgter Umformung wird die reelle Zahl schließlich durch eine Folge von 5 natürlichen Zahlen aus dem Bereich von 0 bis 255 dargestellt. Diese werden im Anschluß an den Variablennamen, der zwei Speicherplätze in Anspruch nimmt, abgelegt, so daß eine reelle Variable insgesamt 7 Bytes benötigt. Die Tatsache, daß zusätzlich Integer-Variablen eingeführt wurden, ist unverständlich. Einerseits führt deren Verwendung nicht, wie man zunächst annehmen mag, zur Einsparung von Speicherplätzen. Sie lassen sich zwar durch zwei Bytes darstellen, belegen aber nichtsdestoweniger genausoviel Platz wie reelle Variablen. Die zur Darstellung nicht benötigten Bytes werden mit Nullen gefüllt und sind somit verschwendet. (Das Gesagte gilt nur für die einfachen Integer-Variablen. Dimensionierte Integer-Variablen = Integer-Arrays benötigen nur je 2 Bytes pro Zahl. Der Vorteil von Integer-Variablen kommt nur bei Compilern zum Tragen. Anm. d. Red.) Ferner tragen sie auch nicht zur Beschleunigung des Programmablaufs bei. Da sie vor der Verarbeitung jedesmal in Fließkommazahlen umgewandelt werden müssen, wird der Programmablauf sogar verlangsamt. Man kann das leicht an einfachen Beispielen nachprüfen:

Beispiel 1

```
10 I = 1: FOR X = 1 TO 12634:
J = I * I: NEXT
```

Beipiel 2

```
10 I% = 1: FOR X = 1 TO 12634:
J% = I% * I%: NEXT
```

(1) wird in 50s abgearbeitet, (2) dagegen braucht 65s, also eine um 30% längere Zeit!

Die bisherigen Ausführungen machen verständlich, daß Applesoft – wie nicht anders zu erwarten – mit beschränkter Genauigkeit arbeitet. Nun könnte aber durchaus gelegentlich der Fall eintreten, daß das Ergebnis einer Berechnung mit größerer Genauigkeit gewünscht wird. Dazu dient das nachfolgende Programm **MULTIPRECISION.** Es behandelt die vier Grundrechenarten, wobei es den folgenden Beschränkungen unterworfen ist. Um dem 40-Spalten-Format des Bildschirms gerecht zu werden, werden nur Zahlen verarbeitet, die aus maximal 18 Zeichen beste-


hen. Eingaben, die länger sind, werden automatisch auf 18 Zeichen gekürzt. Addition, Subtraktion und Multiplikation lassen Dezimalbrüche zu. Negative Zahlen werden nicht akzeptiert. Allerdings darf das Ergebnis einer Subtraktion auch negativ sein. Die Division wird nur in der Menge der natürlichen Zahlen durchgeführt, wobei auch der Divisionsrest ausgegeben wird. Diese Einschränkungen wurden vorgenommen, um mehrere Methoden einbringen zu können und um das Programm nicht unnötig zu komplizieren. Sie sind also nicht grundsätzlicher Natur. Immerhin werden aber in allen Fällen die Ergebnisse in vollständiger Genauigkeit, d. h. mit maximal 36 Ziffern, ausgegeben.
Es folgt eine kurze Beschreibung der verwendeten – eher primitiven – Techniken.

## 1. Eingabe der Operanden
(ab Zeile 7001)

Die Operanden werden als Strings eingegeben und anschließend ziffernweise in den Speichern der Arrays S1(I) und S2(I) abgelegt. Der Vorgang ist relativ umständlich, da die Operanden für die verschiedenen Operationen in unterschiedlichen Positionen plaziert werden müssen. Der Grund hierfür wird später ersichtlich. Zudem müssen Vor- bzw. Nachkommaziffern getrennt abgespeichert werden, da sich anderenfalls, insbesondere beim Multiplikationsalgorithmus, gewisse Schwierigkeiten ergeben würden (7080-7130). Zudem wurde dafür Sorge getragen, daß auch dann korrekte Ergebnisse geliefert werden, wenn die Variable DYM (siehe **Tabelle 4**) mit einer größeren Zahl belegt wird, so daß Eingaben mit mehr als 18 Zeichen möglich sind. Größenvergleiche (7150-7190) konnten sich daher nicht der VAL-Funktion bedienen. Das Programm wird dadurch etwas umfangreicher (und langsamer). Andererseits ist aber so die Zeichenzahl der Operanden nur den für Strings geltenden Beschränkungen unterworfen.

## 2. Sicherung gegen Eingabefehler
(ab Zeile 6001)

Zunächst werden führende Nullen beseitigt (6020). Zahlen, die kleiner als 1 sind und in amerikanischer Art (ohne Vorkommanull) eingegeben wurden, werden dann in deutsche Darstellungsweise umgewandelt (6030). Dann werden Eingaben, die zu lang sind, auf 18 Zeichen gekürzt (ohne

Rundung; 6110). Strings, die nicht-zulässige Zeichen enthalten, werden zurückgewiesen (6060-6100). Hierbei machen Komma und Doppelpunkt allerdings eine Ausnahme, da sie stets zur „?EXTRA IGNORED"-Meldung führen. Da dies keine übliche Fehlermeldung ist, läßt sie sich in Applesoft nicht mit ONERR GOTO abfangen. Man kann aber statt dessen den Bildschirmspeicher abfragen, der in diesem Fall den ersten Buchstaben der Meldung enthält (ASCII-Code 197), und so eine Fehlermeldung erzielen (6010) Eine andere Möglichkeit bestünde darin, GET anstelle von INPUT zu verwenden. Dann müßte aber entweder auf Korrekturmöglichkeit verzichtet oder eine aufwendige Programmierung der Pfeiltasten eingebaut werden. Es wird dann die Ganzzahligkeit überprüft (6120-6140) und schließlich die Position des Dezimalpunkts ermittelt (6150-6160). Dieser Teil ist recht lang geraten, worunter natürlich das Arbeitstempo leidet. Andererseits sind aber somit Eingabefehler, die zum Abbruch oder unerwünschten Ergebnissen führen könnten, ausgeschaltet.

## 3. Addition und Subtraktion

(ab Zeile 2001 bzw. 3001)

Bei der Addition sind die Operanden so abgespeichert, daß Ziffern gleichen Stellenwertes „übereinander stehen". Es wird dann ebenso verfahren, als rechnete man mit Bleistift und Papier. **Tabelle 1** mag dies verdeutlichen.
Dem gleichen Prinzip folgt der Subtraktionsalgorithmus. Ist der Minuend kleiner als der Subtrahend, so werden beide zunächst ausgetauscht und das Vorzeichen V$ geändert (3010-3030).

## 4. Multiplikation

(ab Zeile 4001)

Der Multiplikationsalgorithmus verwendet das Verfahren der sogenannten „indischen Multiplikation". Es wird dabei im Grunde die Regel verwendet, die Sie in der Schule für das Ausmultiplizieren zweier Summenterme gelernt haben („Jedes Glied der ersten Klammer wird mit jedem Glied der zweiten multipliziert..."). **Tabelle 2** macht das Verfahren deutlich.
Die Zwischenprodukte werden wie in **Tabelle 3** gebildet. Da nun hierbei unter Umständen eine Vielzahl von Produkten zu bilden ist, beansprucht der Prozeß in Applesoft oft geraume Zeit. Die Operanden werden hier „ganz oben" in den ent-

### MULTIPRECISION

```
1Ø1Ø L$ = " ":B$ = CHR$ (7):DYM = 35: DIM S(DYM),S1(DYM),S2(DYM): TEXT : HOME
1Ø2Ø INVERSE : PRINT SPC( 43)"<< RECHNEN MIT HOHER PRAEZISION >>" SPC( 43):
     POKE 34,4
1Ø3Ø VTAB 8: HTAB 1Ø: PRINT SPC( 22): PRINT : FOR I = 1 TO 11: HTAB 1Ø: PRINT
     L$;: HTAB 31: PRINT L$: NEXT : HTAB 1Ø: PRINT SPC( 22): NORMAL
1Ø4Ø VTAB 1Ø: HTAB 12: PRINT "A.........ADDITION": PRINT : HTAB 12: PRINT
     "S.....SUBTRAKTION"
1Ø5Ø PRINT : HTAB 12: PRINT "M...MULTIPLIKATION": PRINT : HTAB 12: PRINT
     "D.........DIVISION": PRINT : HTAB 12: PRINT "E.....PROGRAMMENDE"
1Ø6Ø VTAB 23: HTAB 16: PRINT "WAEHLE: < >" CHR$ (8) CHR$ (8);: GET A$
1Ø7Ø IF A$ = "S" THEN O$ = "-":O1$ = "MINUEND    ":O2$ = "SUBTRAHEND": GOTO 3Ø1Ø
1Ø8Ø IF A$ = "M" THEN F = 3:O$ = " * ":O1$ = "MULTIPLIKAND ":O2$ =
     "MULTIPLIKATOR": GOTO 4Ø1Ø
1Ø9Ø IF A$ = "D" THEN F = 4:O$ = " : ":O1$ = "DIVIDEND":O2$ = "DIVISOR ": GOTO
     5Ø1Ø
11ØØ IF A$ = "A" THEN O$ = "+":O1$ = "1. SUMMAND":O2$ = "2. SUMMAND": GOTO 2Ø1Ø
111Ø IF A$ < > "E" THEN PRINT B$: GOTO 1Ø6Ø: REM Falsche Taste!
112Ø TEXT : HOME : END
2ØØ1 REM Addition
2Ø1Ø GOSUB 7Ø1Ø
2Ø2Ø FOR I = DYM TO DYM - M STEP - 1
2Ø3Ø IF S1(I) + S2(I) + U > = 1Ø THEN S(I) = S1(I) + S2(I) + U - 1Ø:U = 1: GOTO
     2Ø5Ø
2Ø4Ø S(I) = S1(I) + S2(I) + U:U = Ø
2Ø5Ø NEXT : PRINT "= ";: IF S(DYM - M) = Ø THEN K = 1: PRINT L$;: REM Kein
     Übertrag
2Ø6Ø FOR I = K + DYM - M TO DYM: IF I = DYM - NK + 1 THEN PRINT ".";
2Ø7Ø PRINT S(I);: NEXT
2Ø8Ø HTAB 11: VTAB 22: PRINT "TASTE DRUECKEN! ";: GET A$: RUN
3ØØ1 REM Subtraktion
3Ø1Ø GOSUB 7Ø1Ø:V$ = L$: ON G = Ø GOTO 3Ø4Ø:V$ = "-"
3Ø2Ø FOR I = DYM - M TO DYM
3Ø3Ø T = S1(I):S1(I) = S2(I):S2(I) = T: NEXT
3Ø4Ø FOR I = DYM TO DYM - M STEP - 1
3Ø5Ø IF S1(I) > = S2(I) + U THEN S(I) = S1(I) - S2(I) - U:U = Ø: GOTO 3Ø7Ø
3Ø6Ø S(I) = S1(I) - S2(I) - U + 1Ø:U = 1
3Ø7Ø NEXT : PRINT "=";:I = DYM - M
3Ø8Ø IF S(I) = Ø AND I < DYM - NK THEN PRINT L$;:K = K + 1:I = I + 1: IF I <
     DYM - NK + 1 - NOT E THEN 3Ø8Ø: REM Führende Nullen beseitigen
3Ø9Ø PRINT V$;: GOTO 2Ø6Ø
4ØØ1 REM Multiplikation
4Ø1Ø GOSUB 7Ø1Ø:M = L1 + L2 - 3:T = Ø
4Ø2Ø FOR I = Ø TO M - 1:U = Ø
4Ø3Ø FOR J = DYM TO DYM - I STEP - 1
4Ø4Ø T = T + S1(J) * S2(DYM + DYM - J - I): NEXT : REM Indische Produkte
4Ø5Ø U = INT (T / 1Ø)
4Ø6Ø P = P + 1:S(P) = T - 1Ø * U:T = U: NEXT
4Ø7Ø IF T > Ø THEN PRINT T;
4Ø8Ø T = M
4Ø9Ø IF S(T) = Ø AND T > M - P1 - P2 + 4 THEN T = T - 1: GOTO 4Ø9Ø: REM
     Führende Nullen
41ØØ FOR I = T TO 1 STEP - 1: IF I = M - P1 - P2 + 3 THEN PRINT ".";
411Ø PRINT S(I);: NEXT : GOTO 2Ø8Ø
5ØØ1 REM Division
5Ø1Ø GOSUB 7Ø1Ø: FOR K = 1 TO M
5Ø2Ø FOR I = L2 TO 1 STEP - 1
5Ø3Ø T = S1(I) - S2(I)
5Ø4Ø IF T < Ø THEN T = T + 1Ø:S1(I - 1) = S1(I - 1) - 1
5Ø5Ø S1(I) = T: NEXT
5Ø6Ø IF S1(Ø) > = Ø THEN J = J + 1: GOTO 5Ø2Ø
5Ø7Ø S(K) = J:J = Ø: REM J+1 Subtraktionen
5Ø8Ø FOR I = L2 TO Ø STEP - 1
5Ø9Ø T = S1(I) + S2(I)
51ØØ IF T > 9 THEN T = T - 1Ø:S1(I - 1) = S1(I - 1) + 1
511Ø S1(I) = T: NEXT : REM Addition
512Ø FOR I = Ø TO L1 - 1:S1(I) = S1(I + 1): NEXT
513Ø S1(L1) = Ø: NEXT : REM Verschiebung nach links
514Ø P = P + 1: IF S(P) = Ø THEN 514Ø: REM Führende Nullen
515Ø FOR I = P TO M: PRINT S(I);: NEXT
516Ø PRINT : HTAB 3: FOR I = P TO M: PRINT "-";: NEXT
517Ø VTAB 16: HTAB 3: PRINT "REST: ";:T = - 1
518Ø T = T + 1: IF S1(T) = Ø AND T < L2 THEN 518Ø
519Ø FOR I = T TO L2 - 1: PRINT S1(I);: NEXT : GOTO 2Ø8Ø
6ØØ1 REM Kontrolle der Eingabe
6ØØ2 REM Beschränkung auf 18
6ØØ3 REM Zeichen
6Ø1Ø INPUT "";S$: ON S$ = "" OR PEEK (1321) = 197 OR PEEK (1577) = 197 GOTO
     6Ø7Ø: REM EXTRA IGNORED
6Ø2Ø IF LEFT$ (S$,1) = "Ø" AND LEN (S$) > 1 THEN S$ = RIGHT$ (S$, LEN (S$) -
     1): GOTO 6Ø2Ø
6Ø3Ø ON F = 4 AND S$ = "Ø" GOTO 6Ø7Ø: IF LEFT$ (S$,1) = "." THEN S$ = "Ø" + S$
6Ø4Ø FOR I = 1 TO LEN (S$):Z$ = MID$ (S$,I,1)
6Ø5Ø IF Z$ = "." AND F = 4 THEN 6Ø7Ø: REM Nur nat. Zahlen
6Ø6Ø IF Z$ = "." OR Z$ > "/" AND Z$ < ":" THEN 61ØØ: REM Kein unzulässiges
     Zeichen
```

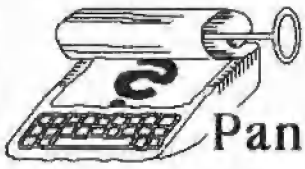
# pandabooks

Bismarckstr. 67, D-1000 Berlin 12 (030) 342 88 00

Ablage Bearbeiten Zeichensatz Größe Stil Hilfsmittel

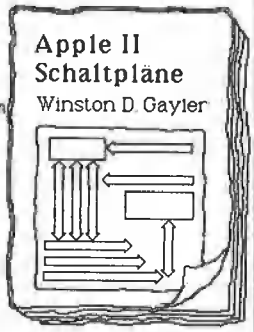
Pandabooks

**Gratis!**
**AppleQuick**
Das handliche Nachschlagewerk für häufig gebrauchte Adressen und Befehle!
64 Seiten über Applesoft, DOS, ProDOS, Pascal, CP/M.
**Gleich anfordern!**

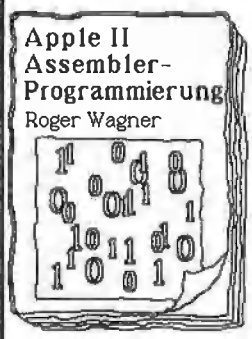


## Pandabooks!

Eine detaillierte Beschreibung der Apple II-Schaltungen.
Wenn Sie Ihren Apple selbst reparieren, Interface-Karten oder Schaltungserweiterungen entwerfen oder einfach nur besser über das Innenleben Ihres Apples Bescheid wissen wollen - dieses Buch bietet Ihnen eine Fülle an Informationen: Schaltpläne und Zeitdiagramme, Theorie und praktische Tips.




ISBN: 3-89058-012-2
215 Seiten, DIN A4 DM 64,--




Das Assembler-Lehrbuch für BASIC-Kenner.
Roger Wagner, der Autor vieler bekannter Software-Pakete, schrieb eine monatliche Kolumne über Assembler-Programmierung in der Apple-Zeitschrift SOFTALK.
Der vorliegende BAnd faßt diese Reihe, korrigiert und erweitert, zusammen: Eine stufenweise Einführung in die Befehle und Strukturen der 6502 Assemblersprache, mit vielen Beispielen von der einfachen Tongenerierung bis zum Diskettenzugriff.

3-89058-003-3 277 Seiten DM 48,--
komplett mit Disk DM 89,--

Ein Simulationsprogramm, das Sie in das Innere des 6502-Mikroprozessors führt. Sie sehen auf dem Bildschirm, wie die einzelnen Instruktionen in Zeitlupe ausgeführt werden, wie sich die Register und die Flags verändern. Ein unverzichtbares Hilfsmittel beim Erlernen der Assembler-Programmierung - danach ein wertvolles Werkzeug beim Testen Ihrer eigenen Programme.


Komplett mit einem 6502-Editor/Assembler, deutschem Handbuch (150 Seiten) und über 30 Beispielprogrammen.

3-89058-019-X DM 129,--

In allen Buchhandlungen und Computershops oder direkt von: Pandabooks, Bismarckstr. 67, 1000 Berlin 12, (030) 342 88 00

**Bestellcoupon**

**Name:**
**Anschrift:**

○ V-Scheck liegt bei (spesenfreie Lieferung)
○ per Nachnahme (zzgl. Versandspesen)

| Menge | Titel | Preis |
|---|---|---|
| 1 | AppleQuick | gratis |
| | | |
| | | |
| | | |

Österreich: OÖ Landesverlag, Linz • Schweiz: Thali AG, Hitzkirch

---

IBM®-Gehäuse ..... 229,-

## ZUSATZ-KARTEN:

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

## Floppy-Controler

**FDC 4** für alle Laufwerke ..... 230,- Bausatz wie links ..... 199,-
Leerplatine wie oben incl. Prom u. Eprom ..... 130,-

## Autopatch-Controler (Erphi Controler)

1 x 35 bis 2 x 80 Tr.-Disk, keine Patch-Disk notwendig, Patch DOS 3.3, Diversi-DOS 2-C, 4-C (DD MOVER), Pascal 1.1, Pascal 1.2, CPM 2.2, ProDOS 1.0.1, 1.1, 1.1.1; Sie können die Laufwerke untereinander mischen .. **298,-**

**Joy Stick** ..... 69,- **Netzteil 5A** ..... 149,-

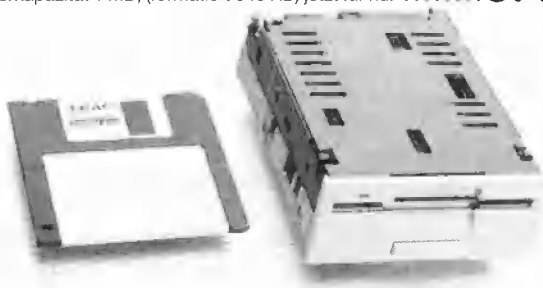
**Halbschalen-Gehäuse** für 2 Slimline-Laufwerke ..... 65,-
**Floppy-Kabel** 34pol. für 2 Laufwerke mit Shugart-Bus ..... 42,-

## 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,-**

## TEAC 3½" Laufwerk FD 35 F 579,-

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



## Nochmalige Preissenkung bei TEAC:

**TEAC FD 55 A** 1 x 40 Track ..... 449,- **TEAC FD 55 B** 2 x 40 Track ..... 535,-
**TEAC FD 55 E** 1 x 80 Track ..... 535,- **TEAC FD 55 F** 2 x 80 Track ..... 638,-

**Philipps Slimline Floppy X3134 A,** 2x80 Track
solange Vorrat ..... nur **638,-**

## EPSON DRUCKER

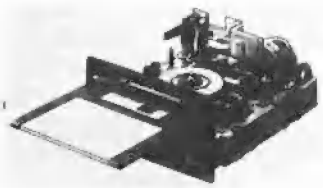
**EPSON FX 80** ..... 1670,- **EPSON FX 100** ..... 2159,-
**EPSON RX 80** ..... 1079,- **EPSON RX 80 FT** ..... 1295,-

**Patch-Diskette** für **SONY 3½" Laufwerke** – ermöglicht die Anpassung an II/IIe und kompatible Computer ..... 80,- Manual vorab 15,-DM (wird beim Kauf der Patch-Diskette angerechnet).
**Disketten** Döb&Böd SS/DD 10St. 62,- **dto.** für 5¼" Laufwerke ..... 80,-
**10 Disketten** f.Sony 3½" Laufw. .... 150,-

## SONY 3½" Laufwerk ..... nur 849,-

**Akustikkoppler AK 300** mit FTZ-Nr. incl. Netzteil ..... 549,-

**Die Microfloppy mit Zukunft:** (o.Abb.)
Speicherkapazität: 2 x 1 MByte formatiert: 2 x 640 kByte. Anschlußfertig mit PROM-residenter Patchsoftware für CP/M 2.2, Apple DOS 3.3, Diversi-DOS 2-C, 4-C (DD MOVER), Apple Pascal 1.1, Pascal 1.2, CPM 2.2, Pro-DOS 1.0.1, 1.1, 1.1.1 zum Preis von ..... **1590,-**
**Low Power Version** ..... 1690,-



## 10 MB Winchester 4490,-

incl. Controler, Software und Gehäuse .....

**Gesamt-Preisliste anfordern!** **Preise incl. ges. Mwst.**

**Ueding electronics**
Holtewiese 2
5750 Menden 1
DFÜ 02373/66877
Tel. 02373/63159

sprechenden Arrays abgelegt, da sich anderenfalls die Positionsbestimmung für den Dezimalpunkt als recht umständlich erweist.

## 5. Division

(ab Zeile 5001)

Hier werden, wie schon erwähnt, nur natürliche Zahlen verarbeitet, wobei der Divisor nicht größer als der Dividend sein darf. Das Verfahren ist primitiv und beansprucht daher recht viel Zeit. Die Operanden stehen „ganz unten" in S1(I) bzw S2(I). Nun wird der Divisor so oft subtrahiert, bis sich ein negatives „Ergebnis" einstellt (5020-5060). Danach wird er einmal addiert (5080-5110). Die um 1 verminderte Anzahl der Subtraktionen wird als Ziffer des Quotienten notiert (5070). Nun wird der Dividend um einen Platz nach links verschoben (5120-5130) und der Prozeß so lange wiederholt, bis der Dividend „abgearbeitet" ist. Im Dividend-Array S1(I) bleibt dann der Divisionsrest übrig und kann ausgedruckt werden (5190).

Die zulässige Länge der Operanden kann durch entsprechende Änderung der Variablen DYM in Zeile 1010 geändert werden. Da INPUT-Eingaben auf 239 Zeichen beschränkt sind, könnte DYM Werte bis zu 477 annehmen. Es wäre dann die Verarbeitung von Zahlen mit maximal 239 Ziffern möglich. Bei der Multiplikation zweier 239stelliger Zahlen dauert allerdings die Eingabe schon etwa eine Minute und die Berechnung des Produkts etwa 30 Minuten. Ändert man die dynamischen Dimensionierungen in Zeile 1010 in DIM S(477), S1(477), S2(477) und compiliert das Programm (z.B. mit dem Hayden-Compiler), dann wird diese Zeit auf unter 8 Minuten heruntergedrückt. Für diesen Fall würde es sich empfehlen, falls vorhanden, die 80-Zeichen-Karte zu aktivieren und einige VTABs so zu ändern, daß sowohl Operanden als auch Ergebnis gleichzeitig auf dem Bildschirm zu sehen sind.

Zum Zeitbedarf: Multiplikation und Division können, abhängig natürlich von der Zeichenzahl der Operanden, bis zu 11 Sekunden in Anspruch nehmen. Vergrößert man DYM, so wächst die Rechenzeit u. U. erheblich. Sie ist in etwa proportional zum Produkt der Zeichenzahlen (= Anzahl der Stellen) der beiden Operanden. Eine Verdoppelung der Zeichenzahl führt also zu rund vierfachem Zeitbedarf. So dauert z. B. die Multiplikation zweier 40stelliger Zahlen etwa 50 Sekunden.

```
6Ø7Ø HTAB 11: VTAB 22: FLASH : PRINT "FALSCHE EINGABE!"B$B$B$
6Ø8Ø FOR O = 1 TO 8ØØ: NEXT : NORMAL
6Ø9Ø POP : ON Z GOTO 7Ø3Ø: GOTO 7Ø1Ø
61ØØ NEXT
611Ø IF LEN (S$) > (DYM + 1) / 2 THEN S$ = LEFT$ (S$,(DYM + 1) / 2): REM 18
     Zeichen
612Ø J = Ø: FOR I = 1 TO LEN (S$): IF MID$ (S$,I,1) = "." THEN J = J + 1
613Ø NEXT : IF J > 1 THEN 6Ø7Ø: REM Höchstens ein Punkt
614Ø ON J GOTO 615Ø:S$ = S$ + "."
615Ø I = 1
616Ø IF MID$ (S$,I,1) > "/" THEN I = I + 1: GOTO 616Ø: REM Position
     Dezimalpunkt
617Ø RETURN
7ØØ1 REM Eingabe der Operanden
7Ø1Ø HOME : VTAB 1Ø: PRINT O1$": ";: GOSUB 6Ø1Ø
7Ø2Ø S1$ = S$:P1 = I:L1 = LEN (S$): IF RIGHT$ (S$,1) = "." THEN S1$ = LEFT$
     (S1$,L1 - 1):F1 = 1
7Ø3Ø G = Ø:Z = 1: VTAB 12: CALL - 958: PRINT O2$": ";: GOSUB 6Ø1Ø
7Ø4Ø S2$ = S$:P2 = I:L2 = LEN (S$): IF RIGHT$ (S$,1) = "." THEN S2$ = LEFT$
     (S2$,L2 - 1):F2 = 1
7Ø5Ø VK = P1: IF P2 > P1 THEN VK = P2
7Ø6Ø NK = L1 - P1: IF NK < L2 - P2 THEN NK = L2 - P2
7Ø7Ø ON F = 4 GOTO 712Ø:M = VK + NK - 1
7Ø8Ø FOR I = P1 - 1 TO 1 STEP - 1:S1(DYM - NK * (F < 3) - (L1 - P1) * (F = 3) -
     P1 + I + 1) = VAL ( MID$ (S1$,I,1)): NEXT : IF F1 THEN 71ØØ
7Ø9Ø FOR I = P1 + 1 TO L1:S1(DYM - NK * (F < 3) - (L1 - P1) * (F = 3) - P1 + I)
     = VAL ( MID$ (S1$,I,1)): NEXT
71ØØ FOR I = P2 - 1 TO 1 STEP - 1:S2(DYM - NK * (F < 3) - (L2 - P2) * (F = 3) -
     P2 + I + 1) = VAL ( MID$ (S2$,I,1)): NEXT : IF F2 THEN 714Ø
711Ø FOR I = P2 + 1 TO L2:S2(DYM - NK * (F < 3) - (L2 - P2) * (F = 3) - P2 + I)
     = VAL ( MID$ (S2$,I,1)): NEXT : GOTO 714Ø
712Ø L1 = L1 - 1:L2 = L2 - 1:M = L1 - L2 + 1: FOR I = 1 TO L1:S1(I) = VAL (
     MID$ (S1$,I,1)): NEXT
713Ø FOR I = 1 TO L2:S2(I) = VAL ( MID$ (S2$,I,1)): NEXT
714Ø IF L1 < > L2 OR P1 < > P2 THEN 716Ø: REM Keine Gleichheit
715Ø FOR I = DYM - L1 TO DYM:E = E + ABS (S1(I) - S2(I)): NEXT
716Ø I = 1:T = L2: IF F < 4 THEN I = DYM - VK - NK:T = DYM
717Ø G = (P1 < P2): ON P1 < > P2 GOTO 72ØØ: REM S2$ > S1$
718Ø IF S1(I) = S2(I) AND I < T THEN I = I + 1: GOTO 718Ø
719Ø G = (S1(I) < S2(I)): REM S2$ > S1$
72ØØ IF F = 4 AND G THEN L1 = L1 + 1: GOSUB 6Ø7Ø: REM Divisor > Dividend
721Ø HOME : VTAB 1Ø: ON F > 2 GOTO 724Ø: HTAB VK - P1 + 4: PRINT S1$
722Ø PRINT O$;: HTAB VK - P2 + 4: PRINT S2$
723Ø PRINT "  ";: FOR I = 1 TO VK + NK + 1 - (F1 AND F2): PRINT "-";: NEXT :
     PRINT : RETURN
724Ø PRINT S1$O$S2$: PRINT : PRINT "= ";: RETURN
```

### Tabelle 1

| | | |
|---|---|---|
| 1. Summand | (S1(I)) | 1 2 . Ø 3 |
| 2. Summand | (S2(I)) | 8 . 1 9 5 |
| Summe | (S1(I) + S2(I)) | 1 1Ø . 1 12 5 |
| Übertrag | (U) | 1 . 1 |
| Ergebnis | (S(I)) | 2 Ø . 2 2 5 |

### Tabelle 2

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Multiplikand | (S1(I)) | | | | 1 | 2 | 3 | 4 | 5 |
| Multiplikator | (S2(I)) | | | | | 6 | 7 | 8 | 9 |
| Übertrag | (U) | | Ø | 2 | 4 | 8 | 11 | 1Ø | 8 | 4 |
| Zwischenprodukt + Übertrag | | Ø | 8 | 23 | 48 | 81 | 11Ø | 1Ø2 | 8Ø | 45 |
| Ergebnis | | Ø | 8 | 3 | 8 | 1 | Ø | 2 | Ø | 5 |

### Tabelle 3

```
 45 = 5*9                                  81 = 5*Ø + 4*6 * 3*7 + 2*8 + 1*9 + 11
 8Ø = 5*8 + 4*9 + 4                        48 = 4*Ø + 3*6 + 2*7 + 1*8 + 8
1Ø2 = 5*7 + 4*8 + 3*9 + 8                  23 = 3*Ø + 2*6 + 1*7 + 4
11Ø = 5*6 + 4*7 + 3*8 + 2*9 + 1Ø            8 = 2*Ø + 1*6 + 2
```

### Tabelle 4

| | |
|---|---|
| DYM | Bestimmt Maximallänge der Operanden zu (DYM + 1)/2 |
| S1(I), S2(I), S(I) | Arrays für Operanden und Resultat |
| S1$, S2$ | Operanden-Strings |
| O1$, O2$ | Namen der Operanden |
| V$ | Vorzeichen des Resultats |
| L1, L2 | Länge der Operanden |
| P1, P2 | Positionszeiger für Dezimalpunkt |
| VK | Größere der Anzahlen der Vorkommastellen |
| NK | Größere der Anzahlen der Nachkommastellen |
| M | Speicherraum für einen Operanden |
| F1, F2 | Flags für Ganzzahligkeit |
| G, E | Größer/Gleich-Flags |
| F | Kenn-Nummer der Operation |
| U | Übertrag |
| T | Zwischenspeicher |
| Z | Gibt an, welcher Operand eingegeben wird |

# Wordstar-Transfer-Refiner

**Konvertierung von CP/M-Wordstar- in DOS-Applewriter-Textfiles**

**von Dr. Jürgen B. Kehrel**

**Zum Lieferumfang des CP/M-Systems gehört ein Programm „APDOS", mit dem Sie DOS-Files nach CP/M übertragen können. Genau den umgekehrten Weg vollzieht der Wordstar-Transfer-Refiner (WS.TRANSFER). Er wurde speziell dafür ausgelegt, Wordstar-Dateien in DOS-Textfiles umzuwandeln, die Sie dann z.B. mit dem Applewriter II lesen können und die auch die Setzmaschine des „Peeker" direkt verarbeiten kann.**

WS.TRANSFER benötigt zwei Laufwerke, DOS 3.3 sowie 48K Hauptspeicher. Er kann CP/M-Textfiles *beliebiger* Länge in 16-Sektor-CP/M nach DOS 3.3 übertragen, wobei einige spezielle Anpassungen für Wordstar-Dateien vorgenommen werden (siehe Ende des Listings). Wenn Sie diese nicht benötigen, können Sie die Konvertierungen streichen und nur den Pufferinhalt lesen und über COUT ausgeben (neue Assemblierung erforderlich!), die übrige Funktion des Programms wird dadurch nicht berührt. Die abgedruckte Version bereitet die Texte der „Peeker"-Redaktion für die Setzmaschine vor. Aus diesem Grunde werden mehrfache Leerzeichen entfernt, wodurch natürlich Tabellen ihre Form verlieren. Sie können diese Eigenschaft vorübergehend abstellen, indem Sie nach $0D2D ein $00 schreiben und somit CPX $00 erzeugen:

```
BLOAD WS.TRANSFER
POKE 3373,0
CALL 2051
```

**Die Bedienung:**

Das Programm arbeitet menügesteuert und ist gegen viele Fehleingaben abgesichert. Nach dem Start mit **BRUN WS.TRANSFER** erscheint der Titel, den ein Tastendruck beseitigt. Legen Sie jetzt die richtigen Disketten ein. Nach einem Return wird das Inhaltsverzeichnis der CP/M-Diskette gelesen und auf den Bildschirm geschrieben. Nach maximal 20 Dateien stoppt die Ausgabe, und erst durch einen erneuten Datendruck sehen Sie die nächsten Dateien. Betätigen Sie allerdings die Escape-Taste, wird die Auflistung sofort beendet. Jetzt geben Sie den Dateinamen ein. Alternativ können Sie mit den Cursortasten nach oben gehen und dann mit dem Rechtspfeil den Namen überfahren. Achtung: Leerzeichen über das Namensende hinaus zählen mit!
Als letztes geben Sie noch den Namen ein, den der DOS-Textfile erhalten soll. Drücken Sie bei einer Namensabfrage nur Return, so gelangen Sie in das Hauptmenü zurück. Sie werden noch einmal um Ihre Bestätigung gebeten, danach besorgt das Programm den Rest. Bei sehr langen Dateien sind mehrere Durchgänge erforderlich. Auf dem Bildschirm können Sie die gelesenen CP/M-Blöcke mitverfolgen. Für jeden gelesenen Sektor wird zusätzlich ein Punkt ausgegeben.
Da WS.TRANSFER sehr lang ist, ist das Assembler-Listing mit vielen Kommentaren versehen. Auf eine weitergehende Beschreibung wurde deshalb verzichtet.

Ein Beitrag über das CP/M-Directory folgt im nächsten Peeker.

## Apple-Hotline

Manchmal habe ich das Gefühl, daß man mich mit der Apple-Hotline in München verwechselt. Am letzten Donnerstag habe ich einmal eine Strichliste angelegt: Es riefen mich tatsächlich über 100 Apple-Besitzer an, wobei einige Anrufer mir sogar sagten, sie seien an mich weiterverwiesen worden. Ich habe dann einmal den Spieß herumgedreht und mich als „unbedarfter Anwender" telefonisch an verschiedene Apple-Händler, Apple-Geschäftsstellen sowie auch an die Apple-Zentrale in München mit folgender Frage gewandt:

*Ich benutzte den Applewriter IIe auf einem Apple IIc. Ich sehe am Bildschirm immer „so komische Zeichen" und nie die Buchstaben, die ich getippt habe. Können Sie mir da helfen?*

Niemand von Hamburg bis München konnte mir helfen! Zur technischen Seite: Der Applewriter IIe ist auf dem Apple IIc zwar funktionell, aber nicht ästhetisch lauffähig, weil der Cursor durch Umwandlung des NORMAL-Zeichens in das entsprechende INVERSE-Zeichen erzeugt wird. Der Apple IIc hat jedoch einen anderen Zeichensatz mit „Maus"-Symbolen, die dann an Stelle des Cursors sichtbar wären.

Eine kurze Analyse zeigte mir, daß sich das Problem mit einem 25-30 Bytes umfassenden Patch beheben ließe, also eine Kulanz, die ein Weltkonzern schon aus Imagegründen seinen Kunden nicht ausschlagen sollte.
Haben Sie schon einmal die Hotline der Firma Apple in München schriftlich oder telefonisch (089/350340) in Anspruch genommen? Schreiben Sie mir, ob Ihre Probleme zufriedenstellend gelöst wurden. Wir werden Ihre Erfahrungen auswerten und darüber berichten.
us

## GETCPM

```
10Ø HOME : PRINT "CPM=>DOS TRANSFER von Thomas
    Fink, 1984"
20Ø REM * BUFFER & KONST. *
21Ø BØ = 16384: HIMEM: BØ
22Ø B2 = BØ + 2Ø48
23Ø D$ = CHR$ (4):G$ = CHR$ (7)
24Ø E = Ø:K = Ø:EØ = 13:E1 = 31
25Ø DIM N$(64),D(64)
26Ø DIM EX(64),P(64),BZ(64)
27Ø GOSUB 62ØØ
1ØØØ REM "* HAUPTPROGRAMM *"
1Ø1Ø PRINT : PRINT
1Ø2Ø PRINT "Source: CP/M     Drive 1"
1Ø3Ø PRINT "Target: DOS3.3   Drive 2"
1Ø4Ø PRINT
1Ø5Ø PRINT "Bitte einlegen und SPACE drücken!"
1Ø6Ø GET E$: PRINT E$
1Ø7Ø IF E$ < > " " THEN END
2ØØØ REM * READ & PRINT DIR *
2Ø1Ø BUF = BØ:BLK = Ø: GOSUB 5ØØØ
2Ø2Ø BUF = BØ + 1Ø24:BLK = 1: GOSUB 5ØØØ
2Ø3Ø NR = Ø
2Ø4Ø FOR D = Ø TO 63
2Ø5Ø J = BØ + 32 * D: IF PEEK (J) > Ø GOTO 217Ø
2Ø6Ø N$(D) = "":P(D) = J + 16:BZ(D) = PEEK (J + 15)
2Ø7Ø FOR K = J + 1 TO J + 8
2Ø8Ø E = PEEK (K): IF E > 32 THEN N$(D) = N$(D) + CHR$ (E)
2Ø9Ø NEXT
21ØØ N$(D) = N$(D) + ".": IF N$(D) = "." GOTO 217Ø
211Ø FOR K = J + 9 TO J + 11
212Ø E = PEEK (K): IF E > 32 THEN N$(D) = N$(D) + CHR$ (E)
213Ø NEXT
214Ø EX(D) = PEEK (J + 12): IF EX(D) > Ø GOTO 217Ø
215Ø NR = NR + 1:D(NR) = D
216Ø PRINT LEFT$ ( RIGHT$ (" " + STR$ (NR),2) + ") " +
    N$(D) + "                ",2Ø);
217Ø NEXT D
218Ø PRINT
219Ø IF NR = Ø THEN PRINT "Keine Files!": GOTO 1ØØØ
3ØØØ REM * KOPIEREN *
3Ø1Ø INPUT "CPM-File-Nr? ";E$:E = VAL (E$): IF E = Ø GOTO
    1ØØØ
3Ø2Ø D = D(E):N$ = N$(D):EX = Ø
3Ø3Ø PRINT D$ "OPEN "N$",D2": PRINT D$ "DELETE "N$
3Ø4Ø PRINT D$ "OPEN "N$: PRINT D$ "WRITE "N$
3Ø5Ø GOSUB 4ØØØ
3Ø6Ø FOR K = B2 TO B2 + BZ(D) * 128:E = PEEK (K): IF E =
    EØ OR E > E1 THEN PRINT CHR$ (E);: NEXT : GOTO 3Ø9Ø
3Ø7Ø IF E = 26 GOTO 314Ø
3Ø8Ø NEXT K
3Ø9Ø IF BZ(D) < 128 GOTO 314Ø
31ØØ EX = EX + 1
311Ø FOR D = Ø TO 63
312Ø IF N$(D) = N$ AND EX(D) = EX GOTO 3Ø5Ø
313Ø NEXT
314Ø PRINT
315Ø PRINT D$ "CLOSE"
316Ø GOTO 1ØØØ
4ØØØ REM * LOAD EXTEND *
4Ø1Ø FOR K = Ø TO 15
4Ø2Ø BLK = PEEK (P(D) + K)
4Ø3Ø IF BLK = Ø GOTO 4Ø7Ø
4Ø4Ø BUF = B2 + K * 1Ø24
4Ø5Ø GOSUB 5ØØØ
4Ø6Ø NEXT
4Ø7Ø RETURN
5ØØØ REM * READ CPM-BLOCK *
5Ø1Ø CMD = 1:VOL = Ø:DRV = 1
5Ø2Ø TR = INT (BLK / 4)
5Ø3Ø S = (BLK - 4 * TR) * 4
5Ø4Ø TR = TR + 3:B = BUF
5Ø5Ø FOR I = Ø TO 3
5Ø6Ø SEC = S + I
5Ø7Ø BUF = B + 256 * I
5Ø8Ø GOSUB 6ØØØ
5Ø9Ø NEXT
51ØØ BUF = B
511Ø RETURN
6ØØØ REM * CALL RWTS *
6Ø1Ø IOB = PEEK (43713) + 256 * PEEK (43714)
6Ø2Ø POKE IOB + 2,DRV
6Ø3Ø POKE IOB + 3,VOL
6Ø4Ø POKE IOB + 4,TR
6Ø5Ø POKE IOB + 5,SI(SEC)
6Ø6Ø POKE IOB + 9,BUF / 256
6Ø7Ø POKE IOB + 8,BUF - PEEK (IOB + 9) * 256
6Ø8Ø POKE IOB + 12,CMD
6Ø9Ø CALL RWTS
61ØØ ERC = PEEK (IOB + 13) * ( PEEK (Ø) > 127)
611Ø IF ERC = Ø THEN RETURN
612Ø EC = INT ( LOG (ERC) / LOG (2) - 3.5)
613Ø PRINT G$;EM$(EC);G$;
614Ø STOP
62ØØ REM * RWTSJMP *
621Ø RWTS = 768
622Ø FOR I = RWTS TO RWTS + 12
623Ø READ E: POKE I,E
624Ø NEXT
625Ø DATA 32,227,3,32,217,3
626Ø DATA 8,1Ø4,1Ø6,1Ø6
627Ø DATA 133,Ø,96
628Ø REM * SOFT SECTORING *
629Ø DIM SI(15)
63ØØ FOR I = Ø TO 15: READ SI(I): NEXT
631Ø DATA Ø,6,12,3,9,15,14,5,11,2,8,7,13,4,1Ø,1
632Ø REM * ERRORS *
633Ø DIM EM$(2)
634Ø FOR I = Ø TO 2: READ EM$(I): NEXT
635Ø DATA "WRITE PROTECTED","VOLUME MISMATCH","DRIVE
    ERROR"
636Ø RETURN
```

## WS.TRANSFER

```
                    1   ********************************
                    2   *  Wordstar  Transfer-Refiner  *
                    3   *  von  Dr. Jürgen B.  Kehrel  *
                    4   *  Heidelberg,  Dezember 1984  *
                    5   ********************************
                    7   *
                    8   * Apple II mit 48 KByte RAM
                    9   *   und normalem DOS 3.3
                    1Ø  *
                    11          ORG  $8Ø3
                    12  *
                    13  PTR     EQU  $Ø          ;Zeiger
                    14  TRACK   EQU  $2          ;Spur
                    15  SECTOR  EQU  $3          ;Sektor
                    16  PUFFER  EQU  $4          ;Speicher
                    17  IND     EQU  $6          ;Zeiger
                    18  ALT     EQU  $8          ;Speicher
                    19  CH      EQU  $24         ;Cursor horz.
                    2Ø  PROMPT  EQU  $33         ;Prompt
                    21  PREG    EQU  $48         ;P-Register
                    22  CURLIN  EQU  $75         ;AS-Zeile
                    23  IOB     EQU  $CE         ;Speicher
                    24  RUNMODE EQU  $D9         ;Run-Modus
                    25  IN      EQU  $2ØØ        ;Tastaturpuf.
                    26  DOSKALT EQU  $3D3        ;Kaltstart
                    27  RWTS    EQU  $3D9
                    28  GETIOB  EQU  $3E3        ;Suche IOB
                    29  TEXT    EQU  $FB2F
                    3Ø  TABV    EQU  $FB5B       ;Position
                    31  CLREOP  EQU  $FC42       ;Löschen
                    32  HOME    EQU  $FC58
                    33  RDKEY   EQU  $FDØC       ;Taste lesen
                    34  RDKEY1  EQU  $FD18       ;Taste lesen
                    35  GETLN   EQU  $FD6A       ;Zeile lesen
                    36  PRBYTE  EQU  $FDDA       ;Byteausgabe
                    37  COUT    EQU  $FDED       ;ASCII-Ausgabe
                    38  COUT1   EQU  $FDFØ       ;Bildschirm
                    39  *
                    4Ø  *
                    41  * Titel und Menü ausgeben
                    42  *
Ø8Ø3: D8            43  START   CLD              ;Binärmode
Ø8Ø4: 2Ø 2F FB      44          JSR  TEXT
Ø8Ø7: 2Ø 58 FC      45          JSR  HOME
Ø8ØA: A9 Ø4         46          LDA  #4
Ø8ØC: 85 24         47          STA  CH
Ø8ØE: A9 ØA         48          LDA  #1Ø
Ø81Ø: 2Ø 5B FB      49          JSR  TABV
Ø813: 2Ø D9 ØC      5Ø          JSR  PRINT
Ø816: AA AØ D7      51          ASC  "* WORDSTAR  TRANSFER"
Ø82A: AD D2 C5      52          ASC  "-REFINER *"
Ø834: ØØ            53          HEX  ØØ
Ø835: A9 Ø6         54          LDA  #6
```

```
0837: 85 24     55            STA  CH
0839: A9 14     56            LDA  #20
083B: 20 5B FB  57            JSR  TABV
083E: 20 D9 0C  58            JSR  PRINT
0841: D6 CF CE  59            ASC  "VON  DR. JUERGEN"
0851: A0 C2 AE  60            ASC  " B. KEHREL"
085B: 8D 00     61            HEX  8D,00
085D: A9 10     62            LDA  #16
085F: 85 24     63            STA  CH
0861: 20 D9 0C  64            JSR  PRINT
0864: A8 B1 B9  65            ASC  "(1984)"
086A: 00        66            HEX  00
086B: 20 18 FD  67  MENUE     JSR  RDKEY1     ;Tastendruck
086E: 20 58 FC  68  MENUE1    JSR  HOME
0871: A9 0A     69            LDA  #10
0873: 20 5B FB  70            JSR  TABV
0876: 20 D9 0C  71            JSR  PRINT
0879: CC C5 C7  72            ASC  "LEGEN SIE ..."
0886: 8D 8D     73            HEX  8D8D
0888: C4 C9 C5  74            ASC  "DIE WORDSTAR DISKETTE"
089D: A0 C9 CE  75            ASC  " IN LAUFWERK 2"
08AB: 8D        76            HEX  8D
08AC: C5 C9 CE  77            ASC  "EINE DOS 3.3 DISKETTE"
08C1: A0 C9 CE  78            ASC  " IN LAUFWERK 1"
08CF: 8D 8D 8D  79            HEX  8D8D8D8D
08D3: BC D2 C5  80            ASC  "<RETURN> = START,"
08E4: A0 BC C5  81            ASC  " <ESC> = ENDE"
08F1: A0 00     82            HEX  A000
08F3: 20 0C FD  83  EINGABE   JSR  RDKEY
08F6: C9 8D     84            CMP  #$8D       ;Return
08F8: F0 0F     85            BEQ  DIR
08FA: C9 9B     86            CMP  #$9B       ;Escape
08FC: D0 F5     87            BNE  EINGABE
08FE: 4C D3 03  88            JMP  DOSKALT
                89  *
                90  *
                91  * Lese CP/M Directory (Block 0 und 1)
                92  *
0901: 05 0E 0F  93  TAB       HEX  05,0E,0F,09 ;Sektor-
0905: 03 0C 06  94            HEX  03,0C,06,00 ;Code
                95  *
0909: A9 03     96  DIR       LDA  #3          ;Track 3
090B: 85 02     97            STA  TRACK
090D: A9 00     98            LDA  #0          ;Puffer ab
090F: 85 04     99            STA  PUFFER      ; $1000
0911: A9 10    100            LDA  #$10
0913: 85 05    101            STA  PUFFER+1
0915: A0 07    102            LDY  #7          ;8 Sektoren
0917: B9 01 09 103  LOOP      LDA  TAB,Y
091A: 85 03    104            STA  SECTOR
091C: 84 08    105            STY  ALT
091E: 20 5C 0C 106            JSR  READ        ;einlesen
0921: A4 08    107            LDY  ALT
0923: E6 05    108            INC  PUFFER+1
0925: 88       109            DEY
0926: 10 EF    110            BPL  LOOP
               111  *
               112  *
               113  * Directory nach Eintragungen durchsuchen
               114  *
0928: 20 58 FC 115            JSR  HOME
092B: A9 14    116            LDA  #20         ;20 Zeilen
092D: 85 08    117            STA  ALT
092F: A9 10    118            LDA  #$10        ;Pufferanfang
0931: 8D 3B 09 119            STA  LADEN+2     ;bei $1000
0934: A9 00    120            LDA  #0
0936: 8D 3A 09 121  LOOP1     STA  LADEN+1
0939: AC FF FF 122  LADEN     LDY  $FFFF       ;Dummy-Wert
093C: C0 E5    123            CPY  #$E5        ;gelöscht?
093E: D0 11    124            BNE  AUSGABE
0940: 18       125            CLC
0941: 69 20    126  ADD       ADC  #$20        ;nächster
0943: 90 F1    127            BCC  LOOP1       ;Eintrag
0945: EE 3B 09 128            INC  LADEN+2
0948: AC 3B 09 129            LDY  LADEN+2
094B: C0 18    130            CPY  #$18        ;Pufferende
094D: 90 E7    131            BLT  LOOP1       ;bei $17FF
094F: B0 7A    132            BGE  FILE
               133  *
               134  *
               135  * Gültige Eintragungen ausgeben
               136  *
0951: AD 3B 09 137  AUSGABE   LDA  LADEN+2
0954: 85 05    138            STA  PUFFER+1
0956: AC 3A 09 139            LDY  LADEN+1
0959: C8       140            INY
095A: 84 04    141            STY  PUFFER
```

```
095C: A0 0B    142            LDY  #11         ;Extension
095E: B1 04    143            LDA  (PUFFER),Y  ;überprüfen
0960: D0 49    144            BNE  PULL        ;nur 1. Eintrag
               145  *
0962: A9 AD    146            LDA  #$AD        ; -
0964: 20 ED FD 147            JSR  COUT        ;ausgeben
0967: A2 07    148            LDX  #7          ;max. 8 Z
0969: A0 00    149            LDY  #0
096B: B1 04    150  OUT       LDA  (PUFFER),Y
096D: C9 21    151            CMP  #$21        ;keine Leer-
096F: 90 05    152            BLT  SPRINGE     ;zeichen
0971: 09 80    153            ORA  #$80        ;Bit 7 setzen
0973: 20 ED FD 154            JSR  COUT        ;ausgeben
0976: E6 04    155  SPRINGE   INC  PUFFER
0978: CA       156            DEX
0979: 10 F0    157            BPL  OUT
097B: A9 AE    158            LDA  #$AE        ;Punkt
097D: 20 ED FD 159            JSR  COUT        ;ausgeben
0980: A2 02    160            LDX  #2          ;max. 3 Z
0982: B1 04    161  OUT1      LDA  (PUFFER),Y
0984: C9 21    162            CMP  #$21        ;keine Leer-
0986: 90 05    163            BLT  WEITER      ;zeichen
0988: 09 80    164            ORA  #$80        ;Bit 7 setzen
098A: 20 ED FD 165            JSR  COUT        ;ausgeben
098D: E6 04    166  WEITER    INC  PUFFER
098F: CA       167            DEX
0990: 10 F0    168            BPL  OUT1
0992: A9 8D    169            LDA  #$8D        ;Return
0994: 20 ED FD 170            JSR  COUT
0997: C6 08    171            DEC  ALT
0999: D0 10    172            BNE  PULL
099B: 20 0C FD 173            JSR  RDKEY       ;Tastendruck
099E: C9 9B    174            CMP  #$9B        ;Escape ?
09A0: D0 02    175            BNE  NEU
09A2: F0 27    176            BEQ  FILE        ;sofort
               177  *
09A4: 20 58 FC 178  NEU       JSR  HOME        ;neue Seite
09A7: A9 14    179            LDA  #20
09A9: 85 08    180            STA  ALT
09AB: AD 3A 09 181  PULL      LDA  LADEN+1
09AE: 18       182            CLC
09AF: 90 90    183            BCC  ADD         ;immer
               184  *
               185  *
               186  * CP/M-Filename (Eingabefile)
               187  *
09B1: 20 42 FC 188  ZULANG    JSR  CLREOP      ;löschen
09B4: 20 D9 0C 189            JSR  PRINT
09B7: 87 87 8D 190            HEX  87878D      ;Piep
09BA: C5 C9 CE 191            ASC  "EINGABE ZU LANG"
09C9: 8D 00    192            HEX  8D00
               193  *
09CB: 20 D9 0C 194  FILE      JSR  PRINT
09CE: A0 8D    195            HEX  A08D
09D0: D7 C5 CC 196            ASC  "WELCHEN WORDSTAR-FILE "
09E6: C2 C5 C1 197            ASC  "BEARBEITEN ?"
09F2: 8D 00    198            HEX  8D00
09F4: A9 3E    199            LDA  #$3E
09F6: 85 33    200            STA  PROMPT
09F8: 20 6A FD 201            JSR  GETLN       ;Eingabe von
09FB: E0 0D    202            CPX  #13         ;max. 12 Z
09FD: B0 B2    203            BGE  ZULANG
09FF: 8A       204            TXA
0A00: D0 03    205            BNE  LÖSCH       ;nur Return
0A02: 4C 6E 08 206            JMP  MENUE1      ;zurück
               207  *
               208  *
               209  * Lösche Eingabefeld
               210  *
0A05: 85 08    211  LÖSCH     STA  ALT         ;Länge
0A07: A9 20    212            LDA  #$20
0A09: A0 0A    213            LDY  #10
0A0B: 99 BB 0D 214  LOOP2     STA  INFILE,Y
0A0E: 88       215            DEY
0A0F: 10 FA    216            BPL  LOOP2
               217  *
               218  *
               219  * Tastaturpuffer -> Eingabefeld
               220  *
0A11: C8       221  INCREM0   INY
0A12: C4 08    222            CPY  ALT         ;alles über-
0A14: B0 3B    223            BGE  FERTIG      ;tragen ?
0A16: B9 00 02 224            LDA  IN,Y        ;Tastaturpuffer
0A19: 29 7F    225            AND  #$7F        ;Bit 7 löschen
0A1B: C9 2E    226            CMP  #$2E        ;Punkt
0A1D: F0 09    227            BEQ  PUNKT
0A1F: C0 08    228            CPY  #8          ;max. Länge
```

# Nutzen Sie Ihren Apple-Computer auch zum Lernen

INTUS-Lernprogramme sind attraktiv und motivierend für vergnügliches Lernen mit hohem Lernerfolg.
Alle Programme in deutscher Sprache für Apple IIe und IIc.

- Maschinenschreiben wie der Blitz. Didaktisch ausgezeichnet. Garantiert erfolgreich in 20 Lektionen. DM 188,—
- Basic-Lernprogramm. Bestes Lernprogramm 1982 in USA. DM 295,—
- Computer-Simulator. Mit Lehrgang in 5 Lektionen. DM 146,—
- Lesen wie der Blitz. Sehr wirkungsvolles Lesetraining. DM 115,—
- Deutsche Grammatik mit Spaß. Für Sekretärinnen, Schüler und "Jedermann". Beste Beurteilungen durch Verwender.
  - Rechtschreibetraining mit Vortest/Nachtest, 5 Bereiche, 4 Disk. je Disk. DM 165,—
  - Zeichensetzung, Wortarten, Verb, Zeitenfolge, Adjektive/Adverbien, Fälle, Satzglieder, Haupt- und Nebensätze 8 Disk. (weitere folgen) je Disk. DM 165,—
- Wortschatztrainer, Franz./Deutsch, 800 Wörter DM 140,—
  Englisch/Deutsch, 800 Wörter DM 140,—
  Business English, 1200 Wörter DM 163,—
- Unternehmens-Planspiel "Markt + Gewinn". Professionelles, lehrreiches Strategiespiel. Vorkenntnisse notwendig. Umfangreiche Dok. DM 495,—
- Schulprogramme wie Rechnen/Mathe, Sprachen, Physik, Chemie, Informatik, Vorschule. Bestens für die Nachhilfe geeignet.

★★★★ Mit Ihrer ersten Bestellung erhalten Sie gratis das Programm "Mein Freund der Apple" mit interessanten Denk-Strategie- und Entspannungsspielen.

INTUS LERN-SYSTEME AG

Kaiserstraße 21 · 7890 Waldshut-Tiengen · Telefon 07751-7920
Schweiz: 6981 Astano · Telefon 091-732551/042-223113

# Apple IIc

**CPM-Karte**

**Preis DM 650,–**

**sofort lieferbar**

**Computerware GmbH**

**Wilhelm-Leuschner-Straße 34**
**6000 Frankfurt 1**
**Tel. 069/23 67 13**
**23 67 16**

**APPLE — DISKETTEN LAUFWERKE — APPLE**

| | | |
|---|---|---|
| Original Disk II m. Controller + DOS 3.3 + Hdbuch | DM | 995,— |
| Original Disk II (2. Laufwerk) | DM | 765,— |
| Duo-Disk Station Slimline 2 x 143KB Chinon | DM | 995,— |
| Siemens Disk-Laufw. 143KB, m. Kabel + Gehause | DM | 585,— |
| Abor-Disk-Laufw. Slimline 143KB, m. Kab. + Geh. | DM | 435,— |
| Chinon Slimline 143KB Superleise, m. Kab. + Geh. | DM | 495,— |
| Teac 55F, 1 MB unf. Kapazität, Shugartbus | DM | 695,— |
| Teac 55F, kpl. in Gehause, 40/80 Track, 1MByte anschlußfertig, mit Umschaltung 40/80 + Kabel | DM | 798,— |
| Zweitlaufwerk fur Apple II C, 143 KB, mit Kabel | DM | 498,— |

**APPLE — INTERFACES + MAINBOARDS — APPLE**

| | | |
|---|---|---|
| Disk Controller f. 2 Original o. kompat. Drives | *DM | 119,— |
| Super-Controller f. 2x Teac 55F, mit Software | **DM | 239,— |
| 80 Zeich-Karte mit 64K RAM + Softsw. f. IIe | DM | 395,— |
| 80 Zeich-Karte II,m. Softswitch, 2 Zeichensätze | *DM | 229,— |
| 16K RAM Erweiterung fur II und kompatible | *DM | 119,— |
| Z 80A Interface Karte fur CPM 2.2 | *DM | 119,— |
| Z 80B Interface Karte fur CPM 3.0, m. 64K RAM | DM | 595,— |
| Printer-Grafik Interface, Epson kompatible | *DM | 139,— |
| Centronic-Parallel Text-Interface Karte | *DM | 98,— |
| Printer-Grafik Interface, NEC/ITOH kompatibel | *DM | 169,— |
| Anschluß Kabel f. Parallel + Grafik Interface | DM | 34,— |
| Buffer Grafik-Interface, benutzbar für Drucker Epson/Nec/Itoh/Okidata u. andere m. 32K Buffer | DM | 395,— |
| Buffer Interface wie vor aber mit 64K Buffer | **DM | 595,— |
| Anschluß Kabel f. Buffer Interface Karte | DM | 39,— |
| 232C Serielle Interface Karte | *DM | 119,— |
| Super Serielle Interface Karte, Fullduplex | **DM | 288,— |
| Sprach (Speech) Karte f. Sprachwiedergabe | *DM | 98,— |
| 6522 Parallel Interface Karte | *DM | 149,— |
| Clock Karte (Datum/Uhrzeit) Ein/Ausgabe | *DM | 129,— |
| Eprom Writer Karte (2716 - 2764) | *DM | 149,— |
| IEEE-488 Interface Karte | *DM | 398,— |
| Logo-Karte mit Diskette und Handbuch | DM | 498,— |
| Musik Karte m. Diskette und Handbuch | *DM | 119,— |
| PAL Color Interface Karte (UHF + Video) | *DM | 129,— |
| RGB Interface Karte (f. Apple II + II+) | *DM | 169,— |
| Wild Karte (kopiert uber RAM-Bereich) | *DM | 149,— |
| 128K RAM Erweiterungs Karte m. Patchsoftware | **DM | 548,— |
| 256K RAM Erweiterungs Karte m. Patchsoftware | **DM | 998,— |
| 6809 Prozessor Exell-9 Interfacekarte | + DM | 488,— |
| IC-Tester Interface Karte (RAM/S/TTL 54/74) | DM | 448,— |
| Hauptplatine 48K, o. Firmware Eproms, 8 Slots | DM | 548,— |
| Hauptplatine 64K, wie vor | DM | 699,— |

**APPLE — LEERPLATINEN — APPLE**

Leerplatinen der obigen Interface Karten sind alle in vergoldet.

| | | |
|---|---|---|
| m. Bestuck-Aufdruck + Best.-Plan lieferbar | | |
| Leerplatinen mit der Kennzeichnung | **DM | 39,— |
| Leerplatinen mit der Kennzeichnung | *DM | 24,50 |
| Leerplatinen mit der Kennzeichnung | + DM | 33,— |
| Experimentier Platine, fur Apple Slot EX300 | DM | 29,90 |
| Experimentier Platine, fur Apple Slot EX500 | DM | 19,98 |
| Leerplatine Motherboard, 48K, mit Best.-Aufdruck | DM | 66,— |
| Leerplatine Motherboard, 64K, mit 6502 + Z80 | DM | 99,— |

100% Apple-kompatibel bei Verwendung des Apple-Betriebssystems

**6 Mon. Garantie** **Reparaturservice**

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

| | | |
|---|---|---|
| Standard Einbau-Tastatur, ASCII, freibel. 9 Tasten Doppelbelg., alter Tasten, Groß + Kleinschr. Nr. N26 | DM | 139,— |
| Tastatur wie vor, jedoch mit 15er Block Nr. N67 | DM | 178,— |
| Tastatur Nr. N67, mit sep. Gehause, kpl. m. Kabel | DM | 249,— |
| Seperate Tastatur IBM-Look, anschlußfertig mit Kabel, Dopp. belegt. Tasten, frei prog. Funkt. Tasten | DM | 298,— |
| Seperate Tastatur, Eprom progr. bar, Cursorblock Tastenfeld mit 24 Funktionen, Deutsch o. ASCII | DM | 398,— |
| Leergehause Standard, passend f. Tastatur Nr. N26 | DM | 98,— |
| dto. wie vor, jedoch passend fur 15er Tast. N67 | DM | 119,— |
| Leergehause wie Mewa 9000, fur Einbau von zwei Slimline-Laufwerken + Tastaturanschluß, Plastic | DM | 189,— |
| Leergehäuse wie vor, jedoch in Metall-Ausfuhrung | DM | 198,— |
| Leergehause IBM-Look, f. 2 Laufw. Standard 5 | DM | 245,— |
| Leergehause fur 1 Slimline Laufwerk, Metall | DM | 19,90 |
| Leergehause f. Duo-Disk = 2x Slimline-Laufwerke | DM | 98,— |
| Leergehause f. Duo-Disk = 2x Standard-Laufw. | DM | 109,— |
| Leergehause f. Duo-Disk = 2x Slimline + Netzteil | DM | 179,— |

Schaltnetzteile fur APPLE u. kompatible Rechner

| | | |
|---|---|---|
| + 5V/3,5A -5V/0,5A + 12V/2A -12V/0,5A N73 | DM | 99,80 |
| + 5V/5 A -5V/0,5A + 12V/2,5A -12V/0,5a N74 | DM | 119,80 |
| + 5V/7,5A -5V/0,5A + 12V/2A -12V/0,5A N75 | DM | 149,50 |

**APPLE — DISKETTEN + DISKETTEN-BOXEN —**

| | | |
|---|---|---|
| Profi Disk Box, m. Schloß, Klars.-Deckel, 100 Disks | DM | 49,— |
| Disk Box mit Klars.-Deckel ca. 70 Disketten 5 | DM | 26,— |
| Hartbox fur 10 Disketten, mit Klappdeckel | DM | 6,90 |
| Disketten mit Verstarkungsring, 3 Jahre Garantie | | |
| ABX 100 SS/SD 10 Stck. a. 3,98 50 Stck. a. 3,88 500 Stck. a. 3,59 | | |
| ABX 200 - SS/DS 10 Stck. a. 4,08 50 Stck. a. 3,98 500 Stck. a. 3,69 | | |
| ABX 400 - DS/DD 10 Stck. a. 4,98 50 Stck. a. 4,88 500 Stck. a. 4,59 | | |

**APPLE — ORIGINAL + KOMPATIBLE - COMPUTER — APPLE**

| | | |
|---|---|---|
| APPLE IIe 64KRAM, Ascii-Tastat. + UHF-Modul | DM | 2198,— |
| APPLE IIe Einstiegspaket: Computer 64KRAM, + Philips-Monitor 18 Mhz grun + Siemens F122 - Laufwerk 143K m. Controller + Lerndiskette | DM | 3098,— |
| APPLE II C 128 KRAM, ASCII Tastatur | DM | 2795,— |
| APPLE II C, wie vor, jedoch deutsche Tastatur | DM | 2998,— |
| MACINTOSH System komplett m. Mouse + Tastatur + Macwrite/Macpaint, Systemguide-Diskette | DM | 5995,— |
| MEWA-48K, Apple kompat., Netzt. 5Amp. Tastatur mit Dopp-Belegung + 10 frei Prog. Tasten, mit UHF-Mod. | DM | 998,— |
| MEWA-64K, wie vor jedoch mit 15er Block | DM | 1098,— |
| MEWA 9000er Serie, mit seper. Tastat. DIN o. ASCII m. dopp. belegt. Funkt-Tasten, Cursorbl. + 15er Block Netzt. 5 Amp. Gehause fur 2x Slimline-Laufwerke UHF-Modulator o. Firmw.-Proms (12KROM), 48K RAM | DM | 1198,— |
| MEWA 9000-64-C, wie vor jedoch 64K RAM | DM | 1298,— |
| MEWA 9000-64-C Einstiegspakel wie vor jedoch mit 1 Disk Drive eingeb. + Disk-Contr. + Monitor grun | DM | 2198,— |
| IBM-Look Gehause fur MEWA 9000 Serie Aufpreis | DM | 68,— |

**MATRIX- UND TYPENRAD — DRUCKER —**

| | | |
|---|---|---|
| CP-80 Matrix Drucker, Epson kompat., vollgrafikf. | DM | 739,— |
| CP-80 X mit eingeb. Interface fur VC 64 f. Sim. B. | DM | 899,— |
| SPEEDY 100-80, Epson kompat. 100 Z/s, v.-grafikf. | DM | 759,— |
| GEMINI 10X, Neu mit Textsp., 100 Z/s., 9x9 Mat. | DM | 998,— |
| GEMINI 15X, wie 10X jedoch bis 375 mm Papierbr. | DM | 1198,— |
| ITOH 8510 B, Die Zuverlassigkeit in Person. | DM | 1475,— |
| JUKI 6100, Typenraddr., 22 Z/s, TA-Typenrader | DM | 1495,— |

**IBM *** KOMPATIBEL ***** — *** KOMPATIBEL**

Alle Teile unterliegen einer sorgfaltigen Endkontrolle und wir übernehmen daher volle Garantie fur die Funktionsfähigkeit sowohl fur die gelieferten Leerplatinen, als auch fur die fertig bestuckten Boards und Interface Karten, die fast ohne Ausnahme mit gesockelten IC's geliefert werden.

**IBM — Kompatible Interface Karten und Mainboards — IBM**

| | | | |
|---|---|---|---|
| Disk Controller Karte fur 2 Disk-Drives | *KL-2020 | DM | 399,— |
| Disk Controller Karte fur 2 Disk-Drives | *KL-2021 | DM | 499,— |
| Color Video Board Farb- und Video Ausg. | *KL-2050 | DM | 799,— |
| Monochrome Video Boards | *KL-2060 | DM | 499,— |
| RS 232 Drucker Interface Karte | KL-2074 | DM | 298,— |
| Centronics Parallel Interface Karte | KL-2072 | DM | 199,— |
| Multifunktion Karte, RAM-Bereich, RS232 Parallel, Clock, mit O.K bestuckt | *KL-2040 | DM | 728,— |
| dto. wie vor jedoch mit 128K bestuckt | *KL-2041 | DM | 998,— |
| dto. wie vor, jedoch mit 256K bestuckt | *KL-2042 | DM | 1288,— |
| 512K RAM Karte, mit O.K bestuckt | KL-2095 | DM | 399,— |
| dto. wie vor, mit 128K bestuckt | KL-2096 | DM | 689,— |
| Multi I/O Interface Board, RS232, Disk-Controller Centronics, Clock, Joystick und Lightpen-Port | *KL-2071 | DM | 998,— |
| Eprom Writer Karte (bis 128K benutzbar) | | | a. A. |
| Hardisk/Winchester Host-Adapter Karte | *KL-2022 | DM | 1188,— |

Mainboard XT Version, 8 Slots, 128K bestuckt

| | | | |
|---|---|---|---|
| mit 1 Boot-Eprom, 6 Eprom-Plätze frei | *KL-1004 | DM | 1499,— |
| dto. wie vor, jedoch mit 256K bestuckt | *KL-1005 | DM | 1789,— |
| dto. wie vor, jedoch mit 512K bestuckt | *KL-1006 | DM | 2079,— |

Mainboard PC Version, 8 Slots, 64K bestuckt

| | | | |
|---|---|---|---|
| mit 1 Boot-Eprom, 6 Eprom-Platze frei | *KL-1010 | DM | 999,— |
| dto. wie vor, jedoch mit 128K bestuckt | *KL-1011 | DM | 1269,— |
| dto. wie vor, jedoch mit 256K bestuckt | *KL-1012 | DM | 1579,— |

Bei den mit * gekennzeichneten Artikeln gehort zum Lieferumfang ein Manual und z. T. ein Schaltbild

**IBM — Kompatible Leerplatinen + Boards —**

| | | |
|---|---|---|
| Disk Controller f. 2 Drives | DM | 78,— |
| Disk Controller f. 4 Drives | DM | 88,— |
| Color Grafik Video Board | DM | 198,— |
| RS232 Interface Karte | DM | 88,— |
| Centronics Parallel Interface Karte | DM | 88,— |
| Multifunktion Interface Board | DM | 128,— |
| Eprom Writer Interface Karte | | a. A. |
| Multi I/O Board | DM | 118,— |
| Mainboard XT Version, 8 Slots | DM | 248,— |
| Mainboard PC Version, 8 Slots | DM | 198,— |
| RAM Card fur Speicher-Erweiterung | DM | 98,— |

Alle Leerplatinen werden mit Bestuckungsplan geliefert
Schaltbilder soweit verfugbar. Lieferung gegen Aufpreis

**IBM — Gehäuse/Netzteile/Tastaturen/Diskdrives — IBM**

| | | |
|---|---|---|
| Rechner Leergehause f. Einbau v. 2 Drives | DM | 236,— |
| Profi-Tastatur m. Kabel im Gehause, ASCII | DM | 395,— |
| dto. wie vor jedoch deutsche DIN Belegung | DM | 489,— |
| Standard Tastatur IBM-Look, ASCII | DM | 298,— |
| Netzteil, mit Ventilator, 100 Watt | DM | 348,— |
| dto. wie vor jedoch 135 Watt | DM | 398,— |
| Disketten Laufwerk, 2x40 Track DS/360 KB | DM | 498,— |
| Harddisk 10MByte, kpl. m. Xebec Controller | DM | 3990,— |

Computer-Artikel Nachnahmeversand unfrei. Zwischenverkauf vorbehalten
Angebote freibleibend unter Anerkennung unserer Lieferbedingungen. Technische Änderungen vorbehalten. *Apple ist eingetragenes Warenzeichen der Fa. Apple-Computer Inc., Kalifornien. Ware mit Rückgaberecht, besonders gekennzeichnet, muß frei zurückgeschickt werden. *IBM* ist eingetragenes Warenzeichen der Firma IBM GmbH Flm

**COMPUTER CENTER CONEX**
5650 SOLINGEN 11 Postfach 11 02 06 -11 G
Telefon (0 21 22) 7 54 49

+

**ERICH-WILLI MEYER**
6343 FROHNHAUSEN Postfach 70 11
Telefon (0 27 71) 3 50 71

**Teac 55F**

Disk-Station komplett anschlußfertig / umschaltbar auf Apple-Format
Patch-Software für Dos, Fastdos, UCSD, CP/M 58 u. 60K inkl. / CPM+, Prodos auf Anf.
Einzelstation 640K - 978 DM / Doppelstation 1,2 MByte - 1698 DM / ext. Netzteil + 150 DM

((Disketten Sentinel in Plastik-Klappbox • SS/DD 47 DM • DS/DD 96Tpl für Teac 55F 80 DM )))
Bei Einsendung der Betriebssysteme kostenloser Patchservice zu bootbaren 160 Track Disketten !!

• JLK GBR • Eugen Twickler • 4400 Münster • Weseler Str. 61 • Tel. 0251/44921 •••

```
ØA21: BØ 8E     229           BGE  ZULANG     ;bis zum Punkt
ØA23: 99 BB ØD  23Ø           STA  INFILE,Y   ;Eingabefeld
ØA26: 9Ø E9     231           BLT  INCREMØ    ;immer
                232  *
ØA28: A2 Ø7     233  PUNKT    LDX  #7
ØA2A: E8        234  INCREM1  INX
ØA2B: C8        235           INY
ØA2C: C4 Ø8     236           CPY  ALT        ;alles über-
ØA2E: BØ 21     237           BGE  FERTIG     ;tragen
ØA3Ø: B9 ØØ Ø2  238           LDA  IN,Y       ;Tastaturpuffer
ØA33: 29 7F     239           AND  #$7F       ;Bit 7 löschen
ØA35: 9D BB ØD  24Ø           STA  INFILE,X   ;Eingabefeld
ØA38: 9Ø FØ     241           BLT  INCREM1    ;immer
                242  *
                243  *
                244  * DOS-Filename (Ausgabefile)
                245  *
ØA3A: 2Ø D9 ØC  246  ZULANG1  JSR  PRINT
ØA3D: 87 87 8D  247           HEX  87878D     ;Piep
ØA4Ø: C5 C9 CE  248           ASC  "EINGABE ZU LANG"
ØA4F: 8D ØØ     249           HEX  8DØØ
                25Ø  *
ØA51: 2Ø 42 FC  251  FERTIG   JSR  CLREOP     ;Lösch bis
ØA54: 2Ø D9 ØC  252           JSR  PRINT      ;Seitenende
ØA57: 8D 8D     253           HEX  8D8D
ØA59: CE C1 CD  254           ASC  "NAME FUER DOS-FILE ?"
ØA6D: 8D ØØ     255           HEX  8DØØ
ØA6F: A9 3E     256           LDA  #$3E       ;Prompt-Zeichen
ØA71: 85 33     257           STA  PROMPT
ØA73: 2Ø 6A FD  258           JSR  GETLN      ;Eingabe von
ØA76: EØ 1F     259           CPX  #31        ;max. 3Ø Z
ØA78: BØ CØ     26Ø           BGE  ZULANG1
ØA7A: 8A        261           TXA
ØA7B: DØ Ø3     262           BNE  LESEN      ;nur Return
ØA7D: 4C 6E Ø8  263           JMP  MENUE1     ;zurück
                264  *
                265  *
                266  * Tastaturpuffer -> Ausgabefeld
                267  *
ØA8Ø: BD ØØ Ø2  268  LESEN    LDA  IN,X       ;Tastaturpuffer
ØA83: Ø9 8Ø     269           ORA  #$8Ø
ØA85: 9D 9D ØD  27Ø           STA  OUTFILE,X  ;Ausgabefeld
ØA88: CA        271           DEX             ;alles über-
ØA89: 1Ø F5     272           BPL  LESEN      ;tragen ?
                273  *
                274  *
                275  * CP/M File im Directory suchen
                276  *
ØA8B: A9 ØØ     277  SUCHEN   LDA  #Ø         ;initialisiere
ØA8D: 85 Ø6     278           STA  IND        ;Blockzeiger
ØA8F: 85 Ø7     279           STA  IND+1      ;Durchläufe
ØA91: 85 ØØ     28Ø           STA  PTR        ;Extension
ØA93: 85 Ø1     281           STA  PTR+1      ;Flagge
ØA95: A9 1Ø     282  SUCHEN1  LDA  #$1Ø       ;Pufferbeginn
ØA97: 85 Ø5     283           STA  PUFFER+1   ;bei $1ØØØ
ØA99: AØ ØØ     284           LDY  #Ø
ØA9B: 84 Ø4     285           STY  PUFFER
ØA9D: FØ 13     286           BEQ  SUCHEN3    ;immer
                287  *
                288  *
                289  * Suche nächstes nicht gelöschtes Feld
                29Ø  *
ØA9F: 18        291  SUCHEN2  CLC
ØAAØ: AØ ØØ     292           LDY  #Ø
ØAA2: A5 Ø4     293           LDA  PUFFER
ØAA4: 69 2Ø     294           ADC  #$2Ø       ;Abstand der
ØAA6: 85 Ø4     295           STA  PUFFER     ;Eintragungen
ØAA8: 9Ø Ø8     296           BCC  SUCHEN3
ØAAA: E6 Ø5     297           INC  PUFFER+1
ØAAC: A5 Ø5     298           LDA  PUFFER+1
ØAAE: C9 18     299           CMP  #$18       ;Pufferende
ØABØ: BØ 2C     3ØØ           BGE  FEHLER     ;bei $17FF
                3Ø1  *
ØAB2: B1 Ø4     3Ø2  SUCHEN3  LDA  (PUFFER),Y ;Index lesen
ØAB4: C9 E5     3Ø3           CMP  #$E5
ØAB6: FØ E7     3Ø4           BEQ  SUCHEN2    ;File gelöscht
                3Ø5  *
                3Ø6  *
ØAB8: AØ ØC     3Ø7  SUCHEN4  LDY  #12        ;Extension ?
ØABA: B1 Ø4     3Ø8           LDA  (PUFFER),Y
ØABC: C5 ØØ     3Ø9           CMP  PTR        ;stimmt ??
ØABE: DØ DF     31Ø           BNE  SUCHEN2
                311  *
                312  *
                313  * Vergleiche Eintrag und Eingabe
                314  *
ØACØ: A2 FF     315  COMP     LDX  #$FF       ; $ØØ - 1
```

```
ØAC2: AØ ØØ     316           LDY  #$Ø        ; $Ø1 - 1
ØAC4: C8        317  COMP1    INY
ØAC5: E8        318           INX
ØAC6: EØ ØB     319           CPX  #11        ;max. Länge
ØAC8: BØ ØB     32Ø           BGE  SUCHEN5
                321  *
ØACA: B1 Ø4     322  COMP2    LDA  (PUFFER),Y
ØACC: 29 7F     323           AND  #$7F       ;Bit 7 löschen
ØACE: DD BB ØD  324           CMP  INFILE,X
ØAD1: FØ F1     325           BEQ  COMP1      ;OK!
ØAD3: DØ CA     326           BNE  SUCHEN2    ;n. gefunden
                327  *
                328  *
ØAD5: E6 Ø1     329  SUCHEN5  INC  PTR+1      ;gefunden
ØAD7: 2Ø Ø5 ØB  33Ø           JSR  BLOCKM     ;Blockmap
ØADA: DØ B9     331           BNE  SUCHEN1    ;nächste Ext.
ØADC: FØ 56     332           BEQ  ALLES      ;alles gefund.
                333  *
                334  *
                335  * Fehlermeldung, wenn File nicht gefunden
                336  *
ØADE: A5 Ø1     337  FEHLER   LDA  PTR+1      ;Flagge
ØAEØ: DØ 52     338           BNE  ALLES
ØAE2: 2Ø D9 ØC  339           JSR  PRINT
ØAE5: 87 87 8D  34Ø           HEX  87878D     ;Piep
ØAE8: C3 DØ AF  341           ASC  "CP/M-FILE NICHT GEFUNDEN"
ØBØØ: 8D ØØ     342           HEX  8DØØ
ØBØ2: 4C 6B Ø8  343           JMP  MENUE      ;zurück
                344  *
                345  *
                346  * Blockmap zusammenstellen
                347  *
ØBØ5: A6 Ø6     348  BLOCKM   LDX  IND        ;Indexzeiger
ØBØ7: AØ 1Ø     349           LDY  #16        ;1. Block
ØBØ9: B1 Ø4     35Ø  BM1      LDA  (PUFFER),Y
ØBØB: DØ Ø3     351           BNE  BM2
ØBØD: 86 Ø6     352           STX  IND        ;Ende gefund.
ØBØF: 6Ø        353           RTS             ;Z gesetzt
                354  *
ØB1Ø: 9D C6 ØD  355  BM2      STA  BLOCKMAP,X
ØB13: E8        356           INX
ØB14: C8        357           INY
ØB15: CØ 2Ø     358           CPY  #32        ;16. BLOCK
ØB17: 9Ø FØ     359           BLT  BM1
ØB19: 86 Ø6     36Ø           STX  IND
ØB1B: E6 ØØ     361           INC  PTR        ;nächste Ext.
ØB1D: 6Ø        362           RTS             ;Z gelöscht
                363  *
                364  *
                365  * Leerer File mit Ø Blocks
                366  *
ØB1E: 2Ø D9 ØC  367  LEER     JSR  PRINT
ØB21: 87 87 8D  368           HEX  87878D     ;Piep
ØB24: CC C5 C5  369           ASC  "LEERER FILE"
ØB2F: 8D ØØ     37Ø           HEX  8DØØ
ØB31: 4C 6B Ø8  371           JMP  MENUE
                372  *
                373  *
                374  * Alles in Ordnung? Sonst zurück
                375  *
ØB34: A5 Ø6     376  ALLES    LDA  IND        ;Blockzahl
ØB36: FØ E6     377           BEQ  LEER       ;leerer File
ØB38: 2Ø D9 ØC  378           JSR  PRINT
ØB3B: 8D        379           HEX  8D
ØB3C: BC D2 C5  38Ø           ASC  "<RETURN> = FERTIG,"
ØB4E: AØ BC C5  381           ASC  " <ESC> = ZURUECK"
ØB5E: AØ ØØ     382           HEX  AØØØ
ØB6Ø: 2Ø ØC FD  383  READKEY  JSR  RDKEY      ;Tastendruck
ØB63: C9 8D     384           CMP  #$8D       ;Return
ØB65: FØ Ø7     385           BEQ  FUND
ØB67: C9 9B     386           CMP  #$9B       ;Escape
ØB69: DØ F5     387           BNE  READKEY
ØB6B: 4C 6E Ø8  388           JMP  MENUE1
                389  *
                39Ø  *
                391  * Track und Sektoren der Blöcke berechnen
                392  * sowie sektorweise nach "Puffer" lesen
                393  *
ØB6E: 2Ø D9 ØC  394  FUND     JSR  PRINT
ØB71: 8D 8D     395           HEX  8D8D
ØB73: C7 C5 CC  396           ASC  "GELESENE BLOECKE:"
ØB84: AØ AØ ØØ  397           HEX  AØAØØØ
ØB87: A9 1Ø     398  FUND1    LDA  #$1Ø       ;Pufferbeginn
ØB89: 85 Ø5     399           STA  PUFFER+1   ;bei $1ØØØ
ØB8B: A9 ØØ     4ØØ           LDA  #$Ø
ØB8D: 85 Ø4     4Ø1           STA  PUFFER
ØB8F: A6 Ø7     4Ø2           LDX  IND+1      ;geles. Blöcke
```

```
ØB91: 86 Ø7     4Ø3  LADEN1   STX  IND+1       ;retten
ØB93: BD C6 ØD  4Ø4          LDA  BLOCKMAP,X ;Block
ØB96: 48        4Ø5          PHA              ;retten
ØB97: 2Ø D1 ØB  4Ø6          JSR  PRHEXZ      ;HEX ausgeben
ØB9A: 68        4Ø7          PLA              ;Block holen
ØB9B: 48        4Ø8          PHA              ;und zurück
ØB9C: 4A        4Ø9          LSR              ; :4
ØB9D: 4A        41Ø          LSR
ØB9E: 18        411          CLC
ØB9F: 69 Ø3     412          ADC  #3          ; +3
ØBA1: 85 Ø2     413          STA  TRACK       ;= Track
ØBA3: 68        414          PLA
ØBA4: 29 Ø3     415          AND  #3          ;%ØØØØØØ11
ØBA6: ØA        416          ASL              ; *4
ØBA7: ØA        417          ASL
ØBA8: A8        418          TAY              ;Tabellenbeg.
ØBA9: B9 C9 ØC  419  SEKT    LDA  TAB2,Y
ØBAC: 85 Ø3     42Ø          STA  SECTOR      ;= Sektor
ØBAE: 84 Ø1     421          STY  PTR+1
ØBBØ: A9 AE     422          LDA  #$AE        ;Punkt/Sektor
ØBB2: 2Ø FØ FD  423          JSR  COUT1       ;ausgeben
ØBB5: 2Ø 5C ØC  424          JSR  READ        ;lesen
ØBB8: A4 Ø1     425          LDY  PTR+1
ØBBA: E6 Ø5     426          INC  PUFFER+1
ØBBC: A5 Ø5     427          LDA  PUFFER+1
ØBBE: C9 94     428          CMP  #$94        ;Pufferende
ØBCØ: BØ 25     429          BGE  TFILE       ;bei $94ØØ
ØBC2: C8        43Ø          INY
ØBC3: 98        431          TYA
ØBC4: 29 Ø3     432          AND  #3          ;alle vier
ØBC6: DØ E1     433          BNE  SEKT        ;Sekt. lesen
ØBC8: A6 Ø7     434          LDX  IND+1
ØBCA: E8        435          INX              ;nächst. Block
ØBCB: E4 Ø6     436          CPX  IND         ;alle Blocks ?
ØBCD: 9Ø C2     437          BLT  LADEN1
ØBCF: BØ 16     438          BGE  TFILE       ;Ja
                439  *
                44Ø  *
                441  * Byte als Hexzahl ausg. (mod. Apple-ROM)
                442  *
ØBD1: 48        443  PRHEXZ   PHA
ØBD2: 4A        444          LSR
ØBD3: 4A        445          LSR
ØBD4: 4A        446          LSR
ØBD5: 4A        447          LSR
ØBD6: 2Ø DC ØB  448          JSR  PRHEX       ;Hi
ØBD9: 68        449          PLA
ØBDA: 29 ØF     45Ø          AND  #$ØF        ;Lo
ØBDC: Ø9 BØ     451  PRHEX    ORA  #$BØ
ØBDE: C9 BA     452          CMP  #$BA
ØBEØ: 9Ø Ø2     453          BCC  OUTPUT
ØBE2: 69 Ø6     454          ADC  #$Ø6
ØBE4: 4C FØ FD  455  OUTPUT   JMP  COUT1
                456  *
                457  *
                458  * Öffne DOS-Textfile, simuliere lau-
                459  * fendes Applesoftprogramm für DOS
                46Ø  *
ØBE7: A5 Ø7     461  TFILE    LDA  IND+1       ;erster
ØBE9: C9 21     462          CMP  #$21        ;Durchgang ?
ØBEB: BØ 3E     463          BGE  REFINE
ØBED: A9 8Ø     464          LDA  #$8Ø        ;Bit 7 setzen
ØBEF: 85 33     465          STA  PROMPT
ØBF1: 85 76     466          STA  CURLIN+1
ØBF3: 85 D9     467          STA  RUNMODE
ØBF5: 2Ø D9 ØC  468          JSR  PRINT
ØBF8: 8D 84     469          HEX  8D84        ;CR, Ctrl-D
ØBFA: CF DØ C5  47Ø          ASC  "OPEN"
ØBFE: ØØ        471          HEX  ØØ
ØBFF: AØ ØØ     472          LDY  #Ø
ØCØ1: B9 9D ØD  473  OPEN     LDA  OUTFILE,Y
ØCØ4: C9 8D     474          CMP  #$8D        ; CR ?
ØCØ6: FØ Ø6     475          BEQ  WRITE
ØCØ8: 2Ø ED FD  476          JSR  COUT
ØCØB: C8        477          INY
ØCØC: DØ F3     478          BNE  OPEN
ØCØE: 2Ø D9 ØC  479  WRITE    JSR  PRINT
ØC11: AC C4 B1  48Ø          ASC  ",D1"
ØC14: 8D 84     481          HEX  8D84        ;CR,Ctrl-D
ØC16: D7 D2 C9  482          ASC  "WRITE"
ØC1B: ØØ        483          HEX  ØØ
ØC1C: AØ ØØ     484          LDY  #Ø
ØC1E: B9 9D ØD  485  WRITE1   LDA  OUTFILE,Y
ØC21: 2Ø ED FD  486          JSR  COUT
ØC24: C9 8D     487          CMP  #$8D        ; CR ?
ØC26: FØ Ø3     488          BEQ  REFINE
ØC28: C8        489          INY
```

```
ØC29: DØ F3     49Ø          BNE  WRITE1
                491  *
                492  *
                493  * Führe Wordstar-Umwandlungen aus
                494  *
ØC2B: 2Ø FA ØC  495  REFINE   JSR  REFINER
ØC2E: E6 Ø7     496          INC  IND+1       ;alle
ØC3Ø: A5 Ø6     497          LDA  IND         ;Blöcke
ØC32: C5 Ø7     498          CMP  IND+1       ;gelesen ?
ØC34: FØ Ø3     499          BEQ  CLOSE
ØC36: 4C 87 ØB  5ØØ          JMP  FUND1       ;weiter
                5Ø1  *
                5Ø2  *
                5Ø3  * Schließe Textfile, zurück zum Menü
                5Ø4  *
ØC39: 2Ø D9 ØC  5Ø5  CLOSE    JSR  PRINT
ØC3C: 8D 84     5Ø6          HEX  8D84        ;CR,Ctrl-D
ØC3E: C3 CC CF  5Ø7          ASC  "CLOSE"
ØC43: 8D 8D     5Ø8          HEX  8D8D
ØC45: C6 C5 D2  5Ø9          ASC  "FERTIG"
ØC4B: 8D 84     51Ø          HEX  8D84        ;CR,Ctrl-D
ØC4D: C3 C1 D4  511          ASC  "CATALOG,D1"
ØC57: 8D ØØ     512          HEX  8DØØ
ØC59: 4C 6B Ø8  513          JMP  MENUE
                514  *
                515  *
                516  * Lese einen Sektor ein nach "Puffer"
                517  *
ØC5C: 2Ø E3 Ø3  518  READ     JSR  GETIOB      ;IOB suchen
ØC5F: 84 CE     519          STY  IOB         ;Lo-Byte
ØC61: 85 CF     52Ø          STA  IOB+1       ;Hi-Byte
ØC63: AØ Ø1     521          LDY  #1
ØC65: A9 6Ø     522          LDA  #$6Ø
ØC67: 91 CE     523          STA  (IOB),Y     ;Slot 6
ØC69: C8        524          INY
ØC6A: A9 Ø2     525          LDA  #2
ØC6C: 91 CE     526          STA  (IOB),Y     ;Drive 2
ØC6E: C8        527          INY
ØC6F: A9 ØØ     528          LDA  #Ø
ØC71: 91 CE     529          STA  (IOB),Y     ;Volume Ø
ØC73: C8        53Ø          INY
ØC74: A5 Ø2     531          LDA  TRACK
ØC76: 91 CE     532          STA  (IOB),Y     ;Track
ØC78: C8        533          INY
ØC79: A5 Ø3     534          LDA  SECTOR
ØC7B: 91 CE     535          STA  (IOB),Y     ;Sektor
ØC7D: AØ Ø8     536          LDY  #8
ØC7F: A5 Ø4     537          LDA  PUFFER
ØC81: 91 CE     538          STA  (IOB),Y     ;Puffer Lo
ØC83: C8        539          INY
ØC84: A5 Ø5     54Ø          LDA  PUFFER+1
ØC86: 91 CE     541          STA  (IOB),Y     ;Puffer Hi
ØC88: AØ ØC     542          LDY  #$ØC
ØC8A: A9 Ø1     543          LDA  #1
ØC8C: 91 CE     544          STA  (IOB),Y     ;Lesen !!
ØC8E: A4 CE     545          LDY  IOB
ØC9Ø: A5 CF     546          LDA  IOB+1
ØC92: 2Ø D9 Ø3  547          JSR  RWTS
ØC95: A9 ØØ     548          LDA  #Ø
ØC97: 85 48     549          STA  PREG        ;zurücksetzen
ØC99: BØ Ø1     55Ø          BCS  FEHLER1
ØC9B: 6Ø        551          RTS
ØC9C: 2Ø 58 FC  552  FEHLER1  JSR  HOME
ØC9F: A9 ØC     553          LDA  #12
ØCA1: 2Ø 5B FB  554          JSR  TABV
ØCA4: 2Ø D9 ØC  555          JSR  PRINT
ØCA7: C4 CF D3  556          ASC  "DOS - FEHLER NUMMER $"
ØCBC: 87 87 ØØ  557          HEX  87,87,ØØ
ØCBF: AØ ØD     558          LDY  #$ØD        ;Fehlerbyte
ØCC1: B1 CE     559          LDA  (IOB),Y
ØCC3: 2Ø DA FD  56Ø          JSR  PRBYTE
ØCC6: 4C D3 Ø3  561          JMP  DOSKALT     ;Kaltstart
                562  *
                563  *
                564  * Tabelle Blocks <--> Sektoren
                565  *
ØCC9: ØØ Ø6 ØC  566  TAB2     HEX  ØØ,Ø6,ØC,Ø3 ;Block Ø
ØCCD: Ø9 ØF ØE  567          HEX  Ø9,ØF,ØE,Ø5 ;Block 1
ØCD1: ØB Ø2 Ø8  568          HEX  ØB,Ø2,Ø8,Ø7 ;Block 2
ØCD5: ØD Ø4 ØA  569          HEX  ØD,Ø4,ØA,Ø1 ;Block 3
                57Ø  *
                571  *
                572  * Drucke einen String aus bis $ØØ
                573  * Nach Andy Hertzfeld Call Apple 8/81
                574  *
ØCD9: 68        575  PRINT    PLA              ;hole Return-
ØCDA: 85 ØØ     576          STA  PTR         ;adresse vom
```

```
ØCDC: 68          577           PLA             ;Stack
ØCDD: 85 Ø1       578           STA  PTR+1
ØCDF: AØ Ø1       579           LDY  #1
ØCE1: B1 ØØ       58Ø  PR1      LDA  (PTR),Y    ;String
ØCE3: FØ Ø6       581           BEQ  ENDE
ØCE5: 2Ø ED FD    582           JSR  COUT       ;ausgeben
ØCE8: C8          583           INY
ØCE9: DØ F6       584           BNE  PR1
ØCEB: 18          585  ENDE     CLC
ØCEC: 98          586           TYA             ;Stringlänge
ØCED: 65 ØØ       587           ADC  PTR        ;addieren u.
ØCEF: 85 ØØ       588           STA  PTR
ØCF1: A5 Ø1       589           LDA  PTR+1
ØCF3: 69 ØØ       59Ø           ADC  #Ø
ØCF5: 48          591           PHA             ;auf Stack
ØCF6: A5 ØØ       592           LDA  PTR        ;zurück,
ØCF8: 48          593           PHA             ;dann dorthin
ØCF9: 6Ø          594           RTS             ;springen
                  595  *
                  596  *
                  597  * Umcodierung spezieller Wordstar-Zeichen
                  598  *
                  599  *  Benutzt einige Routinen aus
                  6ØØ  *  dem CPM-Refiner Programm
                  6Ø1  *  von U. Stiehl, "Apple DOS
                  6Ø2  *  3.3 Tips und Tricks" S.98
                  6Ø3  *
                  6Ø4  *
                  6Ø5  *  Initialisierungen
                  6Ø6  *
ØCFA: A9 1Ø       6Ø7  REFINER  LDA  #$1Ø       ;Beginn ab
ØCFC: 85 Ø5       6Ø8           STA  PUFFER+1   ; $1ØØØ
ØCFE: A9 ØØ       6Ø9           LDA  #Ø
ØDØØ: 85 Ø4       61Ø           STA  PUFFER
ØDØ2: 8D ØØ 94    611           STA  $94ØØ      ;Maximal
ØDØ5: 85 Ø8       612           STA  ALT
                  613  *
                  614  *  Konvertierungen
                  615  *
                  616  *  Beginn mit Punktbefehl?
                  617  *
ØDØ7: AØ ØØ       618           LDY  #Ø
ØDØ9: B1 Ø4       619           LDA  (PUFFER),Y
ØDØB: C9 2E       62Ø           CMP  #$2E       ;Punkt
ØDØD: DØ ØA       621           BNE  EOT
ØDØF: 4C 8F ØD    622           JMP  SCHLEIF1   ;gefunden
ØD12: CA C2 CB    623           ASC  "JBK"
                  624  *
                  625  *
                  626  * Hauptschleife, Test auf Fileende
                  627  *
ØD15: AØ ØØ       628  SCHLEIFE LDY  #Ø
ØD17: B1 Ø4       629           LDA  (PUFFER),Y
ØD19: C9 ØØ       63Ø  EOT      CMP  #Ø         ;Endmarker
ØD1B: FØ Ø4       631           BEQ  ENDE1
ØD1D: C9 1A       632           CMP  #$1A       ;CP/M EOT
ØD1F: DØ 1B       633           BNE  SKIP
ØD21: 6Ø          634  ENDE1    RTS             ;Zurück
                  635  *
                  636  *
                  637  *  BIT 7 setzen, ausgeben
                  638  *
ØD22: Ø9 8Ø       639  AUSGABEØ ORA  #$8Ø       ;Bit 7
ØD24: A6 Ø8       64Ø  AUSGABE1 LDX  ALT        ;mit letztem
ØD26: 85 Ø8       641           STA  ALT        ;Zeichen ver-
ØD28: E4 Ø8       642           CPX  ALT        ;gleichen
ØD2A: DØ Ø4       643           BNE  AUSGABE2   ;ungleich
ØD2C: E0 AØ       644           CPX  #$AØ       ;Leerzeichen
ØD2E: FØ Ø3       645           BEQ  INCREM     ;unterdrücken
ØD3Ø: 2Ø ED FD    646  AUSGABE2 JSR  COUT       ;Zeichen senden
ØD33: E6 Ø4       647  INCREM   INC  PUFFER     ;nächstes
ØD35: DØ DE       648           BNE  SCHLEIFE   ;Zeichen
ØD37: E6 Ø5       649           INC  PUFFER+1
ØD39: DØ DA       65Ø           BNE  SCHLEIFE
ØD3B: 6Ø          651           RTS             ;Zurück
                  652  *
                  653  *
                  654  *  Zeichen ausfiltern
                  655  *
ØD3C: C9 8A       656  SKIP     CMP  #$8A       ;Zeilenvorsch.
ØD3E: FØ F3       657           BEQ  INCREM
ØD4Ø: C9 AØ       658           CMP  #$AØ       ;weiches Leer-
ØD42: FØ EF       659           BEQ  INCREM     ;zeichen
                  66Ø  *
                  661  *
                  662  * Zeichen umwandeln, CTRL ausfiltern
                  663  *
```

```
ØD44: C9 1F       664  CONVERT  CMP  #$1F       ;weiche Trng.
ØD46: FØ 1A       665           BEQ  CONV1
ØD48: C9 2D       666           CMP  #$2D       ;harte Trng.
ØD4A: FØ 24       667           BEQ  CONV2
ØD4C: C9 ØD       668           CMP  #$ØD       ;hartes Return
ØD4E: FØ 3Ø       669           BEQ  CONV3
ØD5Ø: C9 2Ø       67Ø           CMP  #$2Ø       ;hartes Leerz.
ØD52: FØ Ø4       671           BEQ  LEERZ
ØD54: C9 8D       672           CMP  #$8D       ;weiches Return
ØD56: DØ Ø4       673           BNE  CONTROL
ØD58: A9 AØ       674  LEERZ    LDA  #$AØ       ;Leerzeichen
ØD5A: DØ C8       675           BNE  AUSGABE1   ;ausgeben
ØD5C: C9 2Ø       676  CONTROL  CMP  #$2Ø
ØD5E: 9Ø D3       677           BLT  INCREM     ;ausfiltern
ØD6Ø: BØ CØ       678           BGE  AUSGABEØ   ;unverändert
                  679  *                        ;senden
                  68Ø  *
                  681  * Behandlung $1F weiche Trennung
                  682  *
ØD62: AØ Ø1       683  CONV1    LDY  #1         ;nächstes Zei-
ØD64: B1 Ø4       684           LDA  (PUFFER),Y ;chen lesen
                  685  * wenn kein CR folgt überspringen
ØD66: C9 8D       686           CMP  #$8D
ØD68: DØ C9       687           BNE  INCREM
                  688  * sonst zusätzlich CR löschen
ØD6A: A9 AØ       689           LDA  #$AØ
ØD6C: 91 Ø4       69Ø           STA  (PUFFER),Y
ØD6E: DØ C3       691           BNE  INCREM
                  692  *
                  693  * Behandlung $2D harte Trennung
                  694  *
ØD7Ø: AØ Ø1       695  CONV2    LDY  #1         ;nächstes Zei-
ØD72: B1 Ø4       696           LDA  (PUFFER),Y ;chen lesen
                  697  * wenn kein CR folgt, unverändert ausgeben
ØD74: C9 8D       698           CMP  #$8D
ØD76: DØ Ø4       699           BNE  SENDEN
                  7ØØ  * sonst zusätzlich CR löschen
ØD78: A9 AØ       7Ø1           LDA  #$AØ
ØD7A: 91 Ø4       7Ø2           STA  (PUFFER),Y
ØD7C: A9 AD       7Ø3  SENDEN   LDA  #$AD
ØD7E: DØ A4       7Ø4           BNE  AUSGABE1
                  7Ø5  *
                  7Ø6  *
                  7Ø7  * Behandlung $ØD hartes Return
                  7Ø8  * Suche nach Punktbefehlen im Text
                  7Ø9  *
ØD8Ø: A9 8D       71Ø  CONV3    LDA  #$8D
ØD82: 2Ø ED FD    711           JSR  COUT
ØD85: AØ Ø2       712  SUCHE    LDY  #2         ;übernächstes
ØD87: B1 Ø4       713           LDA  (PUFFER),Y ;Zeichen
ØD89: C9 2E       714           CMP  #$2E       ;Punkt
ØD8B: DØ A6       715           BNE  INCREM
                  716  * Punktbefehle überspringen
ØD8D: AØ ØØ       717           LDY  #Ø
ØD8F: E6 Ø4       718  SCHLEIF1 INC  PUFFER
ØD91: DØ Ø2       719           BNE  WEITER2
ØD93: E6 Ø5       72Ø           INC  PUFFER+1
ØD95: B1 Ø4       721  WEITER2  LDA  (PUFFER),Y
ØD97: C9 ØD       722           CMP  #$ØD       ;hartes Return
ØD99: DØ F4       723           BNE  SCHLEIF1
ØD9B: FØ E8       724           BEQ  SUCHE
                  725  *
                  726  *
                  727  * Speicher für Filenamen und Blockliste
                  728  *
                  729  OUTFILE  DS   3Ø         ;Outputfile
                  73Ø  INFILE   DS   11         ;Inputfile
                  731  BLOCKMAP DS   $7F        ;Blockliste
ØE45: EA          732           NOP

16Ø3 bytes
```

# GETCPM: Konvertierung von CP/M- in DOS-Textfiles

## Applesoft-Version

**von Thomas Fink**

Dieses Programm (s. S. 36) ermöglicht es, Textfiles von CP/M nach DOS zu übertragen. Im Gegensatz zu dem vorangehenden Programm ist es – mit Ausnahme der unentbehrlichen RWTS – vollständig in Applesoft geschrieben.

Die Bedienung ist denkbar einfach: Man legt nach dem Start von GETCPM (von einer DOS-Diskette) die CP/M-Diskette in Drive 1 und die DOS-Diskette in Drive 2, drückt SPACE und erhält das CP/M-Directory, aus dem man sich eine Datei per Nummer auswählen kann. Die Datenübertragung erfolgt dann automatisch.

### Anmerkungen zu den Programmzeilen

200 *Initialisieren*
210 Die beiden Buffer für Directory und Daten werden reserviert.
250 Felder für File-Name, Ort, Extension-Nr., Position und Blockzähler.
1000 *Prompt*
2000 *Lesen und Ausdrucken des Directory*
2010 Das Inhaltsverzeichnis befindet sich auf den Blocks 0 und 1.
2040 Es gibt maximal 64 File-Einträge mit jeweils 32 Bytes.
2060 Position der Blocknummernliste und Länge der Datei in Vielfachen von 128.
2070 Die ersten 11 Bytes enthalten den Namen des Eintrags.
2140 Die Extensionnummer besagt, die wievielte 16K-Erweiterung der Eintrag ist.
2140 Die Extensionnummer 0 kennzeichnet einen Hauptblock.
2160 Dieser wird vermerkt, numeriert und ausgedruckt.
3000 *Kopieren*
3010 Der Benutzer wird nach der Nummer des gewünschten Files gefragt.
3030 Unter dem gleichen CPM-Namen wird eine DOS-Textdatei eröffnet.
3050 Einlesen der ersten Extension.
3060 Sie wird zeichenweise herausgeschrieben, Crtl-Zeichen, insbesondere das Ctrl-J hinter jeder Zeile, werden ignoriert.
3070 End of File Marker ist Ctrl-Z.
3090 Ist dieser Block nicht voll gewesen, dann ist die Datei hier zu Ende.
3110 Eine weitere Extension wird unter dem gleichen Namen gesucht und kopiert.
4000 *Einlesen eines max. 16K langen Teiles in den Puffer*
4020 Die Blocknummern befinden sich in den Bytes 16-31 des Directory-Eintrages.
4040 Jeder Block ist 1024 Bytes lang.
5000 *Einlesen eines Blocks*
5010 Parameter für die RWTS: Lesen, keine Volume-Nr. und Drive 1.
5020 Ein CPM-Block umfasst 4 DOS-Sektoren.
5040 Block Null befindet sich auf Track 3, Sektor 0.
6000 *Aufruf der RWTS*
6010 Es wird der Standard-IOB von DOS benutzt.
6020 Hier stehen Drive-, Volume-, Track- und Sektor-Nummern.
6060 Außerdem die Stelle, wohin ein 256-Byte-Sektor geladen wird.
6090 Die Routine, die die RWTS aufruft, wird angesprungen.
6100 Ggf. wird ein Fehlercode berechnet.
6120 Ein Fehler ist aufgetreten, seine Meldung wird ausgedruckt.
6200 Poken der Routine, die die RWTS aufruft.
6280 Die Sektoren haben unter CPM eine andere Numerierung.
6320 Die drei möglichen Fehler werden initialisiert.

### 1-Drive-Version

Das Programm läuft auch mit einem einzigen Laufwerk, wenn man zusätzlich die Zeilen 3015, 4005 und 4065 mit einer entsprechenden Aufforderung, die Diskette zu wechseln, einfügt.

# Die Mas
# und Mec
# entblöß

## Der Macintosh im Leistungstest

von Ulrich Stiehl

Ein Bild sagt mehr als viele Worte (Mac-Philosophie)

# ke fällt
# ki seh' ich nun,
# von allen Reizen

(frei nach Goethe, Torquato Tasso)

## 1. ÄSTHETIK UND EFFIZIENZ

Warum gefällt der Macintosh vielen Mikrocomputer-Neulingen auf den *ersten* Blick? Weil er hinsichtlich der Ästhetik bzw. des sog. „Visual Interface" (Menü-Optik) vom herkömmlichen Mikrocomputer völlig abweicht. Wo findet man sonst einen solch kleinen und niedlichen Computer mit einer solch gefälligen Bildschirmschrift und einem solch ansprechenden Design? Ästhetische Charakteristika wie klein, niedlich, gefällig und ansprechend sind jedoch nur äußerliche Attribute, die aber Neulingen die Angst vor der EDV nehmen können und insofern von psychologischer Relevanz sind.

Wie steht es dagegen mit den Leistungsmerkmalen, die naturgemäß erst auf den *zweiten* Blick zum Vorschein kommen? Ist der Mecki dann immer noch der kleine Tausendsassa, der Großes leistet, wie in der Apple-Werbung versprochen wird? Um dies herauszufinden, haben wir zwei Software-Produkte unter die Lupe genommen und mit analogen Programmen für den Apple IIe verglichen:

– Programmiersprache: Microsoft Basic für den Macintosh versus Applesoft-Basic für den Apple IIe (Applesoft)
– Betriebssystem: Mac-DOS (SYSTEM/FINDER) für dem Macintosh versus DOS 3.3/ProDOS für den Apple IIe

In beiden Fällen ging es nicht um die Ästhetik, sondern um die Effizienz der Programme. Dabei kamen Tatsachen ans Tageslicht, die den äußerlich süßen und niedlichen Mecki als einen innerlich schwachen und leistungsarmen Computer erwiesen. Natürlich ist der Macintosh leistungsfähiger als beispielsweise der Sinclair ZX81, der aber auch erstens keinen MC68000-Prozessor hat und zweitens nur DM 99,- kostet.

Um den Unterschied zwischen Programm-Ästhetik und Programm-Effizienz zu verdeutlichen, zitieren wir als einfache Prozedur das Kopieren von Disketten (volle 400K-Diskette, 1 Macintosh-Drive, Macintosh-FINDER-Utility):
– Bei einem traditionellen Mikrocomputer erscheint ein nüchternes Menü in der sattsam bekannten und aus typografischer Sicht wenig ansprechenden Bildschirmschrift. Dafür dürfte allerdings der Kopiervorgang selbst in höchstens 3 Minuten abgeschlossen sein. Die Ästhetik ist hier also gering, dafür die Effizienz (Kopiergeschwindigkeit) hoch.
– Beim Macintosh erscheinen anstelle des konventionellen Menüs nette und typografisch ansprechend beschriftete Diskettenbildchen am Monitor, die man mit der Maus spielerisch bewegen kann. Dafür dauert allerdings der Kopiervorgang 13 Minuten. Die Effizienz ist hier also gering, dafür die Ästhetik groß.

Aufgrund der Menü-Optik ist die Eingewöhnungsphase beim Macintosh aus psychologischer Sicht für EDV-Greenhorns kürzer, aufmunternder und angenehmer als bei einem konventionellen Mikrocomputer. Man bleibt jedoch nur für einige Wochen Anfänger, und nach dem Vertrautsein mit dem neuen Gerät tritt das Merkmal der Ästhetik hinter das der Effizienz zurück. Beispielsweise wäre es einer Büroangestellten nunmehr sicherlich lieber, wenn sie abends nicht 1 Stunde, sondern nur 20 Minuten für die Datensicherung benötigte, auch wenn sie dabei auf die zunächst lustig anmutenden Bildchen der Disketten verzichten müßte. Doch hat sie jetzt keine Wahl mehr. Auch wenn sie die Bildchen inzwischen nicht mehr sehen kann, die allabendliche Überstunde ist nunmehr vorprogrammiert, weil ohne Bildchen und Maus bei dem gegenwärtigen Macintosh-Betriebssystem gar nichts geht.

### 68000 und 65C02C

Will man den Macintosh mit dem Apple II vergleichen, so sind die technischen Gegebenheiten zu berücksichtigen. Der normale Apple II/IIe/IIc hat einen 6502- bzw. 65C02-Prozessor, der mit 1 MHz getaktet ist. Der Macintosh ist demgegenüber mit einem MC68000-Prozessor bestückt, der laut Firma Apple eine Taktfrequenz von 8 MHz hat („Macintosh Benutzerhandbuch", S. 157, sowie entsprechende Werbemittel). Wir gehen einmal davon aus, daß dies zutrifft, denn eine objektiv unrichtige Angabe wäre ein Verstoß gegen Paragraph 3 UWG. Unsere Skepsis bezüglich der 8-MHz-Angabe ist darin begründet, daß der Macintosh bei unseren Tests so ungewöhnlich schlecht abschneidet.
Der 6502 ist ein 8-Bit- und der 68000 ein 16-Bit-Prozessor. Allein diese Tatsache bewirkt eine massive Überlegenheit des 68000 gegenüber dem 6502. Daher wäre es unfair, wenn man den 1-MHz-6502 mit dem 8-MHz-68000 vergleichen würde. Wir haben infolgedessen für Testzwecke einen Apple IIe mit 3,5-MHz-65C02C herangezogen (siehe „Peeker", Heft 1/84). Jeder Assembler-Programmierer würde annehmen, daß der 3,5-MHz-65C02C theoretisch nur etwa halb so schnell wie der 8-MHz-68000 ist. Deshalb waren wir höchst überrascht festzustellen, daß praktisch genau das Gegenteil der Fall ist. Und das, obwohl die Minimalausstattung des Apple IIe mit 65C02C halb so teuer wie die des Macintosh ist. Man bekommt also für den doppelten Geldbetrag nur die Hälfte der Leistung, d.h. die Mark ist dann nur noch 25 Pfennig wert.

## 2. MICROSOFT BASIC FÜR MACINTOSH UND APPLE IIE

Vielen ist unbekannt, daß der alte Applesoft-Interpreter für den Apple II in Wirklichkeit von der Firma Microsoft stammt und von der Firma Apple lediglich um die Hires-Routinen erweitert wurde. Deshalb bieten sich das Microsoft Basic für den Apple (kurz Apple-MS) sowie das Microsoft Basic für den Macintosh (kurz Mac-MS) bestens für einen Vergleich der einzelnen Leistungsmerkmale an.

Der normale Apple IIe verfügt über 64K RAM sowie einen 10K großen Applesoft-Interpreter im ROM (mit Monitor insgesamt 12K ROM). Die Speicherkapazität für das Basic-Programm beträgt ohne Tricks ca. 35K und mit Tricks (DOS 3.3 in der Language Card) ca. 45K. Es sind mehrere Compiler erhältlich, u.a. von Microsoft selbst (TASC).

Der normale Macintosh verfügt über 128K RAM und 64K ROM, und der Basic-Interpreter nimmt ca. 45K im RAM ein. Trotzdem beträgt die Speicherkapazität für das Basic-Programm ohne Tricks nur ca. 14K und mit Tricks (CLEAR, 29000, 1024) ca. 29K. Es fällt auf, daß trotz des doppelt so großen RAM-Bereichs der freie Arbeitsspeicher nur etwa halb so groß wie beim Apple IIe ist. Deshalb dürfte wohl auch kein Compiler entwickelt werden, denn sonst wäre die Speicherkapazität vollends erschöpft.

Während der geringe freie Speicherraum beim Macintosh-Basic schon eine unerfreuliche Überraschung ist, so ist man erst recht verwundert, wenn man Geschwindigkeitstests anstellt:

| | | |
|---|---|---|
| a) Disk-Test: | Mac 156s, | Apple 151s |
| b) String-Test: | Mac 127s, | Apple 51s |
| c) Mathe-Test: | Mac 174s, | Apple 49s |
| d) Screen-Test: | Mac 246s, | Apple 67s |
| e) Grafik-Test: | Mac 128s, | Apple 45s |
| f) Find-Test: | Mac 71s, | Apple 76s |

### a) Disk-Test

Beim Disk-Test wird ein ca. 103K umfassender, sequentieller Textfile auf Diskette geschrieben und danach wieder komplett eingelesen. Der Apple-MS-Test lief – wie auch die übrigen Tests – unter ProDOS. Die Zeiten in Sekunden:
a) Apple-MS bei 1 MHz: 230s
b) Apple-MS bei 3,5 MHz: 151s
c) Mac-MS bei 8 MHz: 156s

Es wurden auch noch andere Disketten-Lese/Schreib-Tests angestellt, die zeigten, daß das Macintosh-DOS nicht schneller als das Apple-ProDOS ist. Hier halten sich also Apple und Macintosh die Waage, obgleich man angesichts des MC68000 beim Macintosh eine höhere Datenübertragungsrate hätte erwarten können.

```
100 REM DISK.TEST.Apple-MS: 151s
110 PRINT CHR$ (4) "OPEN TEST-
    DATEI":
    PRINT CHR$ (4) "WRITE TEST-
    DATEI"
120 X$ = "_123456789"
130 FOR X = 1000 TO 8000
140 PRINT X; X$
150 NEXT
160 PRINT CHR$ (4) "CLOSE"
170 PRINT CHR$ (4) "OPEN TEST-
    DATEI":
    PRINT CHR$ (4) "READ TEST-
    DATEI"
180 FOR X = 1000 TO 8000
190 INPUT X$
200 NEXT
210 PRINT CHR$ (4) "CLOSE"
220 PRINT CHR$ (7)
```

```
100 REM DISK.TEST.Mac-MS: 156s
110 OPEN "O", #1, "TESTDATEI"
120 X$ = "_123456789"
130 FOR X = 1000 TO 8000
140 PRINT #1, X; X$
150 NEXT
160 CLOSE #1
170 OPEN "I", #1, "TESTDATEI"
180 FOR X = 1000 TO 8000
190 INPUT #1, X$
200 NEXT
210 CLOSE #1
220 BEEP
```

### b) String-Test

Dieser String-Test führt 8000 String-Manipulationen unterschiedlicher Art durch. Der Apple ist hier bereits 2,5mal schneller als der Macintosh:
a) Apple-MS mit 1 MHz: 167s
b) Apple-MS mit 3,5 MHz: 51s
c) Mac-MS mit 8 MHz: 127s

```
100 REM STRING.TEST.Apple-MS: 51s
110 X$ = "1234567890"
120 FOR X = 1 TO 8000
130 Y = LEN (X$)
140 Y$ = LEFT$ (X$,1)
150 Y$ = RIGHT$ (X$,1)
160 Y$ = MID$ (X$,5,1)
170 Y = ASC (X$)
180 Y$ = CHR$ (Y)
190 NEXT
200 PRINT CHR$ (7)
```

```
100 REM STRING.TEST.Mac-MS: 127s
110 X$ = "1234567890"
120 FOR X = 1 TO 8000
130 Y! = LEN (X$)
140 Y$ = LEFT$ (X$,1)
150 Y$ = RIGHT$ (X$,1)
160 Y$ = MID$ (X$,5,1)
170 Y! = ASC (X$)
180 Y$ = CHR$ (Y!)
190 NEXT
200 BEEP
```

### c) Mathe-Test

Beim Mathematik-Test werden zum einen Grundrechenarten und zum anderen Transzendentalfunktionen geprüft. Das Ergebnis ist für den Macintosh enttäuschend: Der Apple ist 3,5mal schneller. Hierzu muß man wissen, daß beim Apple-MS die Fließkommazahlen intern mit 5 Bytes gespeichert werden, während beim Mac-MS die Single-Precision 4 Bytes (!-Variable) und die Double-Precision 8 Bytes (#-Variable) einnimmt. Die Transzendentalfunktionen können nur mit doppelter Genauigkeit ausgeführt werden. Zur Kompensation wurden die Grundrechenarten mit einfacher Genauigkeit vorgenommen. Die frustrierenden Ergebnisse im einzelnen:
a) Apple-MS mit 1 MHz: 163s
b) Apple-MS mit 3,5 MHz: 49s
c) Mac-MS mit 8 MHz: 174s
Für das schwache Abschneiden gibt es nur 3 mögliche Gründe: Entweder ist die Angabe von Apple, daß der Mac mit 8 MHz arbeitet, falsch (wahrscheinlich), oder das Mac-MS ist völlig umständlich programmiert (unwahrscheinich) oder der MC68000 macht während des Programmablaufs noch andere nutzlose Dinge (teils erwiesen). So konnte beispielsweise mit einem „Rührtest" gezeigt werden, daß Mac-MS-Programme bis zu 10% langsamer laufen. Zu diesem Zweck wurde während des Mathe-Tests der Maus-Zeiger permanent im Kreis „gerührt".

```
100 REM MATHE.TEST.Apple-MS: 49s
110 A = 123.456: B = 12.3456
120 D = 1234.56789
130 FOR X = 1 TO 1000
140 C = A * B: C = A / B:
    C = A + B: C = A - B
150 E = SIN (D): E = COS (D):
    E = TAN (D): E = SQR (D)
160 NEXT
170 PRINT CHR$ (7)
```

```
100 REM MATHE.TEST.Mac-MS: 174s
110 A! = 123.456: B! = 12.3456
120 D# = 1234.56789
130 FOR X = 1 TO 1000
140 C! = A! * B!: C! = A! / B!:
    C! = A! + B!: C! = A! - B!
150 E# = SIN (D#): E# = COS (D#):
    E# = TAN (D#): E# = SQR (D#)
160 NEXT
170 BEEP
```

**d) Screen-Test**

Mit dem Screen-Test sollte die Bildschirm-Scrollgeschwindigkeit unter Beweis gestellt werden. Beim Apple IIe wurde zu diesem Zweck die 80-Zeichenkarte eingeschaltet, deren Firmware-Routinen bekanntlich sehr langsam sind. Beim Macintosh kommt hier zum Tragen, daß eine Unterscheidung zwischen Text- und Grafik-Modus gar nicht mehr existiert und somit auch normale Texte stets als Bit-Map-Grafik dargestellt werden. Die Summe aller Pixel (512 Bildpunkte horizontal mal 342 Bildpunkte vertikal) nehmen ca. 21K des RAM ein, wodurch das schlechte Abschneiden des Macintosh plausibel wird:
a) Apple-MS mit 1 MHz: 94s
b) Apple-MS mit 3,5 MHz: 67s
c) Mac-MS mit 8 MHz: 246
Textverarbeitungsprogramme wie etwa Macwrite wissen um diese bedauerlichen Scrollwerte, die jeder flotten Schreibkraft das Leben versauern. Deshalb scrollt Macwrite nicht seiten-, sondern nur zeilenweise und frischt von Zeit zu Zeit den Bildschirm wieder ganz auf. Wenn man bei einem größeren Macwrite-Text über die Funktion Suchen/Ersetzen alle Leertasten und womöglich auch noch alle Returns entfernt, friert der Bildschirm förmlich ein, weil es etwa 3 Sekunden dauert, bis nach einem einzigen Tastendruck eine Reaktion am Monitor erfolgt.
Übrigens bringt eine Reduzierung der Fensterbreite kaum eine Verbesserung der Scrollgeschwindigkeit, wohl aber eine Reduzierung der Fenstertiefe.

```
100 REM SCREEN.TEST.Apple-MS: 67s
110 PRINT CHR$ (4) "PR#3"
120 FOR X = 1000 TO 2000
130 PRINT X; "_12345678901234567890
    12345678901234567890123456789012345678901234567890"
140 NEXT
150 PRINT CHR$ (7)
```

```
100 REM SCREEN.TEST.Mac-MS: 246s
    (mit vollem Fenster!)
110 CLS
120 FOR X = 1000 TO 2000
130 PRINT X; "_12345678901234567890
    12345678901234567890123456789012345678901234567890"
140 NEXT
150 BEEP
```

**e) Grafik.Test**

Die Grafik-Fähigkeiten des Macintosh sind überlegen, weil in die 64K des ROM-Speichers eine Fülle von Spezialroutinen gepackt wurden. Hätte der Apple IIe ähnlich umfangreiche ROM-Routinen, so würde er dem Macintosh nicht nachstehen. Im Gegenteil! Wie der nachfolgende Grafik-Test zeigt, ist der Apple bei Einzel-Bildpunkt-Routinen dem Macintosh sogar um den Faktor 2,8 überlegen:
a) Apple-MS mit 1 MHz: 149s
b) Apple-MS mit 3,5 MHz: 45s
c) Mac-MS mit 8 MHz: 128s
Das Grafik-Test-Beispiel plottet ein Rechteck. Hierfür gibt es bereits entsprechende Mac-MS-Befehle, die nur einen Sekundenbruchteil benötigen würden. Angemessener wäre das Plotten einer mathematischen Funktionskurve gewesen, bei der jedoch die ungünstigen Mathe-Rechenzeiten das Mac-Ergebnis noch schwächer hätten erscheinen lassen, als es in Wirklichkeit ist.

```
100 REM GRAFIK.TEST.Apple-MS: 45s
110 HGR : HCOLOR= 7
120 FOR X = 0 TO 250
130 FOR Y = 0 TO 150
140 HPLOT X, Y
150 NEXT Y, X
160 PRINT CHR$ (7)
```

```
100 REM GRAFIK.TEST.Mac-MS: 128
110 CLS
120 FOR X! = 0 TO 250
130 FOR Y! = 0 TO 150
140 PSET (X!, Y!)
150 NEXT Y!, X!
160 BEEP
```

**f) Find.Test**

Der Find-Test (Such-Test) ist ein typischer Compilertest, denn es werden die Speicherstellen von (a) Variablen und (b) Programmzeilen gesucht. Dies ist der einzige Test, bei dem das Mac-MS etwas günstiger als das Apple-MS abschneidet. Würde man jedoch das Applesoft-Programm mit dem TASC compilieren, so würde die ursprüngliche Ausführungszeit von 76s auf 1s zusammenschrumpfen.
a) Apple-MS mit 1 MHz: 251s
b) Apple-MS mit 3,5 MHz: 76s
c) Mac-MS mit 8 MHz: 71s

```
0 REM FIND.TEST.Apple-MS: 76s
1 GOSUB 100
2 FOR X = 1 TO 3000
3 Y = ZZ
4 GOSUB 800
5 NEXT
6 PRINT CHR$ (7)
7 END
100 AA% = 1
101 AB% = 1
102 AC% = 1
103 AD% = 1
104 AE% = 1
105 AF% = 1
106 AG% = 1
107 REM usw. Zeilen 107-764
765 ZP% = 1
766 ZQ% = 1
767 ZR% = 1
768 ZS% = 1
769 ZT% = 1
770 ZU% = 1
771 ZV% = 1
772 ZW% = 1
773 ZX% = 1
774 ZY% = 1
775 ZZ = 1
800 RETURN
```

```
0 REM FIND.TEST.Mac-MS: 71s
1 GOSUB 100
2 FOR X = 1 TO 3000
3 Y! = ZZ!
4 GOSUB 800
5 NEXT
6 BEEP
7 END
100 AA% = 1
101 AB% = 1
102 AC% = 1
103 AD% = 1
104 REM usw. Zeilen 104-771
772 ZW% = 1
773 ZX% = 1
774 ZY% = 1
775 ZZ! = 1
800 RETURN
```

Um die Suchgeschwindigkeit an einem Anwenderprogramm zu verifizieren, wurde die Suchfunktion beim Applewriter IIe getestet. Zu diesem Zweck wurden zunächst nur 5 Buchstaben mit Return eingegeben:
A
A
A
A
A
Danach wurde mit der automatischen Suchen/Ersetzen-Funktion der Speicher mit

der Folge
/A/BBBBB /A
/B/CCCCC /A
/C/DDDDD /A
/D/EEEEE /A
/E/FFFFF /A
auf insgesamt 19535 Zeichen gefüllt
(1: Gesamtzeit 70s).
Nunmehr wurde mit
/FFFFF/FFF/A
der Speicher zunächst auf 13285 Zeichen geschrumpft (2: 34s) und dann mit
/FFF/FFFFFF/A
auf 22660 Zeichen anwachsen lassen
(3: 53s).

Die entsprechenden Ausführungszeiten für das Macwrite-Programm (1: 289s; 2: 272s; 3: 197s mit Abbruch, weil nicht mehr als ca. 20K verarbeitet werden können) waren enttäuschend und zeigen, daß das Macwrite erheblich langsamer als der Applewriter ist.

### Fazit

Der Macintosh mit 8-MHz-68000 ist im Durchschnitt etwa doppelt so langsam wie der Apple mit 3,5-MHz-65C02C. Berücksichtigt man den Preis für beide Geräte, so kann man konstatieren, daß 2 Apple-Geräte etwas das 4fache eines einzigen Macintosh-Gerätes leisten. Anders formuliert: Für den halben Preis bekommt man mit dem Apple IIe bereits die doppelte Geschwindigkeit des Macintosh. Wer also auf Geschwindigkeit Wert legt, sollte sich in keinem Fall einen Macintosh zulegen, sondern beispielsweise einen Apple IIe mit 65C02C-Accelerator-Karte kaufen. Und wem dies immer noch nicht reicht, erwerbe einen IBM-PC oder einen anderen leistungsfähigen Mikrocomputer.

## 3. MAC-DOS UND APPLE-DOS

Das Diskettenbetriebssystem des Macintosh hat keinen eigenen Namen. Wir nennen es daher kurz Mac-DOS. Programmtechnisch besteht es aus Teilen des 64K-ROM-Inhalts sowie aus den von der Mac-Systemdiskette eingeladenen Programmen SYSTEM und FINDER. Die Namen dieser Dateien lassen sich übrigens durch das Mac-DOS selbst nicht ändern, wohl aber beispielsweise aus dem Mac-MS heraus. Danach ist die Diskette jedoch nicht mehr bootfähig, weil die Systemdateien nicht mehr gefunden werden.
Die Firma Apple macht zur Zeit noch keine technischen Daten über den Macintosh publik. Nur Software-Häuser, die einen sog. „Schweigevertrag" (Non-Disclosure Contract) unterzeichnet haben, wurden mit Vorabunterlagen versorgt. Da wir als Peeker-Redaktion ein solches „Stillhalteabkommen" nie unterzeichnen würden, denn sonst wäre eine objektive Berichterstattung nicht mehr möglich, gehören wir auch nicht zum Kreis der „Privilegierten". Daher basieren die nachfolgenden Angaben auf eigenen Untersuchungen.

### Diskettenformat

Eine 3,5-Zoll-Mac-DOS-Diskette (Fabrikat Sony, einseitig, doppelte Dichte) hat einen Bruttokapazität von 400K und umfaßt 80 Spuren. Die Anzahl der Blocks (= 512 Bytes = 1/2K) variiert von Spur zu Spur. Die ersten (äußersten) 16 Spuren (Spur 0-15) enthalten jeweils 12 Blocks, die nächsten 16 Spuren (Spur 16-31) jeweils 11 Blocks, die nächsten 16 Spuren (Spur 32-47) 10 Blocks, die nächsten 16 Spuren (Spur 48-63) jeweils 9 Blocks und schließlich die letzten (innersten) 16 Spuren (Spur 64-79) jeweils 8 Blocks. Dies ergibt somit 16 * 12 + 16 * 11 + 16 * 10 + 16 * 9 + 16 * 8 = 800 Blocks. Die Rotationsgeschwindigkeit hängt von der Anzahl der Blocks der jeweiligen Spur ab. Wie der obige Mac-MS-Disk-Test gezeigt hat, entspricht die Datenübertragungsrate des Mac-DOS etwa der des Apple-ProDOS.
Blocks 0 und 1 enthalten den Urlader. Danach folgt das Disketteninhaltsverzeichnis (Catalog, Directory). Im Gegensatz zu ProDOS gibt es keine hierarchische Directory-Struktur. Eine Diskette kann nur insgesamt 107 Dateinamen umfassen, wobei die sog. Schreibtisch- oder Desktop-Datei, d.h. das auf dem Bildschirm als Datei-Ikonen formatierte Inhaltsverzeichnis selbst nicht als Dateieintrag ersichtlich ist. Mac-MS zeigt jedoch bei dem Catalog-Befehl namens FILES diese Desktop-Datei mit an. Ein Diskettenname darf bis zu 27 Zeichen, ein Dateiname bis zu 67 Zeichen umfassen, wobei der Doppelpunkt als Zeichen nicht erlaubt ist, weil er als Separator zwischen Disketten- und Dateiname dient (DISKVOLUME-:FILENAME). Mac-MS läßt übrigens längere Dateinamen zu, die dann allerdings nicht mehr vom Mac-FINDER verändert werden können. Ferner sei vermerkt, daß der Doppelpunkt-Separator beim FINDER selbst nicht zum Tragen kommt.
Wenn man eine Macintosh-Systemdiskette bootet, wird zunächst der Urlader, dann die Datei SYSTEM und schließlich die Datei FINDER geladen. Der Mac-FINDER entspricht funktionell etwa dem ProDOS-FILER. Es handelt sich also um ein Disketten- und Datei-Kopierprogramm. Nach dem Laden des FINDER s zeigt dieser zunächst das Bildchen der gebooteten Diskette an (stilisierte Sony-Diskette mit darunter gesetztem Diskettennamen). Es ergeben sich nunmehr folgende FINDER-Funktionen:

#### a) Anzeige des Directory

Während man beim Apple-DOS usw. den Befehl CATALOG eintippt, muß man beim Mac-DOS mit der Maus auf die Disketten-Ikone zeigen und zweimal „klicken" (dada klickiklick). Danach kann man aus der Menü-Leiste „Inhalt" folgende Darstellungsformen des Disketteninhaltsverzeichnisses auswählen: (a) nach Abbild (Datei-Ikonen; „wirre" Reihenfolge) sowie (b) nach Namen, Datum, Größe oder Art (4 mögliche sortierte Reihenfolgen). Die „wirre" Datei-Ikonen-Darstellungsform kann man sich durch Verrücken der Ikonen innerhalb des Darstellungsfensters selbst gestalten sowie über die Funktion „Aufräumen" („Clean up" der Menü-Leiste „Spezial") automatisch neu anordnen lassen. „Wirr" besagt, daß man bei einer Diskette, auf der wiederholt Dateien gelöscht und gespeichert wurden, nicht mehr automatisch diejenige Ikonen-Reihenfolge erhält, die der physischen Reihenfolge der Dateinamen im Directory entspricht.
Im übrigen gibt es noch eine weitere Option „Informationen" aus der Menü-Leiste „Ablage", die u.a. die Dateigröße in Bytes einer einmal geklickten (dada klick) Datei-Ikone anzeigt.

#### b) Gruppierung von Dateinamen

Unter ProDOS lassen sich sog. Subdirectories oder Diskettenunterinhaltsverzeichnisse bilden. Beim Mac-DOS gibt es nur sog. Pseudo-Subdirectories, d.h. ästhetisch in einer „Ordner"-Ikone („Aktentasche") zusammengefaßte Dateinamen, denen jedoch kein physisches Subdirectory auf der Diskette entspricht. Das Zusammenfassen von Datei-Ikonen in einer gemeinsamen Ordner-Ikone dient erstens der Menü-Optik und zweitens dem Komfort des Verrückens von Ikonen-Gruppen beim Kopieren und Löschen von Dateien.

#### c) Löschen von Disketten (Formatieren)

Während man unter Apple-DOS eine Diskette durch den Befehl „INIT HELLO" o.ä. löscht, nachdem man zuvor die entsprechende Diskette in das Laufwerk eingelegt hat, kann der Mac-FINDER eine Diskette

nur dann löschen (entweder erstmals formatieren oder neu formatieren), wenn entweder eine fabrikneue Leerdiskette in das Laufwerk gesteckt wird oder wenn die Disketten-Ikone einer bereits früher formatierten Diskette auf dem Bildschirm („Schreibtisch") sichtbar ist. Ein „Blind-Formatieren" ist nicht möglich, weil das Mac-Drive im Gegensatz zu den normalen Sony-Laufwerken keinen Knopf zum Auswerfen von Disketten hat. Deshalb kann eine Diskette mit vollem Inhaltsverzeichnis (107* Dateinamen einschließlich der unsichtbaren „Schreibtisch"-Datei) nicht mehr neu formatiert werden (siehe unten). Sinngemäß dürften auch partiell zerstörte oder kopiergeschützte Disketten in bestimmten Fällen nicht mehr neu formatierbar sein.

*Theoretisch 109, s. unten

Man beachte, daß die Option „Löschen" aus der Menü-Leiste „Bearbeiten" *nicht* zum Löschen „geklickter" Datei-Ikonen dient. Analoges gilt für die Option „Kopieren" aus derselben Menü-Leiste. Sowohl „Löschen" als auch „Kopieren" beziehen sich hier auf das Editieren von Textstellen und nicht auf Disketten oder Dateien.

Das „Auswerfen" einer Diskette ist beim Mac-DOS eine spezielle Option aus der Menü-Leiste „Ablage". Kopiergeschützte Disketten lassen sich oft nicht mehr auswerfen, z.B. die „Guided Tour of Macintosh", wenn bei dieser Diskette der rote Schreibschutzknopf nach oben geschoben wird. In diesem Fall kann man sich helfen, indem man einen kleinen Nagel in das Loch schlägt, das rechts neben dem Laufwerkschlitz für diesen Brachialauswurf bereits vorgesehen ist. Man beachte ferner, daß im Gegensatz zu den Apple-Disketten die Mac-Disketten, wenn sie in das Laufwerk eingeführt werden, automatisch zunächst vom Mac-DOS gelesen werden, wobei entweder das ganze Directory oder zumindest der Diskettenname in den RAM-Speicher übernommen wird. Dadurch wird ein Verwechseln von Disketten insbesondere bei 1-Drive-Benutzern vermieden.

### d) Löschen von Dateien

Während man unter Apple-DOS eine Datei mit dem Befehl DELETE DATEINAME o.ä. löscht, wird das Löschen einer Datei beim Mac-DOS dadurch veranlaßt, daß die Ikone der Datei auf der Ikone des „Papierkorbs" plaziert wird und nachträglich über die Option „Papierkorb entleeren" aus der Menü-Leiste „Spezial" der „Papierkorb" de facto geleert wird. Das Löschen einer Gruppe von Dateien kann global geschehen, wenn diese Dateien zuvor einer „Ordner"-Ikone subsumiert worden sind. Der sog. „Leere Ordner" symbolisiert übrigens keine physisch existente Datei, sondern dient nur als „Duplizier-Aktentasche" für weitere „echte" Aktentaschen.

### e) Duplizieren von Disketten

Während unter Apple-DOS das Duplizieren von Disketten mit speziellen Kopierprogrammen vorgenommen wird (COPYA, QUICKCOPY aus „Peeker", Heft 1/85 usw.), die auch „Blind-Kopieren" erlauben, wird beim Mac-FINDER das Kopieren ganzer Disketten dadurch veranlaßt, daß die Ikone der Originaldiskette auf der Ikone der Duplikatdiskette plaziert wird. Im Gegensatz zu den üblichen Apple-DOS-Kopierprogrammen erzeugt das Duplizieren jedoch keine 1:1-Kopie, weil nicht Spur für Spur, sondern nur Datei für Datei kopiert wird, so daß die Kopierdauer von der Belegung der Originaldiskette abhängt. Die nicht-belegten Blocks werden nicht kopiert. Auch der Diskettenname wird nicht automatisch übernommen, sondern muß nachträglich manuell geändert werden.

### f) Duplizieren von Dateien

Während unter Apple-DOS das Duplizieren einzelner Dateien entweder manuell (LOAD BASICPROGRAMM, D1 ... SAVE BASICPROGRAMM, D2) oder mit Hilfe spezieller Dateikopierprogramme (FID, FILER) geschieht, wird unter Mac-DOS das Kopieren einzelner Dateien von der Originaldiskette auf die Duplikatdiskette dadurch veranlaßt, daß die Ikone der Originaldatei aus dem Ikonen-Inhaltsverzeichnis der Originaldiskette in das Ikonen-Inhaltsverzeichnis der Duplikatdiskette gerückt wird. Wenn entweder die Originaldiskette mehr als ca. 100 Dateinamen umfaßt oder wenn die Summe der Dateinamen von Original- und Duplikatdiskette ca. 120 überschreitet, ist das Kopieren von Dateien nicht mehr möglich (siehe unten).

Im übrigen besteht auch noch die Möglichkeit des Duplizierens einer Datei auf *dieselbe* Diskette durch die Option „Duplizieren" aus der Menü-Leiste „Ablage". Hierzu muß die Datei-Ikone nur einmal „geklickt" werden (dada klick). Alternativ ist es hier ausnahmsweise auch möglich, auf den *Namen* der Datei im namentlich gelisteten Directory einmal zu „klicken".

### g) Starten eines Programms

Während unter Apple-DOS ein Programm entweder automatisch (Hello-Programm der Bootdiskette) oder durch RUN PROGRAMMNAME o.ä. gestartet wird, wird unter Mac-DOS der Programmstart dadurch veranlaßt, daß die Ikone der Programmdatei zweimal „geklickt" wird (dada klickiklick). Alternativ ist es hier ausnahmsweise auch möglich, auf den *Namen* der Datei im namentlich gelisteten Directory zweimal zu „klicken" (dada klikkiklick).

### Systemfehler von Mac-DOS

Das Mac-DOS hat 3 schwerwiegende Nachteile:

– Alle wesentlichen DOS-Vorgänge kann man nur dann durchführen, wenn entweder die *Ikone* der Diskette(n) und/oder die *Ikonen* der Dateien sichtbar sind, so daß auf diese „geklickt" werden kann. Die *Namen* der Disketten und/oder Dateien *allein* genügen also in der Regel nicht. Dies bewirkt, daß sog. „Blind-Operationen" (Blind-Duplizieren, Blind-Formatieren) nicht mehr möglich sind. Solange die Kapazität des vom Mac-DOS belegten Arbeitsspeichers noch nicht erschöpft ist, ist dies belanglos. Danach folgt jedoch ein Systemabsturz dem anderen, der das Ende der Mecki-Gemütlichkeit bedeutet.

– Während das Apple-DOS das Directory einer Diskette je nach Bedarf sukzessive sektoren- oder blockweise in den Speicher einliest und sich darüber hinaus unter ProDOS höchsten die Volume-Namen selbst, aber niemals die Dateien aller Volumes merkt, ist bislang wenig bekannt, daß das Mac-DOS von *allen* Dateien *aller* aktiven Disketten eine umfangreiche RAM-Directory-Buchhaltung anzulegen versucht, die jedoch bei der 128K-RAM-Version des Macintosh nur für etwa 100-120 Dateinamen Platz hat und bei Erreichen dieser Obergrenze stets zum Systemabsturz führt, wenn ikonenabhängige Operationen durchgeführt werden.

– Bei DOS 3.3 konnten nur 105 Dateien auf der Diskette angelegt werden, während unter ProDOS beliebig viele Subdirectories mit beliebig vielen Dateien gebildet werden können, auch wenn das sog. Volume-Directory selbst nur 51 Dateinamen aufnehmen kann. Unter Mac-DOS können demgegenüber nur maximal 107 Dateien angelegt werden (106 eigentliche Dateien sowie 1 versteckte Desktop-Datei). Für eine 400K-Diskette ist dies zwar oft ausreichend (400 : 106 = ca. 4K als Datei-Durchschnittsgröße), für größere Datenträger (z.B. für die angekündigten 800K-Disketten sowie für Festplattenlauf-

werke aller Art) jedoch tödlich. Bei einer Harddisk mit 10M (= 10240K) müßte jede Datei bereits im Durchschnitt mindestens 10240 : 106 = 96K groß sein, was mehr als illusorisch ist. Die Folge ist, daß Harddisks nur partiell genutzt werden können. Noch gravierender ist jedoch der Umstand, daß das Mac-DOS nicht einmal 100 Dateien verwalten kann, ohne die Ikone der „Bombe" (ID = 25) auszulösen, die jetzt bei 100 Dateien so häufig erscheint wie früher die Ikone des freundlichen Mecki beim Booten von weitgehend leeren Systemdisketten. Und was am peinlichsten ist: Das Mac-DOS wird jetzt so schwerfällig, daß man eine panische Angst vor dem Bewegen von Ikonen bekommt, weil jetzt jede „Bilderschieberei" mit minutenlangem Schweigen, dem meist als Krönung ein „Bombenangriff" folgt, quittiert wird.

Daß man all dies bereits bei einer normalen 400K-Diskette erleben kann, zeigt der folgende Test. Zu diesem Zweck legten wir eine Diskette an, die zunächst nur folgende Dateien enthielt:
1. Schreibtisch (Desktop, unsichtbar)
2. Zwischenablage
3. Notizblockdatei
4. SYSTEM
5. FINDER
6. MS-BASIC

Hierzu kamen dann später noch folgende Dateien:
7. SAVER
8.-107. XXX00-XXX99

Dazu wurde über Mac-MS das folgende Programm namens „SAVER" gestartet, das sich selbst 100mal unter den Dateinamen XXX00, XXX01 ... XXX98, XXX99 abspeichert. Auf der Diskette waren dann übrigens noch 124K frei, womit gezeigt wird, daß man zumindest bei einer reinen Mac-MS-Diskette (ohne die großen Systemfiles) auch unabhängig von unserem gekünstelten Testprogramm unschwerlich 100 echte, kleinere Programme abspeichern könnte.

```
100 DEFINT A-Z
110 GOSUB 220: X = VARPTR (V)
120 X = X - 50
130 IF PEEK (X) = 88 AND PEEK (X + 1)
    = 88 AND PEEK (X + 2) = 88
    THEN 150: REM 88 = "x"-Buchstabe
140 X = X + 1: GOTO 130
150 X = X + 3
160 FOR Y = 0 TO 9
170 FOR Z = 0 TO 9
180 POKE X, Y + 48: POKE X + 1,
    Z + 48: REM ASCII-Ziffern 0-9
190 GOSUB 230
200 NEXT Z, Y
210 END
220 V = 1: RETURN
230 SAVE "XXX00": RETURN: REM "00"
    von "xxx00" wird gepokt
```

Mit dem Mac-MS-Catalog-Befehl FILES konnten wir uns davon überzeugen, daß alle Dateien XXX00 ... XXX99 korrekt auf der Diskette gespeichert waren. Es muß an dieser Stelle hinzugefügt werden, daß dieser Test nicht in dieser linearen Form entstand. Vielmehr gingen frustrierende, stundenlange Versuche voraus, bei deren Gelegenheit 2 Disketten zerstört wurden. Hinzu kommt, daß erst einmal experimentell erwiesen werden mußte, daß das Mac-Directory nur 107 Dateinamen aufnehmen kann.

– Wenn man das Mac-MS verlassen will, gibt man den Befehl SYSTEM ein, der die gleichnamige Datei sowie darauf den FINDER neu einlädt. Was meinen Sie nun, wann nach dem Verlassen von Mac-MS der gewohnte „freundliche Schreibtisch" wieder am Bildschirm erschien? Nach exakt 508 Sekunden, sprich 8 Minuten und 28 Sekunden! Davon vergingen 20 Sekunden auf das Einlesen des Directory und die restlichen 8 Minuten wurden für das Sortieren und Anzeigen der Datei-Ikonen „verbraten". Kein Wunder, daß wir bei den anfänglichen Experimenten 2 Disketten zerstörten, denn wir dachten zunächst verfrüht, daß sich Klein-Mecki bereits „aufgehängt" hätte.
– Da die Datei-Ikonen etwas wirr angeordnet waren, rückten wir einige Ikonen gerade und wählten dann die Menü-Funktion „Aufräumen" (Clean up). Amerikaner würden den Aufräum-Vorgang mit „Little Mekky is taken to the cleaners" (Slang für „Klein-Mecki wird kaltgemacht") umschreiben. Denn Sie werden es nicht glauben: Es dauerte jetzt 260 Sekunden, d.h. 4 Minuten und 20 Sekunden, bis Klein-Mekki „aufgeräumt" hatte. Nebenbei sei erwähnt, daß der Menü-Wechsel vom Abbild-Directory zum Namen-Directory jetzt 25 Sekunden dauerte – eine für die Firma Apple offenbar angemessene Zeit für das Sortieren von 100 Namen. (Zum Vergleich: Ein Assembler-Quicksort-Progamm benötigt 1-2 Sekunde für 1000 Namen.)
– Und was glauben Sie, was Sie sehen, wenn Sie diese Diskette auswerfen wollen? Natürlich die „Bombe", die Sie von nun ab bei dieser Diskette nicht mehr verläßt.

Nach zeitraubenden Experimenten, die von permanenten „Bombenangriffen" begleitet waren, konnte dann schließlich erhärtet werden, daß eine Mac-Diskette zwar bis zu 107 Dateien umfassen darf, die beispielsweise vom Mac-MS problemlos verwaltet werden, daß jedoch im 128K-RAM nur maximal 95 Dateien *einer einzigen* Diskette vom FINDER „verbucht" werden können. Dies setzt indessen voraus, daß zuvor vom FINDER *keine andere* Diskette aktiviert und somit als Disketten-Ikone auf den „Schreibtisch" gebracht wurde. Um mögliche Systemfehler des Mac-MS auszuschließen, wurde das Experiment auch mit dem Textprogramm Macwrite durchgeführt, wobei die einzelnen Textdateien eine nach der anderen manuell gespeichert werden mußten. Die Phänomene waren hier die gleichen, so daß der „Bombenangriff" am FINDER und nicht am Mac-MS liegt.

Wenn eine Diskette nunmehr 95 Dateien enthält, kann sie trotzdem nicht kopiert werden, denn bevor man eine Originaldiskette duplizieren kann, muß bekanntlich mangels der skizzierten „Blind-Operationen" das Bildchen der zukünftigen Duplikatdiskette am Monitor erscheinen, und diese enthält im günstigsten Fall als frisch formatierte Diskette mindestens 1 Datei, nämlich den „Schreibtisch". 1 + 95 = 96 Dateien: Dies wäre für Klein-Mecki zuviel, und die „Bombe" wäre unvermeidlich. Folglich reduzierten wir die Testdiskette auf dem Umweg über Mac-MS mit dem KILL-Befehl auf 94 Dateien, denn die „Papierkorb"-Option hätte erneut zu unerquicklichen „Bombenangriffen" geführt. Danach wurde die Testdiskette zunächst korrekt dupliziert, was mit 1 Laufwerk übrigens fast exakt 13 Minuten dauerte. Wir dachten, wir könnten jetzt ordnungsgemäß – also nicht über den während der anfänglichen Experimente oft benutzt Reset-Griff – die Originaldiskette auswerfen. Falsch gedacht! Es kam statt dessen die Meldung „Es ist nicht genügend Speicherkapazität verfügbar, um die Systemdiskette (gemeint war unsere Testdiskette) auszugeben. Löschen Sie die grauen Diskettenabbilder". Also mußten wir die Duplikat-Ikone in den „Papierkorb" stecken. Dieser Vorgang würde sicherlich viele Computer-Anfänger verwirren, denn wem erschiene es schon als sinnvoll, eine gerade erstellte Duplikatdiskette in den „Papierkorb" werfen?

Wer meint, daß nunmehr mit der geschrumpften 94-Datei-Diskette die Mecki-Welt wieder in Ordnung ist, irrt. Man wird vielmehr jetzt *jedesmal*, wenn man die Clean-up-Funktion wählt, *mehrere* Minuten warten müssen, bis sich Klein-Mecki wieder erholt hat. Daraus folgt, daß eine Diskette mit mehr als 90 Dateien das Mac-DOS praktisch zum Erliegen bringt. Außerdem kann man mit einer solchen Diskette de facto nur noch isoliert arbeiten, denn jede andere Diskette, die mehr als 13 Dateien (107 – 94 = 13) enthält, wird vom Betriebssystem mangels Speicherkapazität unweigerlich wieder ausgeworfen. Als Faustregel kann man davon ausgehen, daß die Summe der Dateien aller aktiven Disketten nicht wesentlich über 100-120 liegen darf, weil sonst „Bombenangriffe" unvermeidlich sind. Wenn man beispielsweise zwei Disketten A und B hat, die jeweils mehr als etwa 60 Dateien umfassen, kann man nicht mehr von der Diskette A nach B und umgekehrt kopieren, sondern muß eine fast leere Diskette C als Zwischendatenträger benutzten. Die Datei x der Diskette A wird dann zunächst auf die Zwischendiskette C und – nach dem Aus- und Einschalten des Macintosh, damit er das Directory von A vergißt – von der Zwischendiskette C auf die Diskette B kopiert. Daß dies alles nichts mehr mit Benutzer-Komfort zu tun hat, leuchtet selbst Macintosh-Fanatikern ein.

### Fazit

Ein Betriebssystem in der Art des Mac-DOS, das nicht mehr als etwa 100 Dateinamen verwalten kann und bereits unterhalb dieser Obergrenze in einer für unbedarfte Anwender völlig unverständlichen Weise Systemabstürze bewirkt und einige Funktionen wie „Aufräumen", „Duplizieren" usw. im Schneckentempo ausführt, kann nur mit einem Wort umschrieben werden: „TRASH". Cary Lu sagte in seinem „Apple Macintosh Book", S. 247: „Many people who spend several thousand dollars on a computer system develop an emotional attachment to their investment and lose all their objectivity. If you buy a Mac, you may fall prey to the same syndrome." Wir möchten hinzufügen, daß man beim Macintosh nicht nur wegen des hohen Preises, sondern auch wegen des zunächst freundlich und lustig anmutenden „Visual Interface" ein Opfer dieses Verlusts der Objektivität werden könnte. Reißt man jedoch erst einmal die Maske herunter, wird das Stückwerk um so deutlicher sichtbar. „Die Maske fällt, es bleibt der Mensch, und alles Heldentum entweicht" (Rousseau). Da paßt es wie die Faust aufs Auge, wenn die Firma Apple in ihrer Werbung den Mecki mit den Größen der (wenngleich kommunistischen) Weltgeschichte vergleicht. Immerhin haben Marx und Lenin eines dem emphemeren Mecki voraus: Man wird über sie auch noch dann sprechen, wenn Klein-Mecki längst vergessen ist.

## 4. GESAMTURTEIL

Welche Empfehlungen lassen sich nunmehr in bezug auf den Macintosh geben. Wir haben gesehen, daß der Macintosh als Positivum ein insbesondere EDV-Anfänger und „Computer-Verängstigte" ansprechendes und moralisch aufmunterndes Benutzer-Interface hat. Wir haben jedoch gleichzeitig gesehen, daß der Macintosh erstens trotz MC68000 – egal ob die von Apple verkündeten 8 MHz nun zutreffen oder nicht – ungebührlich langsam ist und daß zweitens das Mac-DOS systemimmanente, quasi „versteckte" Mängel hat, die erst bei längerer Benutzung zum Tragen kommen. Bei den ersten „Übungsdateien" ist Klein-Mecki noch quicklebendig. Erst einige Wochen nach dem Kauf, wenn sich die Datendisketten füllen, läßt er die Hose herunter.

Gegenwärtig kann man den Macintosh nur zwei Gruppen empfehlen, zum einen den Managern, die aus Prestige-Gründen ein Gerät zum Präsentieren und weniger zum Arbeiten brauchen, sowie zum anderen den Systemprogrammierern, die sich jetzt an die Arbeit machen müßten, um die gröbsten Betriebssystemfehler auszumerzen. Für reine Anwender ist der Macintosh erst in etwa 1-2 Jahren empfehlenswert, zumal erst dann die erforderliche deutsche Anwendersoftware in angemessenem Umfang vorliegen dürfte. Wer den Macintosh jetzt kauft, kann zwar wie seinerzeit beim Apple II noch einmal die Pionierphase mit all den freud- und leidvollen Entdekkungen eines echten Freaks durchmachen. Sinnvolle praktische Nutzanwendungen wie etwa beim Apple IIe ergeben sich demgegenüber im Moment noch kaum. Im übrigen scheint es uns zweifelhaft, ob sich Hobbyisten wie ehemals beim Apple II die Mühe machen werden, die „Features" des Macintosh auszubügeln. Der frühere Pepsi-Cola-Manager und jetzige Apple-Präsident John Sculley hält nicht viel von einem „Open System" in der Art des Apple II mit der damit verbundenen Preisgabe von technischen Details, die allererst die Beseitigung von „Features" ermöglichen würde. Vielmehr favorisiert er das „Closed System", „Non-Disclosure Agreements" und andere Methoden des „Schweigens". Selbst der größte amerikanische Apple-Club Call A.P.P.L.E. hatte keine Vorabunterlagen zum Macintosh erhalten. So muß denn nunmehr die Firma Apple einmal selbst zeigen, ob sie den Karren aus dem Dreck ziehen kann, damit sich nicht IBM ins Fäustchen lacht.

### Technischer Nachtrag

1. Der 68000-Prozessor läuft lt. Firma Apple mit ca. 8 MHz und nach unseren eigenen Untersuchungen mit 5,5 MHz. Die fehlenden 2,5 MHz erhöhen also nur die Taktfrequenz der Werbung.
2. Die Anzahl der Directory-Einträge hängt von der Länge der Dateinamen ab: Je länger die einzelnen Dateinamen, desto geringer die Anzahl der Einträge. Bei künstlichen 1-Zeichen-Dateinamen passen maximal 109 (inkl. Desktop) und bei „normalen" Dateinamen ca. 60-80 Einträge in das Directory. Daraus erklärt sich die nicht näher begründete Angabe aus „Mac-World", Heft 12/1984, daß Festplattenlaufwerke für den Macintosh ab ca. 60 Einträgen oft „aussteigen".
3. Die Anzahl der eigentlichen Dateinamen steht im umgekehrten Verhältnis zu den Pseudo-Subdirectory-Ordnern: Je mehr Ordner ein Directory enthält, desto geringer ist die Anzahl der eigentlichen Dateinamen. Bei Harddisks, deren Kapazität man ohnehin nicht ausschöpfen kann, sollte man deshalb auf Ordner weitgehend verzichten.
4. Es gibt inzwischen eine neue Version 1.1 des FINDERs, der aufgrund flüchtiger Tests genauso langsam und fehlerhaft wie die alte Version 1.0 zu sein scheint.
5. Ferner gibt es jetzt ein schnelles 1-Drive-Kopierprogramm, das in etwa 2,5 Minuten eine komplette Diskette „blind" dupliziert. Hier hat man die gesamte Macintosh-Philosophie total über Bord geworfen und verwendet jetzt sogar den Bildschirmspeicher als I/O-Puffer, wodurch beim Kopieren der ganze Schnickschnack vom „Schreibtisch" gefegt wird. Dieses Programm zeigt auf das deutlichste, daß man dem Mecki nur dann „Beine machen" kann, wenn man die gesamte Ikonen-Ästhetik radikal ausmerzt.

# peeker Börse

## Verkauf Software

**MACINTOSCH:** Professionelle Softwareentwicklungen für kommerzielle Anwendungen. Schreiben Sie an: H + B-EDV-Beratung, Postf. 600304, 2000 Hamburg 60.

APPLE II – PROGRAMMGENERATOR für Eingabemasken und Felder DM 59,–
DATENBANK mit Editor, Grafik- u. Listenpaket DM 65,–. Bstllg. gg. Vorkasse, Info –,80 bei:
FAHRENBERGER Rittershausstr. 4, 5300 Bonn 1

Wir haben etwas Neues für Ihren Apple! Info gegen 3,– DM in Briefm. bei FANTASTIC-Software Abt. 4A, Grasweg 7,
2857 Langen 3

* * **Finanzbuchhaltung** * * für Apple II mit CP/M. Programmierbüro Kurt Kastner, Nikolausstr. 3, 7500 Karlsruhe

**Astrologieprogramme.** Info n. Voreinsendung 1 DM in Briefmarken. C. Landscheidt, Im Dorfe 14, 2804 Lilienthal.

High-Res Taschenrechner in Assembler für Apple II/IIe/IIc mit Apple Maus. 20-stellige Rechner-Anzeige Vorkasse-Scheck 35,– DM, Kurt Nolte, Feldstr. 26,
6102 Pfungstadt

**Apple II** Lightpen für Textpase Disk mit Software ausf. Beschr. und Bauanleitung DM 40,–,
Tel. 06105/41595 W. Roell

## Verkauf Hardware

**Apple II** Supermodem alle Standards, auch BTX; komplett auf einer Karte, mit Supersoftware superbillig bei Rolf Kiupel,
Tel. 0431/555427

**EDV-ZUBEHÖR,** Akust. Koppler m. FTZ 298,–, Disketten ab 4,40, Papier, Etiketten, Farbb., Preisl. a. Anfr. W. Kotschenreuther, Gg.-Buchner-Str. 29, 85 Nürnberg 10, Tel. 0911-516739

**Gelegenheit:** 2 DISK-Drives 8" für Apple IIe DDSS incl. CTRL, sofort lauffähig m. DOS/CPM/PASC., ausbaubar bis 4×1, 6MB, softw. vorh. sowie alle tech. Unterl. Preis DM 1350 + SW,
ROTH 06221/65052.

**APPLE-HITS:** RGBinterf. + 64K+80Z.: 500,– DM und CP-80-MATRIXDRUCKER + Grafikinterf., Kabel, Bücher: 600,– DM (alles neuwertig, orig. verpackt)
Tel. 0621-412683.

Verkaufe Texam RGB/80 Zeichen Karte. VB 350,– DM.
P. Pausch Kulmbacher Str. 34
8662 Helbrechts 09252/6595.

## Verschiedenes

**An alle Apple II Systeme**
Ihr 8 Bit Computer wird Augen machen, wenn Sie ihn auf 16 Bit erweitern.
Unsere CCU 8086 128 K-1 MB, 5 MHz macht es möglich (s. „CHIP" 12/84).
Durch unser Spezial-Leasing ab DM 78,–/mtl. + MWST wird Ihr Apple II-System plötzlich wie ein IBM – PC mit MS DOS 2.11. (+ CP/M 86 etc).
Auslieferung in der Reihenfolge der Bestellung über PO Box:
Orion GmbH Microcomputer
2800 Bremen 102705
Tel.: 0421-327655
Tx. 244337 orion d

**Ein Mac-Traum** geht in Erfüllung. Deshalb opfere ich meine brandneue 10 MB Festplatte System Novo-Comp für Apple IIe. (Spezifikationen siehe peeker Sept. 84, S. 21 oder Dez. 84, S. 79). Anfragen an H. Berberich, Agnes-Bernauer-Str. 72, 8000 München 21.

## Kontakte

Wer hat **Erfahrung** mit Architekten-sw für Apple II? Bitte melden bei: H. G. Erhardt, Dannerweg 5, A-4040 Linz/DO, T: 0043-732-234077.

## Tausch

**MAC**-Software zum Tauschen/Kaufen gesucht: PF 600304, 2000 Hamburg 60

Apple II-**Software-Tausch** Liste gegen Freiumschlag, N. Abraham, Am Hohberg 13,
5880 Lüdenscheid

**Apple II** sw. Tausch: Über 7 MB sw vorh. Liste an: W. Schäfer Deswatines Str. 72, 4150 Krefeld 1

Rüter, Rahd. Str. 65, 4955 Hille

## Erscheinungs- und Anzeigenschlußtermine für peeker

| Ausgabe | Erscheinungstermin | Anzeigenschluß |
|---|---|---|
| 4 | 25. 3. 85 | 22. 2. 85 |
| 5 | 22. 4. 85 | 22. 3. 85 |
| 6 | 20. 5. 85 | 19. 4. 85 |
| 7 | 24. 6. 85 | 24. 5. 85 |
| 8 | 22. 7. 85 | 21. 6. 85 |
| 9 | 26. 8. 85 | 26. 7. 85 |
| 10 | 23. 9. 85 | 23. 8. 85 |
| 11 | 21. 10. 85 | 20. 9. 85 |
| 12 | 25. 11. 85 | 25. 10. 85 |

## Inserentenverzeichnis peeker 3/85


| | Seite |
|---|---|
| E. Böhmer, DRV, Dreieich | 2. US |
| CCP Datentechnik, Hamburg | 65 |
| Computerware, Frankfurt | 39 |
| CPS-Datenservice, Offenbach | 55 |
| Data Becker, Düsseldorf | 4. US |
| Digitanalog, Krombach | 55 |
| D.O.S. Schwäbisch Hall | 37 |
| N. Hunstig, Münster | 37 |
| IBS Computertechnik, Bielefeld | 14, 15 |
| Interkom, Isernhagen | 71 |
| Intus, Waldshut-Tiengen | 39 |
| Jeschke, Kelkheim | 55 |
| McGraw-Hill-Book, Hamburg | 69 |
| Memsoft, Frankfurt | 69 |
| E.-W. Meyer, Frohnhausen | 39 |
| Micromint, Erkrath | 37 |
| NovoComp, Trier | 55 |
| pandabooks, Berlin | 33 |
| pandasoft, Berlin | 3. US |
| Pyramid Versand Koslik, Freiburg | 9 |
| r + r electronic, Heidelberg | 43 |
| Softline, Oberkirch | 55 |
| te-wi Verlag, München | 25 |
| TLK GBR, Münster | 39 |
| Tonacord, Eckernförde | 65 |
| Ueding electronics, Menden | 33 |

# peeker Börse

## Gelegenheitsanzeigen

Sie können unter dieser Rubrik zu einem besonders günstigen Preis

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

## Musteranzeige privat

1 Druckzeile à 32 Buchstaben nur DM 5,– zuzügl. ges. MwSt.

Beispiel:

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

**nur DM 17,10 inkl. MwSt.**

## Musteranzeige gewerblich

Die einspaltige Millimeterzeile (42 mm breit) kostet 5,50 zuzügl. ges. MwSt.

Beispiel:

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

**nur DM 137,94 inkl. MwSt.**

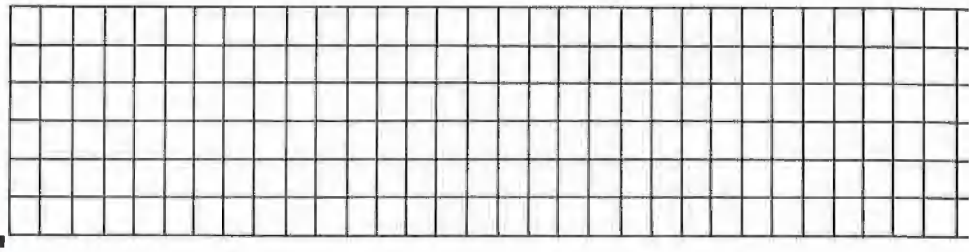
peeker MAGAZIN FÜR APPLE-COMPUTER

## peeker-Börse

### AUFTRAG FÜR KLEINANZEIGEN

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

☐ suche Hardware ☐ suche Software ☐ Tausch ☐ verschiedenes
☐ biete Hardware ☐ biete Software ☐ Kontakte ☐ Chiffre



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

peeker MAGAZIN FÜR APPLE-COMPUTER

## Produkt-Karte

Zu der in »**peeker**«, Heft_____, Seite_____ erschienenen
☐ Anzeige ☐ Produktbesprechung

über _____

_____

bitte ich um detaillierte Information.

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

| Menge | Produkt und Bestellnummer | à DM | gesamt DM |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

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

_____
Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

peeker MAGAZIN FÜR APPLE-COMPUTER

## Info-Karte

Schreiben Sie uns, wenn Sie Fragen, Ergänzungen oder Anregungen zu Aufsätzen in »**peeker**« haben:

_____

_____

_____

_____

_____

_____

## peeker-Börse

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

## Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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

## Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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.

PEEKER
MAGAZIN FÜR APPLE-COMPUTE


# Ges
# kein

**Nachdem wir im Peeker, Heft 2/1984, S. 70-71, zum Primzahlen-Wettbewerb nur den Algorithmus des Hauptgewinners veröffentlicht hatten, teilten uns viele Leser enttäuscht mit, daß sie weitere Lösungen erwartet hätten. Deshalb bringen wir nachfolgend das sehr ausführlich kommentierte Primzahlen-Programm von Michael G. Schneider, dem Gewinner des zweiten Preises. Anm. d. Red.**

# chwindigkeit ist e Hexerei

## Erastosthenes erneut betrachtet

**von Michael G. Schneider**

Im folgenden möchte ich meine Lösung der Aufgabe vorstellen. Für 1024 Durchläufe benötigt mein Programm 69.7s, so daß sich eine Laufzeit von 0.068s ergibt. Falls das im Peeker beschriebene Verfahren zur Bestimmung der Zeit angewendet wird, kommt man auf Grund der diversen Abrundungen sogar auf die noch niedrigere Zeit von 0.06s.

### 1. Der Algorithmus

Da bei der gestellten Aufgabe die Schnelligkeit das oberste Qualitätskriterium war, habe ich von Anfang an geplant, die Programmierung in Assembler durchzuführen. Vor der Kodierung war aber noch die Frage nach einem geeigneten Algorithmus zu klären. In von mir durchgeführten Tests zeigte sich, daß das „Sieb des Eratosthenes" allen anderen (mir bekannten) Verfahren weit überlegen war. Insbesondere diejenigen Methoden, welche häufige Multiplikationen und Divisionen erfordern, wurden von dem „Sieb" deutlich geschlagen.

Bevor ich meine Verbesserungen des Algorithmus erläutere, soll kurz die prinzipielle Vorgehensweise des „Siebs" beschrieben werden: Bei diesem geht man zunächst einmal davon aus, daß alle Zahlen des Intervalls (hier: 2 bis 8191) potentielle Primzahlen sind, und eliminiert erst nach und nach diejenigen Zahlen, welche als Nicht-Primzahlen erkannt wurden.
Dazu schreibt man etwa alle Zahlen des Intervalls auf ein Blatt Papier und geht diese dann in aufsteigender Reihenfolge durch. Falls dabei eine nicht durchgestrichene Zahl erreicht wird, hat man eine Primzahl p gefunden. Die zum Intervall gehörigen Vielfachen von p (also 2 * p, 3 * p, 4 * p, etc.) sind dann aber sicherlich keine Primzahlen mehr und können somit aus der Liste gestrichen werden. Nach diesen Streichungen kehrt man zu p zurück und sucht ab dort nach der nächsten Primzahl, um auch deren Vielfache zu streichen. Falls man bei dieser Suche die obere Grenze des Intervalls überschreitet, wurden alle Primzahlen herausgesiebt.

Es folgen nun die von mir zur Beschleunigung benutzten Modifikationen.
**a** Nach den Wettbewerbsbedingungen ist bekannt, daß 2 eine Primzahl ist. Aus diesem Grund müssen nicht erst alle Zahlen als potentielle Primzahlen angesehen werden. Vielmehr können von Anfang an die 2 als Primzahl, alle anderen geraden Zahlen als Nicht-Primzahlen und die ungeraden Zahlen als potentielle Primzahlen gekennzeichnet werden.
**b** Da es wegen (a) außer der 2 keine geraden Primzahlen gibt, müssen bei der Suche nach einer neuen Primzahl nur die ungeraden Zahlen untersucht werden.
**c** Zu der Primzahl p müssen nicht alle Vielfachen gestrichen werden. Da nämlich 2 * p, 4 * p, 6 * p, etc. gerade Zahlen sind, werden sie bereits in der Initialisierung (a) behandelt. Es reicht also, nur die ungeraden Vielfachen von p (3 * p, 5 * p, 7 * p, etc.) zu markieren.
**d** Eine erhebliche Beschleunigung des Algorithmus läßt sich wie folgt plausibel machen. Angenommen man hätte soeben die 11 als Primzahl entdeckt. Nach den bisherigen Ausführungen würden nun 3 * 11, 5 * 11, 7 * 11, etc. markiert. Von diesen ist aber z.B. 3 * 11 schon als Vielfaches der 3 und 5 * 11 als Vielfaches der 5 markiert worden. Da also alle i * 11 mit i < 11 bereits behandelt wurden, müssen die Streichungen auch erst bei 11 * 11 beginnen.

Insbesondere bei großen Primzahlen lohnt sich die zusätzliche Berechnung des Quadrats einer Primzahl, da dort sehr viele Zahlen übersprungen werden können. Ferner wird automatisch ein Abbruchkriterium mitgeliefert. Ist nämlich das Quadrat p * p einer Primzahl p größer als 8191, so würde bereits die erste zu streichende Zahl außerhalb des Intervalls liegen. Man kann in diesem Fall die Suche nach neuen Primzahlen einstellen.
Das bekannte Prinzip, nur Primzahlen bis zur Quadratwurzel der Obergrenze zu untersuchen, ist zwar mit dem obigen Verfahren eng verknüpft, jedoch ist es ihm weit unterlegen. Denn einerseits wird bei (d) auch eine Einschränkung der zu streichenden Zahlen vorgenommen. Und andererseits kann, wie bald gezeigt wird, ein Quadrat sehr viel einfacher berechnet werden als eine Quadratwurzel.
Auch bei Anwendung der 4 Regeln erfolgen immer noch mehrfache Streichungen. So wird z.B. 105 = 3 * 5 * 7 bei Behandlung aller drei Primfaktoren markiert. Trotzdem habe ich keinen weiter verfeinerten Algorithmus benutzt. Weitaus wichtiger erschien es mir, den vorliegenden optimal zu implementieren.

## 2. Die Implementation

Meine Implementation des Problems besteht aus drei Komponenten: dem Basicprogramm PRIM.FP und den zwei Assemblerprogrammen PRIM.0 und PRIM.1.
**PRIM.FP** beginnt bei $0803 (normaler Start für Applesoft), lädt die beiden Maschinenprogramme und bildet die Schnittstelle zum Benutzer. HIMEM:, die höchste für Applesoft zur Verfügung stehende Adresse (plus 1), wird auf $1000 gesetzt.
**PRIM.0** führt die eigentliche Berechnung der Primzahlen durch und läuft in der Zero-Page (Speicherbereich von $0000 bis $00FF). Da es aber unmöglich ist, einen File direkt in die Zero-Page zu laden, wird PRIM.0 zunächst zwischen $1000 und $10FF abgelegt. Der Bereich von $1100 bis $11FF nimmt, während PRIM.0 arbeitet, die ursprüngliche Zero-Page auf.
**PRIM.1** erledigt einige nicht direkt zum Primzahlenproblem gehörige Aufgaben und startet PRIM.0. Es nimmt den Speicher von $1200 bis $12FF ein.

Die 8192 Bytes zwischen $6000 und $7FFF dienen als Flags für die zu untersuchenden Zahlen. Deren Initialisierung erfolgt durch eine etwa 24K lange Routine, welche auf die 2 Bereiche $1300 bis $5FFF und $8100 bis etwa $9300 verteilt ist. Eine der Aufgaben von PRIM.1 wird darin bestehen, diesen Code zu generieren, so daß nicht etwa tausende von Anweisungen eingetippt werden müssen.
Es wäre sehr naheliegend gewesen, bei den Flags folgende Zuordnung vorzunehmen:

$6000: Flag für 0
$6001: Flag für 1
$6002: Flag für 2
bis
$7FFF: Flag für 8191

Jedoch hat es sich als vorteilhaft erwiesen, die Flags für gerade und ungerade Zahlen wie folgt zu trennen:

$6000: Flag für 0
$6001: Flag für 2
bis
$6FFF: Flag für 8190
$7000: Flag für 1
$7001: Flag für 3
bis
$7FFF: Flag für 8191

Falls ein Flag gleich 0 ist, so soll dies bedeuten, daß die zugehörige Zahl eine potentielle Primzahl ist. *Jeder* von 0 verschiedene Wert zeichnet eine Zahl als Nicht-Primzahl aus.

Man kann unmittelbar einsehen, daß sich für einen Index I in der Speicherstelle $7000 + I das Flag für die ungerade Zahl u = 2 * I + 1 befindet. Somit wird z.B. in $700F das Flag für u = 2 * 15 + 1 = 31 sein.

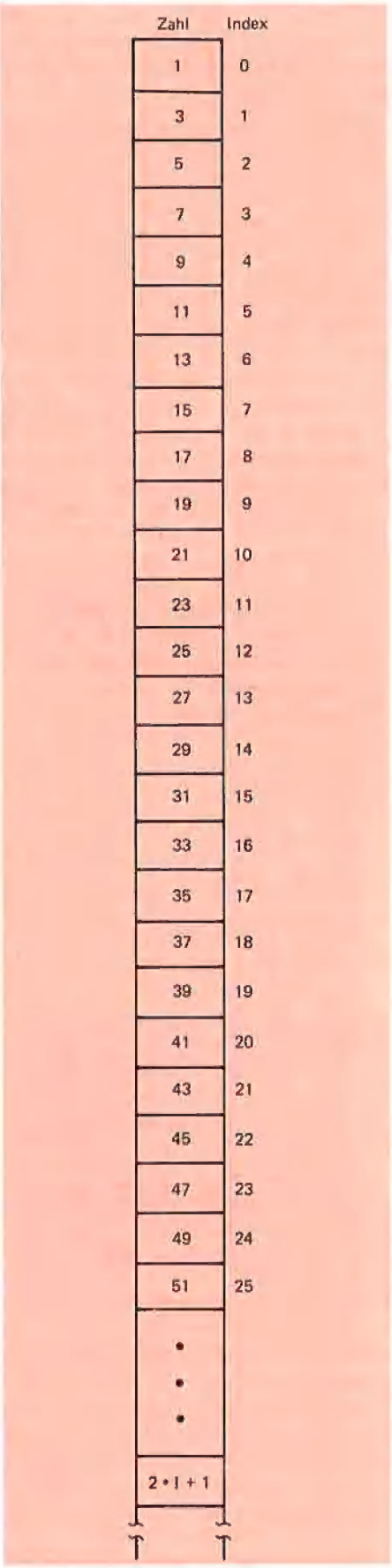
Da alle geraden Zahlen in der Initialisierung der Flag-Bereiche behandelt werden, kann man sich im weiteren auf $7000 bis $7FFF konzentrieren. Weiterhin soll zur Vereinfachung der Argumentation die additive Konstante $7000 zunächst einmal vergessen werden.

Bei der Beschreibung des Algorithmus wurde erläutert, daß das Quadrat einer Primzahl die erste zu streichende Zahl darstellt. Falls also eine ungerade Zahl u mit dem Index I gegeben ist, muß der Index für u * u gefunden werden. Dieser könnte einfach durch (u * u − 1) / 2 berechnet werden. Allgemeine Multiplikationen sind jedoch sehr zeitaufwendig, so daß ich nach einer schnelleren Methode gesucht habe.

Es seien dafür die zwei Indizes I und IQ für die ungerade Zahl u und deren Quadrat q = u * u gegeben. Dann läßt sich zeigen, daß der Index für das Quadrat der nächsten ungeraden Zahl u + 2 durch IQ + 4 * (I + 1) gegeben ist. Anstatt den (sehr einfachen) Beweis zu führen, soll ein Beispiel gegeben werden. Gesucht sei etwa der Index für 49 = 7 * 7. Angenommen man weiß aus vorherigen Überlegungen, daß zu u = 5 der Index I = 2 und zu q = 25 der Index IQ = 12 gehört. Dann weiß man auch sofort, daß der Index von 49, dem Quadrat von u + 2, durch 12 + 4 * (2 + 1) = 12 + 12 = 24 errechnet werden kann.
Da alle ungeraden Zahlen in aufsteigender Reihenfolge untersucht werden, fallen die in IQ + 4 * (I + 1) enthaltenen Teilterme automatisch an. IQ ist der Index des Quadrats der letzten ungeraden Zahl, und I + 1 ist der Index der letzten ungeraden Zahl, erhöht um 1. Somit ist I + 1 der Index der aktuellen Zahl. Als Anfangswerte werden I = 0 und IQ = 0 benutzt, was 1 * 1 = 1 entspricht.

Obwohl in der Formel IQ + 4 * (I + 1) auch noch eine Multiplikation vorkommt, wird sich diese sehr einfach realisieren lassen, da der eine Faktor immer konstant 4 ist.

Nachdem durch obige Rechnung die Adresse des ersten zu setzenden Flags




gefunden wurde, muß nur noch die Schrittweite zwischen den einzelnen Flags bestimmt werden. Nun sollten doch für eine Primzahl p die ungeraden Vielfachen 3 * p, 5 * p, 7 * p, etc. markiert werden. Da aber die geraden Zahlen ausgelagert wurden, liegen die Flags dieser Zahlen genau p Bytes voneinander entfernt.

## 3. PRIM.FP

Da das Applesoft-Programm auf jegliche Tricks verzichtet, sollen die Kommentare auf ein Minimum beschränkt werden. Die Zeilen 1000–1060 legen HIMEM: auf $1000, so daß die Binärteile des Programms nicht etwa durch Stringvariablen zerstört werden. CO definiert die Startadresse von PRIM.1, und AN bestimmt diejenige Stelle, wo die Information über die Anzahl der Durchläufe gespeichert wird. 1100–1120 laden die zwei Binärfiles nach $1000 bzw. $1200.

1200–1450 schreiben ein Menu auf den Bildschirm und erwarten vom Benutzer, daß er eine Taste von 0 bis 3 drückt. Um mit einer 1-Byte-Speicherstelle auszukommen, wird in AN die durch 4 geteilte Anzahl der Durchläufe abgelegt.

1500–1560 geben Instruktionen bezüglich der Zeitnahme und rufen PRIM.1 auf. Weil dieses unter anderem auch die Zero-Page überschreibt, darf ab hier keinesfalls RESET betätigt werden.

1600–1870 geben die Möglichkeit, die Primzahlen formatiert auszugeben. Dazu wird jeweils das i-te Flag der Bereiche für die geraden und für die ungeraden Zahlen angesehen. Nur falls dessen Wert 0 ist, liegt auch tatsächlich eine Primzahl vor. Da während des Drucks der Primzahlen kein Return ausgegeben wird, kann das Listing nicht durch Ctrl-S angehalten werden. Aus diesem Grund wird in Zeile 1850 die Tastatur ($C000) direkt abgefragt. Sollte eine Taste gedrückt worden sein, wird solange gewartet, bis eine weitere Taste gedrückt wird.

1630 bzw. 1900–1930 springen zum Start des Programms zurück und erlauben so eine zweite Zeitnahme.

## 4. PRIM.1

Das Assemblerprogramm PRIM.1 wird nacheinander die folgenden Aufgaben erfüllen: Code für die Initialisierung der Flag-Bereiche generieren, Zero-Page retten, PRIM.0 von $1000 nach $0000 kopieren, Signal zum Start der Uhr geben, mehrfach (4, 256 oder 1024mal) PRIM.0 aufrufen, Signal zum Anhalten der Uhr geben, Zero-Page wieder herstellen und nach PRIM.FP zurückkehren. Im einzelnen:

1000–1010 definieren den Ursprung (ORigin) als $1200 und dirigieren den Objektcode in den Ausgabefile (Target File) PRIM.1.BIN.

1490–1530 ist eine Routine, welche den Inhalt des Akkumulators im Speicher ablegt und den Zeiger auf den Speicher inkrementiert. Dort, wo sich momentan noch das Label DUMMY befindet, wird von der aufrufenden Stelle ein aktueller Wert eingesetzt.

1630–2240 generieren den Code zur Initialisierung der Flag-Bereiche ($6000 bis $7FFF). Zum besseren Verständnis sollten zunächst die Zeilen 2960–3040, wo sich der Anfang dieses Unterprogramms befindet, betrachtet werden: Nachdem die Flags für die Zahlen 0 bis 5 vorbereitet wurden, müssen ab INIT.1 immer ein STY $6xyz und ein STX $7xyz erzeugt werden. 1630–1690 belegen die beiden Register X und Y mit dem Adreßfeld ($7003) der ersten zu erzeugenden Anweisung und bereiten die Routine STORE vor.

1740–1800 bzw. 1820–1870 legen STY $6xyz bzw. STX $7xyz im Speicher ab. Die Anweisung AND #$EF macht aus dem Wert $7x ein $6x.

1890–1920 inkrementieren das Adreßfeld. Falls das hochwertige Byte negativ wird, wäre die nächste Adresse $8000, so daß alle Anweisungen erzeugt wurden. In diesem Fall wird in 2230–2240 noch ein RTS-Befehl an das Ende von INIT gehängt.

Sofern aber noch Anweisungen generiert werden müssen, wird in 1960–2010 überprüft, ob noch mindestens 9 Bytes bis zum Start der Flag-Bereiche vorhanden sind. Dabei errechnen sich die 9 Bytes aus einer STY-, einer STX- und einer JMP-Anweisung.

Die Zeilen 2050–2190 behandeln den Fall, daß nicht mehr genug Platz existiert. Sie generieren ein JMP INIT.2 und belegen den Zeiger in STORE + 1 mit dem neuen Wert.

Wenn INIT später zur Ausführung gelangt, werden also zunächst einige tausend Anweisungen von $1300 bis etwa $6000 ausgeführt. Dann erfolgt ein Sprung nach $8100, wo es mit den restlichen STY- und STX-Befehlen weitergeht.

Die Entscheidung, die Flags von $6000 bis $7FFF (und nicht etwa unterhalb oder oberhalb des INIT-Codes) zu speichern, kann damit begründet werden, daß dies eine schnellere Version von PRIM.0 erlaubt. Auf diese Weise ist nämlich die erste

Adresse jenseits der Flags durch $8000 gegeben. Und diese zeichnet sich durch ein negatives höherwertiges Byte aus.
2290–2350 kopieren die Zero-Page in einen sicheren Bereich und das Programm PRIM.0 an seine eigentliche Startadresse.
2400–2500 löschen die Taste (BIT $C010) und erzeugen durch zwei Schleifen ein Brummen ($C030 ist der Lautsprecher). Falls dabei irgendwann eine Taste gedrückt wird ($C000 > $7F), bricht der Brumm-Ton ab, und durch 2550–2600 erfolgt dann die verlangte Anzahl der Aufrufe von PRIM.0.
Im Anschluß daran wird durch 2650–2710 ein Klick-Ton als das Signal zum Anhalten der Uhr gegeben. 2760–2850 stellen die alte Zero-Page wieder her und kehren zu PRIM.FP zurück.

## 5. PRIM.0

Da sich jeder 6502-Anweisung eine bestimmte Zykluszahl zuordnen läßt, kann die zur Ausführung eines Befehls erforderliche Zeit exakt bestimmt werden. So ist z.B. von einem LDA #0 bekannt, daß er 2 Zyklen benötigt. Und weil der Apple II mit einer Taktfrequenz von (etwa) 1 MHz arbeitet, nimmt dieser Befehl 2/1000000 s in Anspruch.
Um nun ein möglichst schnelles Programm zu erhalten, habe ich mich bemüht, die Gesamtzahl der Zyklen zu minimieren. Eine der daraus resultierenden Maßnahmen ist etwa der Trick, PRIM.0 in der Zero-Page laufen zu lassen. Denn ein STA DATA dauert 1 Zyklus weniger, wenn DATA eine Zero-Page-Adresse ist.
Beim Speichern und Laden von Ergebnissen kann ein anderer Trick angewendet werden. Angenommen man würde in einem Teil des Programms einen Wert errechnen, der im Speicher abgelegt werden muß, da er in einem anderen Teil noch einmal verwendet wird. Normalerweise wird dazu hinter dem Code ein Byte DATA reserviert, welches dann mit STA DATA oder LDA DATA angesprochen wird.
Ich habe jedoch des öfteren die folgende Konstruktion verwendet:

```
    STA LOC+1
LOC LDA #DUMMY
```

Die STA-Anweisung speichert den Akkumulator direkt in das Operandenfeld der LDA-Anweisung. Da diese dann die unmittelbare Adressierung benutzen kann, wird 1 Zyklus eingespart. Um diejenigen Punkte im Programm, bei denen solche Veränderungen auftreten, deutlich zu machen, wird am rechten Rand des Quelltextes „<<<<< Name“ vermerkt. In 1270–1320 werden 6 derartige Stellen, deren Funktion im weiteren erklärt wird, definiert.
Bei der Suche nach einer Primzahl (1810–1950) befindet sich im X-Register der Index der aktuellen ungeraden Zahl und im Y-Register das H-Byte des Zeigers auf die Quadrate. Dessen L-Byte liegt immer in QUAD.L (H/L: High/Low 8 Bits eines 16-Bit-Wertes). Sobald man die Register anderweitig benötigt, werden sie in INDEX und QUAD.H abgelegt.
1430–1470 initialisieren die Flag-Bereiche sowie X, Y und QUAD.L. Wenn das Programm erstmalig bei NEXT (Zeile 1810) angelangt ist, weist sowohl der Index (X-Register) als auch der Zeiger auf die Quadrate [QUAD.L, Y] auf $7000.
1810–1900 erhöhen den Index und bereiten so die nächste ungerade Zahl vor. Ferner wird der Zeiger auf das Quadrat dieser Zahl berechnet (alter Zeiger + 4 * Index). Falls beim Inkrementieren des H-Bytes ein negatives Ergebnis entsteht, wäre die erste zu streichende Zahl größer als 8191. PRIM.0 kann dann verlassen werden.
1940–1950 testen, ob die vorliegende Zahl schon gestrichen wurde. Falls ja, ist der Wert des Flags ungleich 0, und die Suche nach einer Primzahl kann weitergehen. Wenn nun aber doch eine Primzahl vorliegt, müssen in 1990–2000 das X- und Y-Register gerettet werden.
Die Schrittweite zwischen den zu setzenden Flags wird in 2040–2080 durch 2 * Index + 1 berechnet und dann in INC gespeichert. Man beachte, daß ROL beim Schieben nach links das Carry-Bit in den Akkumulator zieht.
Um den folgenden Code verstehen zu können, soll zunächst einmal der prinzipielle Aufbau der MARK-Schleife vorgestellt werden. Es seien dafür das A- und Y-Register gleich 0, und das Carry nicht gesetzt:

```
MARK ADC #DUMMY
     STA DUMMY,Y
     TAY
     BCC MARK
```

Angenommen im Operandenfeld des ADC befindet sich die Schrittweite zwischen den zu setzenden Flags (immer größer als 2), und im Adreßfeld des STA die Adresse des ersten derartigen Flags. Dann legt die Schleife in allen Flags, die nicht weiter als 255 Bytes vom ersten Flag entfernt sind, den Inhalt des A-Registers ab. Es ist wichtig, daß das ADC vor dem STA erfolgt, weil sonst beim ersten Durchlauf eine 0 (Primzahl) abgespeichert werden würde. Da die Schrittweite generell eine Primzahl ist und da 256 nur Vielfaches von Zweierpotenzen sein kann, muß auch beim letzten Durchlauf im A-Register ein von 0 verschiedener Wert sein.

Um nun die Flags jenseits der 255 Bytes behandeln zu können, wird ein zweites STA DUMMY,Y eingefügt. Dessen Adreßfeld wird auf dasjenige zu setzende Flag eingestellt, welches von der ersten STA-Anweisung gerade nicht mehr erreicht wird. Da aber zwei STA-Anweisungen normalerweise noch nicht ausreichen, wird solange iteriert, bis alle Flags abgedeckt sind.

Um die verschiedenen Adressen schnell zur Verfügung zu haben, wird in 2130–2180 die kleinste Zahl bestimmt, welche größer als 255 und durch INC teilbar ist. Dazu wird INC möglichst oft mit 2 multipliziert und dann so oft addiert, bis das Resultat 255 überschreitet.

2230–2370 erzeugen dann genügend viele Adressen und legen diese in der MARK-Schleife (1560–1740) ab. Da nicht im voraus bekannt ist, wieviele STA-Anweisungen benötigt werden, muß bei MARK.E begonnen und dann in Richtung MARK hochgearbeitet werden.
Das X-Register enthält jeweils diejenige Adresse, an welcher eine neue Adresse gespeichert werden muß. Es wird daher mit $38 (MARK + 1) initialisiert. Das H-Byte der zu speichernden Adresse befindet sich im Y-Register, während das L-Byte im Akkumulator dauernd verändert wird. Von einer Adresse zur nächsten wird das L-Byte um DIFF und das H-Byte um 1 erhöht. Falls bei Addition von DIFF ein Überlauf entstehen sollte, muß das H-Byte sogar nochmals inkrementiert werden. Gestartet wird natürlich mit der Adresse des Quadrats der Primzahl.

Wenn die Zeile 2390 erreicht ist, wurden alle erforderlichen Adreßfelder der STA-Anweisungen belegt. Das X-Register zeigt nun entweder auf das Adreßfeld derjenigen STA-Anweisung, welche nicht mehr benötigt wird, oder aber auf MARK. In jedem Fall wird dort durch 2390–2430 ein ADC #DUMMY, wobei DUMMY die Schrittweite ist, gespeichert.

2450–2470 richten den relativen Sprung in Zeile 1740 auf den neuen Anfang der MARK-Schleife ein. 2490–2550 springen zu diesem Anfang hin, nachdem das A- und Y-Register mit 0 initialisiert wurden. Das Carry ist noch aus 2270–2370 gleich 0.

Nach Abschluß von MARK werden die X- und Y-Werte in 1760–1770 wieder hergestellt, und die Suche nach einer weiteren

Primzahl kann erneut aufgenommen werden.
Dies beendet die Erläuterungen zu PRIM.0, und es sollte nun auch klar sein, warum der zweite Teil der INIT-Routine bei $8100 beginnen muß: Die Adresse im Adreßfeld der STA-Anweisung direkt hinter ADC #DUMMY hat einen Wert zwischen $7F00 und $7FFF. Wenn dann durch die indizierte Adressierung noch das Y-Register addiert wird, sind effektive Adressen von $7F00 bis $80FE möglich. Somit werden also auch einige unnötige Flags jenseits von $8000 gesetzt.

## 6. Analyse

Selbstverständlich ist es auch mir nicht gelungen, auf Anhieb ein so schnelles Programm zu schreiben. Dem oben beschriebenen Code gingen 3 andere Versionen voraus, deren erste eine Laufzeit von 0.142s besaß.
Bei meinen Versuchen, irgendwo noch ein paar Zyklen einzusparen, trat immer das gleiche Problem auf: Ein statischer Programmtext sagt nicht genug über das dynamische Verhalten eines Programms aus. Man weiß zwar, wieviele Speicherplätze eine Schleife benötigt oder wieviele Zyklen ein Durchlauf erfordert. Wie oft die Schleifen dann aber tatsächlich durchlaufen werden, bleibt verborgen. Aus diesem Grund habe ich eine (hier nicht abgedruckte) Utility namens „Flag-Monitor" erstellt, mittels deren solche Ergebnisse gewonnen werden können. (Der „Flag-Monitor" wird in einem der nächsten Peeker-Hefte veröffentlicht. Anm. d. Red.)

**Tabelle 1**

| Zeile | Flag=0 | Flag=1 | Summe |
|---|---|---|---|
| 1870 | CC = 29 | CS = 16 | SU = 45 |
| 1900 | PL = 15 | MI = 1 | SU = 16 |
| 1950 | NE = 21 | EQ = 23 | SU = 44 |
| 2140 | PL = 36 | MI = 23 | SU = 59 |
| 2160 | CC = 80 | CS = 23 | SU = 103 |
| 2330 | CC = 239 | CS = 15 | SU = 254 |
| 2370 | PL = 231 | MI = 23 | SU = 254 |

In der **Tabelle 1** werden die Ergebnisse bez. der bedingten Sprünge aus PRIM.0 zusammengefaßt. Aus ihr kann man etwa entnehmen, daß der Befehl in der Zeile 1870 (BCC TEST) 45mal passiert wird. Davon gilt 29mal CC (Carry Clear) und 16mal CS (Carry Set). Die Zahlen in der Tabelle, wie auch alle weiteren Werte dieses Kapitels, beziehen sich jeweils auf einen Durchlauf von PRIM.0 (und nicht etwa auf 4, 256 oder 1024 Durchläufe).

Hiermit ist es nun möglich, die Laufzeit der Schleifen zu bestimmen. Betrachten wir dazu einmal die Zeilen 2130–2160. Darin benötigen ASL und ADC je 2 Zyklen, und abhängig davon, ob der Sprung erfolgt oder nicht, müssen für BPL und BCC je 3 bzw. 2 Zyklen angesetzt werden. Es fallen dann die folgenden Zeiten an: 59 * 2 (ASL), 36 * 3 (BPL mit PL), 23 * 2 (BPL mit MI), 103 * 2 (ADC), 80 * 3 (BCC mit CC) und 23 * 2 (BCC mit CS). Insgesamt ergeben sich somit 764 Zyklen.
Aus der gestoppten Zeit kann man errechnen, daß PRIM.0 etwa 69000 Zyklen benötigt. Da die MODULO-Schleife nur einen sehr geringen Teil der Gesamtzeit bildet, wäre es nicht übermäßig sinnvoll, hier eine Beschleunigung anzustreben.

**Tabelle 2**

| Pr | STA | CC | Zyk | Z/STA |
|---|---|---|---|---|
| 3 | 16 | 85 | 7481 | 5.44 |
| 5 | 16 | 51 | 4523 | 5.44 |
| 7 | 16 | 36 | 3218 | 5.44 |
| 11 | 16 | 23 | 2087 | 5.43 |
| 13 | 16 | 19 | 1739 | 5.43 |
| 17 | 15 | 15 | 1311 | 5.46 |
| 19 | 15 | 13 | 1147 | 5.46 |
| 23 | 14 | 11 | 923 | 5.49 |
| 29 | 15 | 8 | 737 | 5.46 |
| 31 | 13 | 8 | 647 | 5.53 |
| 37 | 14 | 6 | 538 | 5.49 |
| 41 | 12 | 6 | 468 | 5.57 |
| 43 | 13 | 5 | 431 | 5.53 |
| 47 | 11 | 5 | 371 | 5.62 |
| 53 | 11 | 4 | 309 | 5.62 |
| 59 | 8 | 4 | 234 | 5.85 |
| 61 | 8 | 4 | 234 | 5.85 |
| 67 | 7 | 3 | 167 | 5.96 |
| 71 | 6 | 3 | 147 | 6.13 |
| 73 | 5 | 3 | 127 | 6.35 |
| 79 | 4 | 3 | 107 | 6.69 |
| 83 | 2 | 3 | 67 | 8.38 |
| 89 | 1 | 2 | 35 | 11.67 |

Zyklen (Gesamt) : 27048
Z/STA (Schnitt) : 5.48

Die MARK-Schleife (1560–1740) entzieht sich bisher einer derartigen Berechnung, da sich ihre Länge von einer Primzahl zur nächsten verändert. Trotzdem habe ich auch sie analysiert und die Ergebnisse in der **Tabelle 2** zusammengefaßt. Sie ist so zu lesen, daß z.B. für die Primzahl 3 (erste Zeile) 16 STA-Anweisungen erzeugt werden. Die Schleife wird 85mal mit CC und dann noch 1mal mit CS durchlaufen. Insgesamt werden dafür 7481 Zyklen benötigt. Dividiert man die Zyklenzahl durch die Anzahl der *ausgeführten* STA-Anweisungen, erhält man 5.44.
Der letzte Wert Z/STA wird durch Zyk./(STA * (CC + 1)) errechnet. Er ist ein Maß dafür, wieviel Zeit für eine einzelne Markierung erforderlich ist.

**Tabelle 3**

| von | bis | Zyklen |
|---|---|---|
| 1430 | 1470 | 32793 |
| 1560 | 1740 | 27048 |
| 1760 | 1770 | 92 |
| 1810 | 1870 | 704 |
| 1890 | 1900 | 65 |
| 1940 | 1950 | 285 |
| 1990 | 2000 | 138 |
| 2040 | 2080 | 207 |
| 2130 | 2140 | 272 |
| 2150 | 2160 | 492 |
| 2180 | 2250 | 230 |
| 2270 | 2370 | 6118 |
| 2390 | 2550 | 690 |
| 1430 | 2550 | 69134 |

Nachdem nun auch MARK untersucht wurde, können die errechneten Zeiten für die diversen Programmsegmente in **Tabelle 3** zusammengefaßt werden. Aus ihr ersieht man, daß die Initialisierung der Flags und die MARK-Schleife am meisten Zeit erfordern. Von einiger Bedeutung ist dann nur noch die Routine zum Belegen der Adreßfelder. Da INIT perfekt ist, sollten weitere Verbesserungen in 1560–1740 oder 2270–2370 ansetzen.
Abgesehen von INIT beansprucht MARK fast die gesamte Laufzeit, so daß Z/STA zu einem Qualitätskriterium für das gesamte Programm wird. Meine erste Version von PRIM.0 hatte eine MARK-Schleife konstanter Länge mit einem Z/STA von 12. Alle durchgeführten Verbesserungen liefen im wesentlichen darauf hinaus, Z/STA zu verkleinern.
Von $7000 bis $7FFF werden insgesamt 4824 Flags gesetzt. Davon sind jedoch nur 3069 verschieden, so daß 1755 mehrfache Streichungen erfolgen. Die Flags für die Zahlen 1155 (3 * 5 * 7 * 11) und 5005 (5 * 7 * 11 * 13) werden am häufigsten gesetzt, nämlich jeweils 4mal. Von $8000 bis $80FF werden zusätzlich 111 Flags gesetzt, wovon 79 verschieden sind.
Ich habe natürlich versucht, auch den Algorithmus zu verbessern, so daß etwa die mehrfachen Markierungen eines Flags unterbleiben. Wesentliche Verkürzungen der Laufzeit konnten jedoch nicht erzielt werden, da der erhöhte Aufwand zum Generieren der MARK-Schleife jeglichen Zeitgewinn wieder zunichte machte. Lediglich die Zeilen 2270–2370 liegen mir auch in einer etwa 1200 Zyklen kürzeren Version vor. PRIM.0 wird dabei aber so lang, daß es in den System-Stack ($0100 bis $01FF) reicht. Wegen der dadurch zusätzlich zu behandelnden Probleme habe ich jene Version nicht für diesen Artikel benutzt.

## PRIM.0

```
                    1000        .OR Ø
                    1010        .TF PRIM.Ø.BIN
                    1020
                    1030 ****************************************
                    1040 *                                      *
                    1050 *      PRIM.Ø                          *
                    1060 *                                      *
                    1070 *      Michael G. Schneider            *
                    1080 *                                      *
                    1090 *      November 1984                   *
                    1100 *                                      *
                    1110 ****************************************
                    1120
                    1130 * INDEX ist ein Index bzgl. BASIS.U.
                    1140
                    1150 * <QUAD.L, QUAD.H> zeigt auf die Quadrate
                    1160 * der untersuchten Zahlen (L für Low- und
                    1161 * H für High-Byte: untere / obere 8 Bit).
                    1170
                    1180 * In INC wird die Schrittweite abgelegt.
                    1190
                    1200 * DIFF gibt eine Differenz an.
                    1210
                    1220 * JMP.ADR nimmt eine Sprungadresse auf.
                    1230
                    1240 * ADC.CODE ist ein 6502-Befehlscode.
                    1250
                    1260 *-------------------------------------
003E-               1270 INDEX          .EQ $3E
0040-               1280 QUAD.H         .EQ $40
0046-               1290 QUAD.L         .EQ $46
0060-               1300 INC            .EQ $60
0072-               1310 DIFF           .EQ $72
008D-               1320 JMP.ADR        .EQ $8D
                    1330 *-------------------------------------
0069-               1340 ADC.CODE       .EQ $69
                    1350 *-------------------------------------
1300-               1360 INIT           .EQ $1300
7000-               1370 BASIS.U        .EQ $7000
FFFF-               1380 DUMMY          .EQ $FFFF
                    1390 *-------------------------------------
                    1400
                    1410 * Initialisiere die Flag-Bereiche.
                    1420
0000- 20 00 13      1430 PRIM   JSR INIT          X = Ø, Y = $7Ø
                    1440
0003- 86 46         1450        STX QUAD.L
                    1460
0005- 4C 41 00      1470        JMP NEXT
                    1480
                    1490 * Im weiteren : X == INDEX
                    1500 *               Y == QUAD.H
                    1510
                    1520 *-------------------------------------
                    1530
                    1540 * Markiere die Vielfachen der Primzahl.
                    1550
0008- 69 FF         1560 MARK   ADC #DUMMY
000A- 99 FF FF      1570        STA DUMMY,Y
000D- 99 FF FF      1580        STA DUMMY,Y
0010- 99 FF FF      1590        STA DUMMY,Y
0013- 99 FF FF      1600        STA DUMMY,Y
0016- 99 FF FF      1610        STA DUMMY,Y
0019- 99 FF FF      1620        STA DUMMY,Y
001C- 99 FF FF      1630        STA DUMMY,Y
001F- 99 FF FF      1640        STA DUMMY,Y
0022- 99 FF FF      1650        STA DUMMY,Y
0025- 99 FF FF      1660        STA DUMMY,Y
0028- 99 FF FF      1670        STA DUMMY,Y
002B- 99 FF FF      1680        STA DUMMY,Y
002E- 99 FF FF      1690        STA DUMMY,Y
0031- 99 FF FF      1700        STA DUMMY,Y
0034- 99 FF FF      1710        STA DUMMY,Y
0037- 99 FF FF      1720 MARK.E STA DUMMY,Y
003A- A8            1730        TAY
003B- 90 CB         1740        BCC MARK
                    1750
003D- A2 FF         1760        LDX #DUMMY          <<<<< INDEX
003F- A0 FF         1770        LDY #DUMMY          <<<<< QUAD.H
                    1780
                    1790 * Berechne Zeiger auf nächstes Quadrat.
                    1800
0041- E8            1810 NEXT   INX
0042- 8A            1820        TXA
0043- 0A            1830        ASL
0044- 0A            1840        ASL
0045- 69 FF         1850        ADC #DUMMY          <<<<< QUAD.L
0047- 85 46         1860        STA QUAD.L
0049- 90 03         1870        BCC TEST
                    1880
004B- C8            1890        INY
004C- 30 41         1900        BMI RETURN
                    1910
                    1920 * Wurde die Zahl schon gestrichen?
                    1930
004E- BD 00 70      1940 TEST   LDA BASIS.U,X
0051- D0 EE         1950        BNE NEXT
                    1960
                    1970 * Es wurde eine Primzahl gefunden!
                    1980
0053- 86 3E         1990        STX INDEX
0055- 84 40         2000        STY QUAD.H
                    2010
                    2020 * Schrittweite INC = 2 * X + 1.
                    2030
0057- 8A            2040        TXA
0058- 38            2050        SEC
0059- 2A            2060        ROL
                    2070
005A- 85 60         2080        STA INC
                    2090
                    2100 * DIFF = INC - (256 modulo INC)
                    2110 *      = INC * (1 + 256//INC) - 256
                    2120
005C- 0A            2130 MODULO ASL
005D- 10 FD         2140        BPL MODULO
005F- 69 FF         2150 .1     ADC #DUMMY              <<<<< INC
0061- 90 FC         2160        BCC .1
                    2170
0063- 85 72         2180        STA DIFF
                    2190
                    2200 * Erzeuge den Code zum Durchstreichen und
                    2210 * beginne damit bei MARK.E.
                    2220
0065- 18            2230 CODE   CLC
0066- A5 46         2240        LDA QUAD.L
0068- A2 38         2250        LDX #MARK.E+1
                    2260
006A- 95 00         2270 .1     STA PRIM,X
006C- 94 01         2280        STY PRIM+1,X
006E- CA            2290        DEX
006F- CA            2300        DEX
0070- CA            2310        DEX
0071- 69 FF         2320        ADC #DUMMY              <<<<< DIFF
0073- 90 02         2330        BCC .2
0075- 18            2340        CLC
0076- C8            2350        INY
0077- C8            2360 .2     INY
0078- 10 F0         2370        BPL .1
                    2380
007A- A9 69         2390        LDA #ADC.CODE
007C- 95 00         2400        STA PRIM,X
                    2410
007E- A5 60         2420        LDA INC
0080- 95 01         2430        STA PRIM+1,X
                    2440
0082- 8A            2450        TXA
0083- 69 C3         2460        ADC #-MARK.E-6
0085- 85 3C         2470        STA MARK.E+5
                    2480
0087- A9 00         2490        LDA #Ø
0089- A8            2500        TAY
                    2510
                    2520 * Sprung zum Anfang des erzeugten Codes.
                    2530
008A- 86 8D         2540        STX JMP.ADR
008C- 4C 08 00      2550        JMP MARK            <<<<< JMP.ADR
                    2560 *-------------------------------------
                    2570
008F- 60            2580 RETURN RTS
                    2590 *-------------------------------------
```

## PRIM.1

```
                    1000        .OR $1200
                    1010        .TF PRIM.1.BIN
                    1020
                    1030 ****************************************
                    1040 *                                      *
                    1050 *      PRIM.1                          *
                    1060 *                                      *
                    1070 *      Michael G. Schneider            *
                    1080 *                                      *
                    1090 *      November 1984                   *
                    1100 *                                      *
                    1110 ****************************************
```

```
                    1120
                    1130 * JMP.CODE etc. sind 6502-Befehlscodes.
                    1140
                    1150 * PRIM.RUN ist die Laufadresse, PRIM.LOAD
                    1160 * die Ladeadresse von PRIM.0.
                    1170
                    1180 * Ab SICHER wird die Zero Page gerettet.
                    1190
                    1200 * BASIS.G bzw. BASIS.U (gerade, ungerade)
                    1210 * sind die Adressen der Flag-Bereiche.
                    1220
                    1230 * DUMMY ist ein Platzhalter.
                    1240
                    1250 * ZAEHLER wird von PRIM.FP initialisiert.
                    1260
                    1270 *-------------------------------------
004C-               1280 JMP.CODE       .EQ $4C
0060-               1290 RTS.CODE       .EQ $60
008C-               1300 STY.CODE       .EQ $8C
008E-               1310 STX.CODE       .EQ $8E
                    1320 *-------------------------------------
0000-               1330 PRIM.RUN       .EQ 0
1000-               1340 PRIM.LOAD      .EQ $1000
1100-               1350 SICHER         .EQ $1100
6000-               1360 BASIS.G        .EQ $6000
7000-               1370 BASIS.U        .EQ $7000
FFFF-               1380 DUMMY          .EQ $FFFF
                    1390 *-------------------------------------
                    1400
1200- 4C 10 12      1410        JMP GEN
                    1420
                    1430 *-------------------------------------
1203-               1440 ZAEHLER        .BS 1
                    1450 *-------------------------------------
                    1460
                    1470 * Lege das A-Register im Speicher ab.
                    1480
1204- 8D FF FF      1490 STORE  STA DUMMY
1207- EE 05 12      1500        INC STORE+1
120A- D0 03         1510        BNE .1
120C- EE 06 12      1520        INC STORE+2
120F- 60            1530 .1     RTS
                    1540 *-------------------------------------
                    1550
                    1560 * Erzeuge den für die Initialisierung der
                    1570 * Flag-Bereiche benötigten Code von $1316
                    1580 * bis $5FFF und $8100 aufwärts.
                    1590
                    1600 * (X,Y) : Adreßteil des nächsten Befehls.
                    1610 * STORE+1/+2 : Zeiger in den Speicher.
                    1620
1210- A2 03         1630 GEN    LDX #BASIS.U+3
1212- A0 70         1640        LDY /BASIS.U+3
                    1650
1214- A9 16         1660        LDA #INIT.1
1216- 8D 05 12      1670        STA STORE+1
1219- A9 13         1680        LDA /INIT.1
121B- 8D 06 12      1690        STA STORE+2
                    1700
                    1710 * Lege zwei Befehle (STY/STX) im Speicher
                    1720 * ab.
                    1730
121E- A9 8C         1740 .1     LDA #STY.CODE
1220- 20 04 12      1750        JSR STORE
1223- 8A            1760        TXA
1224- 20 04 12      1770        JSR STORE
1227- 98            1780        TYA
1228- 29 EF         1790        AND #%11101111
122A- 20 04 12      1800        JSR STORE
                    1810
122D- A9 8E         1820        LDA #STX.CODE
122F- 20 04 12      1830        JSR STORE
1232- 8A            1840        TXA
1233- 20 04 12      1850        JSR STORE
1236- 98            1860        TYA
1237- 20 04 12      1870        JSR STORE
                    1880
123A- E8            1890        INX
123B- D0 03         1900        BNE .2
123D- C8            1910        INY
123E- 30 2A         1920        BMI .3
                    1930
                    1940 * Noch genug Platz bis BASIS.G vorhanden?
                    1950
1240- AD 06 12      1960 .2     LDA STORE+2
1243- C9 5F         1970        CMP /BASIS.G-9
1245- D0 D7         1980        BNE .1
1247- AD 05 12      1990        LDA STORE+1
```

```
124A- C9 F7         2000        CMP #BASIS.G-9
124C- 90 D0         2010        BCC .1
                    2020
                    2030 * Konflikt mit BASIS.G! Also JMP INIT.2.
                    2040
124E- A9 4C         2050        LDA #JMP.CODE
1250- 20 04 12      2060        JSR STORE
1253- A9 00         2070        LDA #INIT.2
1255- 20 04 12      2080        JSR STORE
1258- A9 81         2090        LDA /INIT.2
125A- 20 04 12      2100        JSR STORE
                    2110
                    2120 * Neu-Initialisierung von STORE+1/+2.
                    2130
125D- A9 00         2140        LDA #INIT.2
125F- 8D 05 12      2150        STA STORE+1
1262- A9 81         2160        LDA /INIT.2
1264- 8D 06 12      2170        STA STORE+2
                    2180
1267- 4C 1E 12      2190        JMP .1
                    2200
                    2210 * Ein RTS an das Ende von INIT.
                    2220
126A- A9 60         2230 .3     LDA #RTS.CODE
126C- 20 04 12      2240        JSR STORE
                    2250 *-------------------------------------
                    2260
                    2270 * Tausche die Zero Page und PRIM.0 aus.
                    2280
126F- A2 00         2290 TAUSCH LDX #0
1271- B5 00         2300 .1     LDA 0,X
1273- 9D 00 11      2310        STA SICHER,X
1276- BD 00 10      2320        LDA PRIM.LOAD,X
1279- 95 00         2330        STA 0,X
127B- E8            2340        INX
127C- D0 F3         2350        BNE .1
                    2360 *-------------------------------------
                    2370
                    2380 * Zeige, daß die Generierung fertig ist.
                    2390
127E- 2C 10 C0      2400 BRUMM  BIT $C010
1281- 2C 30 C0      2410 .1     BIT $C030
1284- A2 00         2420        LDX #0
1286- A0 03         2430 .2     LDY #3
1288- 2C 00 C0      2440 .3     BIT $C000
128B- 30 08         2450        BMI AUFRUF
128D- 88            2460        DEY
128E- 10 F8         2470        BPL .3
1290- CA            2480        DEX
1291- D0 F3         2490        BNE .2
1293- F0 EC         2500        BEQ .1                    immer !
                    2510 *-------------------------------------
                    2520
                    2530 * Rufe das Primzahlenprogramm auf.
                    2540
1295- 20 00 00      2550 AUFRUF JSR PRIM.RUN
1298- 20 00 00      2560        JSR PRIM.RUN
129B- 20 00 00      2570        JSR PRIM.RUN
129E- 20 00 00      2580        JSR PRIM.RUN
12A1- CE 03 12      2590        DEC ZAEHLER
12A4- D0 EF         2600        BNE AUFRUF
                    2610 *-------------------------------------
                    2620
                    2630 * Gib das Signal zum Anhalten der Uhr.
                    2640
12A6- A0 40         2650 KLICK  LDY #$40
12A8- 2C 30 C0      2660 .1     BIT $C030
12AB- A2 1B         2670        LDX #$1B
12AD- CA            2680 .2     DEX
12AE- 10 FD         2690        BPL .2
12B0- 88            2700        DEY
12B1- D0 F5         2710        BNE .1
                    2720 *-------------------------------------
                    2730
                    2740 * Stelle die alte Zero Page wieder her.
                    2750
12B3- A2 00         2760 ORIG   LDX #0
12B5- BD 00 11      2770 .1     LDA SICHER,X
12B8- 95 00         2780        STA 0,X
12BA- E8            2790        INX
12BB- D0 F8         2800        BNE .1
                    2810
                    2820 * Zurück nach PRIM.FP.
                    2830
12BD- 2C 10 C0      2840        BIT $C010
12C0- 60            2850        RTS
                    2860 *-------------------------------------
                    2870
```

```
                2880 * INIT soll auf $1300 beginnen.
                2890
1201-           2900         .BS $3F
                2910 *-------------------------------------
                2920
                2930 * Flag = 0 : potentielle Primzahl
                2940 * Flag > 0 : keine Primzahl
                2950
1300- A2 00     2960 INIT    LDX #0
1302- A0 70     2970         LDY #$70
                2980
1304- 8C 00 60  2990         STY BASIS.G                    0
1307- 8C 00 70  3000         STY BASIS.U                    1
130A- 8E 01 60  3010         STX BASIS.G+1                  2
130D- 8E 01 70  3020         STX BASIS.U+1                  3
1310- 8C 02 60  3030         STY BASIS.G+2                  4
1313- 8E 02 70  3040         STX BASIS.U+2                  5
                3050
                3060
                3070 * Ab hier wird der Code generiert.
                3080
1316- 8C 03 60  3090 INIT.1  STY BASIS.G+3
1319- 8E 03 70  3100         STX BASIS.U+3       etc. etc. etc.
                3110
8100-           3120 INIT.2      .EQ $8100
                3130 *-------------------------------------
```

Hinweis: Dieses Programm wurde mit dem S-C-Macroassembler erstellt.

### PRIM.FP

```
1000 REM PRIM.FP
1020 HIMEM: 16 * 256: REM $1000
1040 C0 = (16 + 2) * 256: REM $1200 : Code von PRIM.1
1050 AN = C0 + 3: REM $1203 : Anzahl der Durchläufe / 4
1100 PRINT CHR$ (4);"BLOAD PRIM.0.BIN,A$1000"
1110 PRINT CHR$ (4);"BLOAD PRIM.1.BIN,A$1200"
1200 HOME
1210 PRINT "Primzahlen": PRINT
1220 PRINT "von Michael G. Schneider"
1230 PRINT "    Kranenbergstr. 82"
1240 PRINT "    5810 Witten 4": PRINT : PRINT
1300 PRINT "Programmende  (0)"
1310 PRINT "       4 mal  (1)  ---  ca  0.3 sec"
1320 PRINT "     256 mal  (2)  ---  ca 17.4 sec"
1330 PRINT "    1024 mal  (3)  ---  ca 69.7 sec": PRINT
1340 PRINT "Stoppuhr noch nicht starten : ";
1350 GET CH$: IF CH$ < "0" OR CH$ > "3" THEN 1350
1400 PRINT CH$: PRINT : PRINT
1410 IF CH$ = "0" THEN PRINT : PRINT : PRINT : END
1420 IF CH$ = "1" THEN POKE AN,1: REM 4/4
1430 IF CH$ = "2" THEN POKE AN,64: REM 256/4
1440 IF CH$ = "3" THEN POKE AN,0: REM 1024/4 (modulo 256)
1500 PRINT "Irgendwann nach Beginn des Brumm-Tons"
1510 PRINT "eine Taste drücken und in dem gleichen"
1520 PRINT "Augenblick die Stoppuhr starten.": PRINT
1530 PRINT "Das Ende wird durch KLICK angezeigt.": PRINT
1550 CALL C0: REM Start von PRIM.1
1600 PRINT "Ausgabe der Primzahlen (J/N) : ";
1610 GET CH$: IF CH$ < > "J" AND CH$ < > "N" THEN 1610
1620 PRINT CH$: PRINT : PRINT
1630 IF CH$ = "N" THEN 1200
1650 REM $6000 : Basisadresse für Flags der geraden Zahlen
1660 REM $7000 : Basisadresse für Flags der ungeraden
     Zahlen
1680 G = 6 * 16 * 256
1690 U = 7 * 16 * 256
1700 BL$ = "          "
1800 REM Formatierte Ausgabe aller Primzahlen
1820 FOR I = 0 TO 16 * 256 - 1
1830 IF PEEK (G + I) = 0 THEN PRINT RIGHT$ (BL$ + STR$ (2
     * I),8);
1840 IF PEEK (U + I) = 0 THEN PRINT RIGHT$ (BL$ + STR$ (2
     * I + 1),8);
1850 IF PEEK (12 * 16 * 256) > 127 THEN GET CH$: GET CH$
1860 NEXT I
1900 PRINT : PRINT : PRINT
1910 PRINT "Bitte Taste drücken ";
1920 GET CH$
1930 GOTO 1200
```

# Pascal 1.2
# Evolution statt Revolution

**getestet von Claus Rautenstrauch**

Seit Oktober 1984 ist die Apple Pascal Version 1.2 auf dem Markt. Was ist nun anders gegenüber der Version 1.1? Fangen wir mit der Systeminstallation an.
Für die Version 1.1 waren alle Apple-Varianten gleich. Ob diese Version nun auf einem Apple II+ der ersten Stunde oder dem vergleichsweise modernen Apple IIc installiert wurde, war sie doch immer gleich (schlecht) zu bedienen und nur recht umständlich über das Setup-Pogramm an den jeweiligen Rechnertyp anzupassen.
Anders Pascal 1.2: Hiermit kann man die Möglichkeiten des jeweiligen Apple-Typs voll ausnutzen. Zum einen gibt es für jeden Typ (II+ mit und ohne 80 Z/Z, IIe mit und ohne 80 Z/Z) eine eigene Miscinfo, zum anderen gehört auch ein 128K-SYSTEM.APPLE und 128K-SYSTEM.PASCAL zum Lieferumfang.
Der Pascal-1.2-Benutzer muß sich also zunächst einmal das für ihn optimale System zusammenstellen, braucht dann mit SETUP keine passende Miscinfo erstellen und braucht sich auch nicht damit quälen, in Assembler eine Memory-Management-Utility für die „oberen" 64K zu schreiben. Daraus kann man schon sehen, wer am meisten von der Version 1.2 profitiert, nämlich der IIe- bzw. IIc-Benutzer.

Ich möchte aber zuerst auf die neuen Features eingehen, die alle Apple-Benutzer betreffen:
**%-Präfix:** Eine wichtige Neuerung ist hier das %-Präfix. Bisher konnte man nur auf einen File zugreifen, indem man zum Filenamen explizit den Diskettennamen (z.B. DISK1: DBANK.DATA) oder die Unitnummer (z.B. #5: DBANK.DATA) angab. Mit dem %-Präfix findet man einen File immer, wenn er „online" ist, unabhängig davon, ob er in Drive 1 oder Drive 2 gespeichert ist. Beispiel: RESET (datafile, "%DBANK.DATA") öffnet automatisch den File DBANK.DATA auf jedem beliebigen Laufwerk.

**Unitnummern:** Neu ist auch die Tatsache, daß jetzt auch die Unitnummern 13 bis 20 und 128 bis 143 zur Verfügung stehen. Diese braucht man in erster Linie zum Arbeiten mit Festplatten, insbesondere mit der Profile. Die Units 128 bis 143 können nur über UNITREAD und UNITWRITE angesprochen werden, erscheinen also nicht im Directory und eignen sich somit bestens zum „Verstecken" von Daten.
Dennoch ist Pascal 1.2 von sich aus noch nicht voll festplattentauglich. Dazu braucht man noch den jeweiligen Treiber (SYSTEM.ATTACH) und den Pascal-Profile-Manager mit dem Extended Filer, die allesamt zum Lieferumfang der Profile gehören.
**Swapping-Option:** Es steht auch eine neue Swapping-Option zur Verfügung. Sie ist jetzt zweistufig: Die erste „swappt" alle RESET- und REWRITE-, die zweite auch noch alle GET- und PUT-Aufrufe. Die Programme sind mit Swapping-Option 2 sehr langsam in der Ausführung. Diese Option soll auch noch nicht benutzt werden, da der geschaffene Platz später (vielleicht bei Pascal 1.3?) für andere Zwecke genutzt werden soll. Option 1 schafft zusätzlich 2262, Option 2 zusätzlich 3084 Bytes.
**Formatter:** Zum Betriebssystem gehört auch ein neuer Formatter, wobei dieser auch Auskunft über die Disk-Speed gibt, wenn diese nicht stimmt. Sehr schade ist, daß Apple versäumt hat, die Möglichkeit zu schaffen, 40 Tracks zu formatieren, da sowohl die Duodisk als auch die Apple-IIc-Laufwerke 40 Tracks haben, was in keinem Handbuch steht!. (Bei unseren IIe-Duodisk- und IIc-Laufwerken in der Redaktion lassen sich allerdings bestenfalls 38 Spuren formatieren. Leser mögen uns bitte über ihre diesbezüglichen Experimente berichten. us).
**Gleichnamige Volume-Namen:** Abgesehen von diesen Features gibt es auf Betriebssystemebene für den Pascal Benutzer noch eine

wichtige Sache zu beachten. Die Warnung, daß die Disketten in Laufwerk 1 und 2 gleichnamig sind, sollte man unbedingt ernst nehmen. Bei der Version 1.1 konnte man diese Warnung ignorieren, wenn man mit Unitnummern arbeitete. Bei der Version 1.2 führt ein Transfer im Filer *immer* dazu, daß das Directory der Diskette aus Laufwerk 1 auf die Diskette in Laufwerk 2 kopiert wird.

Verlassen wir nun die Betriebssystemebene und schauen uns die Änderungen im Sprachbereich von Pascal 1.2 an.

**CHAINSTUFF:** Die erste Konsequenz aus dem vorhergehenden ist eine neue CHAINSTUFF-Unit. Diese akzeptiert ebenfalls das %-Präfix in der SETCHAIN-Prozedur und hat die Swapping-Prozeduren SWAPON (= Swapping Option 1) und SWAPGPON (=Swapping Option 2) sowie SWAPOFF und SWAPGPOFF.

**APPLESTUFF:** Neben der CHAINSTUFF-Unit ist auch die APPLESTUFF-Unit erweitert worden.

**REMSTATUS:** Wer einen Apple IIc oder einen anderen Apple mit der Super Serial Card (SSC) in Slot 2 hat, kann diese mit Hilfe der neuen Funktion REMSTATUS (RSCHANNEL) kontrollieren. Das ist vor allem sinnvoll, wenn man die SSC mit Modem, Btx oder Akkustikkoppler betreibt. REMSTATUS liefert einen Wert vom Typ RSTATTYPE = (RSTATBUSY, RSTATREADY, RSTATOFFLINE). Der Typ RSCHANNEL ist deklariert als TYPE RSCHANNEL = (RSOUTPUT, RSINPUT). Der Wert RSINPUT wird übergeben, wenn das Programm von der Schnittstelle lesen soll, der Wert RSOUTPUT, wenn es über die Schnittstelle schreiben soll. REMSTATUS liefert den Wert RSTATOFFLINE, wenn dort keine Schnittstelle installiert ist, den Wert RSTATBUSY, wenn über die Schnittstelle gerade gelesen oder geschrieben wird und RSTATREADY, wenn sie zwar online, aber nicht aktiv ist.
Ein Programm zur Terminalemulation könnte das Gerippe von **Listing 1** haben (aus „Pascal 1.2 Update Manual").

**Bildschirmsteuerung:** Weiterhin kann man mit Ctrl-F = CHR (6) den Cursor unsichtbar und mit Ctrl-E = CHR (5) wieder sichtbar machen. Das ist vor allem bei Benutzung der Prozedur GOTOXY ganz nützlich, weil nämlich sonst, vor allem beim Aufbau von Bildschirmmasken, der Cursor sehr unelegant über den Bildschirm flitzt. Mit Ctrl-O = CHR (15) kann man die Bildschirmausgabe auf invers und mit Ctrl-N = CHR (14) wieder auf normal schalten (gilt nur für IIc!).

**UNITSTATUS:** Außerdem ist in den Pascal-Sprachumfang die Prozedur UNITSTATUS hinzugefügt worden. Mit dieser kann man überprüfen, ob die Shift-Taste (ohne Hardware-Modifikation nur auf dem Apple IIc!), die offene Apfel-Taste (bzw. Knopf von Paddle 0) oder die geschlossene Apfel-Taste (bzw. Knopf von Paddle 1) gedrückt sind. UNITSTATUS hat drei Parameter: die Unit-Nummer, einen Packed Array of Byte (wie UNITWRITE) sowie einen Kontrollwert, der immer 1 sein muß. Die Prozedur UNITSTATUS arbeitet nur zusammen mit der Tastatur. Folglich muß die Unitnummer immer 2 sein. Der Gebrauch von UNITSTATUS ist ziemlich trickreich, denn genau wie UNITWRITE und UNITREAD arbeitet UNITSTATUS byteorientiert und liest den Tastaturpuffer Data-Byte in den Array ein. Da man sich jedoch mit BUTTON (0) und BUTTON (1) helfen kann, was auch viel einfacher ist, ist von UNITSTATUS abzuraten, zumal diese Prozedur sehr fehlerempfindlich ist. Interessant ist jedoch, daß man mit der offenen-Apfel-Taste das Bit 7 setzen und somit ASCII-Werte oberhalb von 127 erzeugen kann.

**Bugs:** Im Update Manual zum Pascal 1.2 ist ein ganzes, 5seitiges Kapitel den beseitigten Bugs gewidmet. Das sind zum größten Teil ziemlich exotische Dinge, auf die ich hier nicht näher eingehen möchte. Eine Sache ragt jedoch heraus: Die komplette Neuimplementierung der Prozedur SEEK. Zum einen arbeitet diese jetzt etwas schneller, zum anderen ist sie auch dazu geeignet, einen File zu erweitern. Hat man bei der Version 1.1 mit SEEK „hinter" den File gegriffen, so hat SEEK den Filepointer auf das letzte Byte im letzten Block des Files gesetzt. Wird durch SEEK dieser auf eine bestimmte Anzahl Records hinter den File gesetzt, so versucht es den File entsprechend zu erweitern und aufzufüllen. Ist kein Platz mehr für diese Erweiterung, so bricht das Programm mit der Fehlermeldung „no room" ab oder weist IORESULT

**Listing 1**

```
PROGRAM emulator;
  USES APPLESTUFF;
  CONST quitchar = 'Q';
  VAR buf : PACKED ARRAY [Ø..Ø] OF Ø..255;
  ...
  REPEAT
    {Daten zum Terminal schicken}
    IF REMSTATUS (RSINPUT) = RSTATREADY THEN
    {d.h. wenn Schnittstelle nicht aktiv}
    BEGIN
     UNITREAD (7, buf [Ø], 1, 12);
     {holt Daten von REMIN}
     UNITWRITE (1, buf [Ø], 1, 12);
     {schreibt Daten auf Monitor}
    END;
    {Daten von Terminal einlesen}
    IF KEYPRESS AND (REMSTATUS (RSINPUT) = RSTATREADY) THEN
    {d.h. wenn Tastendruck und Schnittstelle nicht aktiv}
    BEGIN
     UNITREAD (2, buf [Ø], 1, 12);
     {holt Daten von Tastatur}
     UNITWRITE (8, buf [Ø], 1, 12);
     {schreibt Daten auf REMOUT}
    END;
    ....
```

**Listing 2**

```
FUNCTION peek (adr : INTEGER);
  TYPE fenster : PACKED ARRAY [Ø..Ø] OF Ø..255;
  VAR ptr : ↑fenster;
  BEGIN
    MOVELEFT (adr, ptr, 2);
    peek := ptr↑[Ø]
  END;
```

**Tabelle 1**

| | Bit | Wert | Zuordnung |
|---|---|---|---|
| | Ø | Ø | Pascal arbeitet auf Betriebssystemebene |
| | Ø | 1 | Pascal Laufzeitsystem in Aktion |
| | 1 | 1 | Fließkommaoperationen werden nicht unterstützt |
| | 2 | 1 | SET-Operationen werden nicht unterstützt |
| | 5 | 1 | |
| und | 6 | Ø | 48K-Interpreter aktiv |
| | 5 | Ø | |
| und | 6 | Ø | 64K-Interpreter aktiv |
| | 5 | Ø | |
| und | 6 | 1 | 128K-Interpreter aktiv |
| | 7 | Ø | Output geht auf Textseite |
| | 7 | 1 | Output geht auf Grafikseite |

**Accelerator-Karte abstellen**

Das folgende kleine Maschinenprogramm stellt die Accelerator-IIe-Karte ab und (re)aktiviert den eigentlichen Apple-Prozessor. Damit wird das Booten der mitgelieferten Accel-Diskette überflüssig. us

```
1Ø  HOME : PRINT "Accelerator IIe abstellen"
15  DATA 141,2,192,141,4,192,141,6,192,141
2Ø  DATA 8,192,173,129,192,173,129,192,141,Ø
25  DATA 192,141,12,192,141,14,192,32,137,254
3Ø  DATA 32,147,254,32,88,252,169,21Ø,141,Ø
35  DATA 4,169,3,141,134,192,169,76,141,Ø
4Ø  DATA 255,169,Ø,141,1,255,169,255,141,2
45  DATA 255,169,Ø,141,252,255,169,255,141,253
5Ø  DATA 255,169,13,141,134,192,Ø
55  PRINT : PRINT "RESET drücken"
6Ø  RESTORE : FOR X = 4Ø96 TO 4172:
    READ Y: POKE X,Y: NEXT : CALL 4Ø96
```

# BUCH-SHOP

### Apple DOS 3.3
von Ulrich Stiehl
2. Aufl. 1984, 203 S., kart., DM 28,–

Dies ist die erste deutschsprachige Darstellung des Diskettenbetriebssystems DOS 3.3 für den Apple II/II Plus/IIe, die sich sowohl an Applesoft- als auch an Assembler-Programmierer wendet. Sinngemäß ist das Buch zweigeteilt:
Der erste Teil behandelt ausführlich die dem Applesoft-Programmierer zur Verfügung stehenden DOS-Befehle, wobei die Textfiles wegen ihrer großen Bedeutung und der vergleichsweise komplizierten Handhabung besonders dargestellt werden. Viele Textfile-Tricks werden hier zum erstenmal geschildert.
Aber auch im zweiten Teil findet der reine Applesoft-Programmierer insbesondere in dem Kapitel „Vermischte Tips, Tricks und Patches" zahlreiche Anregungen. Im übrigen ist der zweite Teil für Assembler-Programmierer gedacht. Neben einer detaillierten Beschreibung der DOS-Interna enthält dieser Teil elf vollständige RWTS-Anwenderprogramme – z. B. CPM-Refiner, DOS-lose Datendisk, TSL-Maker, File-Reader, Pseudo-Disk-Driver und Fastbrun-Routine –, die Techniken enthüllen, die bislang noch niemals publiziert worden sind. Dieses DOS-Buch ist deshalb der unentbehrliche Begleiter für jeden Apple-Programmierer.

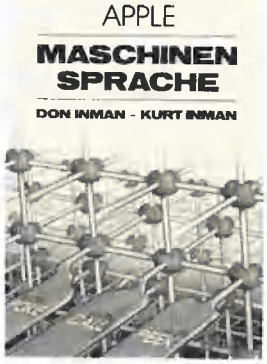
### Apple II Basic Handbuch
von Douglas Hergert
304 Seiten, 116 Abb.
DM 32,–

Das Buch ist als Nachschlagewerk konzipiert, daß seinen Platz neben jedem APPLE II, II+ und IIe haben sollte. Es richtet sich an Anfänger und fortgeschrittene Programmierer.


Aus der Praxis heraus präsentiert der Autor Tips und Vorschläge, die das Programmieren leichter und zugleich effizienter machen. Alle Applesoft- und Integer-BASIC-Begriffe sind alphabetisch aufgelistet und werden eingehend erklärt.

Dazu werden alle DOS-Befehle (neben vielen Begriffen der Computerterminologie) vorgestellt.
Beispielprogramme zeigen dem Nutzer, wie jeder Befehl funktioniert und helfen, die richtige Anwendung zu üben. Unter anderem lernt der Leser den besten Weg, um FOR/NEXT-Schleifen und IF/THEN-Entscheidungen für seine Zwecke einzusetzen.
Durch die präzise und leicht verständliche Sprache des Autors werden auch schwierige Befehle einfach in der Anwendung.

### Apple Maschinensprache
von Don und Kurt Inman
1984, 208 S., zahlr. Abb. und Tabellen, DM 49,–




Dieses Buch ist wahrscheinlich die beste Einführung in die 6502-Programmierung für denjenigen Assembler-Anfänger, der zuvor noch nie ein Maschinenprogramm geschrieben hat.
Aus dem Inhalt:
Applesoft II BASIC – kurzgefaßt – Alles über Zeichen – Alles über Speicher – Alles über Maschinenbefehle – Maschinenprogramme mit BASIC eingeben – Graphik – Text – Ton – Arithmetik – Was tun mit den Maschinenprogrammen

### Apple II leicht gemacht
von Joseph Kascmer
1984, 185 S., zahlr. Abb., kart., DM 28,–

Dies ist ein Buch, wie es sich jeder Apple-Anfänger nur wünschen kann: Schrittweise, leichtverständliche Anleitung zum Umgang mit dem Apple mit einigen durchsichtigen, unkomplizierten Beispielen in Applesoft, die ihn nicht Abschrekken, sondern ermutigen sollen, sich mit dem Gerät näher vertraut zu machen. Damit ist „Apple II leicht gemacht" das ideale Einsteigerbuch für den reinen Anwender, der nicht nur „auf den Knopf drücken", sondern zumindest einige Details aus der Black Box namens Apple erfahren will.


Aus dem Inhalt:
Kontrolle des Geräts – Schreiben und Zeichnen auf dem Bildschirm – Geheimnisvolle Abläufe: Programme – Verschiedene Eingriffsmöglichkeiten – Mobile Speicher: Disketten – Kontrollmöglichkeiten – Das Innenleben

### Apple Assembler
Tips und Tricks
von Ulrich Stiehl
1984, 226 S., 3 Abb., kart., DM 34,–

„Apple Assembler" wendet sich an alle, die bereits Anfängerkenntnisse der 6502-Programmierung haben – z. B. aufgrund des Buches „Apple Maschinensprache" – und nunmehr ein Nachschlagewerk für ihren Apple II Plus/IIe/IIc suchen, in dem alle wichtigen ROM-Routinen sowie eine Vielzahl sonstiger Hilfsprogramme in einer systematischen Form zusammengestellt werden. Insgesamt umfaßt dieses Buch über 40 Utilities, darunter mehrere völlig neuartige Programme wie Double-Lores, Double Hires, Screen-Format u. a.
Der erste Teil enthält ein Repetitorium der wichtigsten Befehle, Adressierungsarten und sonstigen Besonderheiten des 6502.
Im zweiten Teil werden alle Adressen des Monitors zusammengestellt, die für Assembler-Programmierer von Nutzen sein können. Darüber hinaus findet der Leser Unterroutinen für hexadezimale Addition/Subtraktion/Multiplikation/Division, Binär-Hex-ASCII-Umwandlung usw.
Der dritte Teil befaßt sich mit der Speicherverwaltung der Language Card und der IIe-64K-Karte und enthält Move-Programme zum Verschieben von Daten in die und aus der Language Card sowie der 64K-Karte.

Der vierte Teil ist dem Applesoft-ROM gewidmet und listet eine große Anzahl nützlicher Interpreter-Adressen. Bei den Utility-Programmen liegt das Schwergewicht auf Fließkommamathematik einschließlich Print Using.
Der letzte Teil behandelt den Text- und Graphikspeicher. Neben einem professionellen Maskengeneratorprogramm werden auch Routinen zur Double-Lores- und Double-Hires-Grafik vorgestellt.

### Arbeiten mit dem Macintosh
von N. Hesselmann
416 Seiten, 320 Abb. DM 54,–

Das Buch erklärt den Umgang mit dem Macintosh von Grund auf, wobei auch auf elementare Dinge eingegangen wird, wie z. B. die Benutzung der Tastatur und der Maus, das Einlegen von Disketten und den Systemstart. Ganz besonderes Augenmerk wird auf die Erklärung der speziellen Software-Umgebung des Macintosh gelegt, wobei das Menü- und Fensterkonzept sowie das Anwählen durch Piktogramme gekennzeichneter Funktionen klar dargestellt wird.


Der Umgang mit den Programmen MacPaint und MacWrite wird erläutert; dies geschieht teilweise anhand von Beispielen, die leicht nachvollzogen werden können. Ein umfangreiches Kapitel ist dem für den Macintosh erhältlichen Microsoft-BASIC gewidmet.

### BASIC Übungen für den Apple
von J. P. Lamoitier
1983, 252 S., zahlr. Abb., kart., DM 38,–

Das Buch ist konzipiert, allen Apple-Anwendern Applesoft-BASIC durch praktische Übungen an Hand von reellen Programmen beizubringen. Datenverarbeitung, Statistik, kommerzielle Programme, Spiele und vieles mehr. Jede Übung beinhaltet eine Beschreibung der Problemstellung, eine Analyse der Lösungsmöglichkeiten, ein Flußdiagramm und ein fertiges Programm samt Probelauf.


Aus dem Inhalt:
Ihr erstes BASIC-Programm – Flußdiagramme – Übungen mit Integerzahlen – Elementare Beispiele aus der Geometrie – Allgemeine Übungen aus der Datenverarbeitung – Mathematische Berechnungen – Kaufmännische Berechnungen – Spiele – Operations Research – Statistik

### Apple ProDOS für Aufsteiger
Band 1
von Ulrich Stiehl
1984, 202 S., kart., DM 28,–

ProDOS ist das neue „professionelle DOS" (Professional Disk Operating System) für den Apple IIe sowie den mit einer Language Card ausgestatteten Apple II Plus. Band 1 befaßt sich mit den theoretischen Grundlagen von ProDOS, der internen und externen Speicherorganisation und enthält grundlegende Beispielprogramme für Assembler-Programmierer sowie generelle Untersuchungen zum BASIC-SYSTEM. Da ProDOS über erheblich vielfältigere und leistungsfähigere, zugleich jedoch erheblich kompliziertere Dateistrukturen verfügt, sind theoretische Kenntnisse von ProDOS unabdingbar, wenn man die Features von ProDOS voll ausschöpfen will.
Aus dem Inhalt:
Ein erster Überblick – ProDOS und DOS 3.3 – Interne Speicherorganisation – Externe Speicherorganisation – MLI (Machine Language Interface) – ProDOS für Applesoft-Programmierer

**Beachten Sie die Buch-Shop-Karte**

den Wert 8 zu. Auch kann man nun einen File mit SEEK und PUT erweitern. Bisher führte das dazu, daß immer der erste angehängte Record zerstört wurde.

**Peeken der Version:** Werfen wir nun einen Blick in das Innere des Systems. Wie schon erwähnt, unterscheidet Pascal 1.2 die verschiedenen Typen der Apple-II-Familie, außerdem unterscheidet es noch die verschiedenen Pascal-Versionen und Interpreter-Typen. Diese Informationen stehen in drei Info-Bytes, die auch von einem Pascal-Programm abgefragt werden können, und zwar indem man in die entsprechenden Stellen „peekt". Eine Funktion peek gibt es (immer noch) nicht, man kann sie aber wie in **Listing 2** implementieren.
Die Adresse -16591 gibt Auskunft über den Rechnertyp. Interessant sind nur die Bits 7, 0 und 1. Sind all diese Bits 0, so handelt es sich um einen Apple II, ist Bit 7 gesetzt, um einen Apple IIe. Ist Bit 0 gesetzt, ist eine 80-Zeichen-Karte installiert, ist Bit 1 gesetzt, stehen auch noch die zusätzlichen 64K zur Verfügung.
Die Adresse -16607 beinhaltet die Information über die jeweils aktive Pascal-Version. Ist ihr Wert 3, so ist die Version 1.2 aktiv und beim Wert 2 die Version 1.1.
Die aktive Interpreter-Version erfährt man aus der Adresse -16606. Die **Tabelle 1** gibt Aufschluß über die einzelnen Bitzuordnungen

**Speicherverwaltung:** Die nun folgenden Features sind nur möglich, wenn man einen Apple IIe mit 128K oder Apple IIc benutzt. Wie schon zu Anfang erwähnt, werden diese „oberen" 64K von Pascal 1.2 ausgenutzt, wobei allerdings die Unterstützung der Double Lores und Hires fehlt. Aber darüber kommt man schnell hinweg, denn die jetzt offenstehenden Möglichkeiten sind recht beeindruckend. Die oberen 64K werden nicht als RAM-Disk wie beim ProDOS, sondern „pseudo-echt" verwendet, d.h. dem Pascal-Programmierer wird suggeriert, er könne mit echten 128K arbeiten. Dazu muß man wissen, daß beim Pascal 64K-System nur der Bereich von $0C00 bis $B000, also 41K als User-Bereich zur Verfügung stehen. Diesen Bereich teilen sich dann also der Pascal-Program-Stack (p-Code), der Daten-Stack (für die dynamischen Variablen), der Daten-Heap (für die statischen Variablen) und der eventuell eingesetzte Assembler-Code. Außerdem ist in diesem Bereich auch noch die HGR Seite 1 und 2. Wer schon einmal mit größeren Datenmengen im Speicher gearbeitet hat, dem ist bestimmt der Stack-Overflow vertraut, der vom unzureichenden Speicherraum herrührt. Die 128K-Version teilt den Speicher in beiden Sektionen, d.h. in den oberen wie den unteren 64K genauso auf; auch hier wird der Bereich von $0000 bis $0C00 und der Bereich von $B000 bis $FFFF vom Betriebssystem belegt, der Rest, also 2 * 41K = 2K, steht dem Benutzer zur Verfügung. Es ist aber nun nicht so, daß im oberen und unteren User-Bereich (= die 41K in den oberen bzw. unteren 64K) die Daten, bestehend aus Assembler- und p-Code, Daten-Stack und -Heap, so „gemischt" abgelegt werden, sondern es ist der obere Bereich nur für p-Code und der untere nur für die anderen Daten reserviert. Daraus folgt, daß man im Editor 10K mehr Platz hat (knapp 28K), beim Compilieren keine Swapping-Option mehr braucht, größere Datenstrukturen in den Speicher passen (Programme werden schneller) und damit auch der Stack-Overflow wesentlich später erfolgt – und das alles ohne programmiertechnische Verrenkungen. Außerdem werden immer größtmögliche Betriebssystemteile mit in den Speicher gezogen, z.B. wird der Diskettenzugriff beim Compilieren halbiert.

**Program-Library:** Aber Apple hat auch eine Delikatesse in das System eingebaut: Man kann nun von einem Programm aus bis zu 64 Segmente benutzen (statt bisher 32). Da man aber weiterhin nur 16 Segmente im Speicher haben darf, müssen diese zusätzlichen Segmente als Intrinsic Units in eine Library eingebaut werden. Das 64K-System akzeptiert nur die SYSTEM.LIBRARY als „Behälter" für diese Units. Da diese jedoch nur 16 Segmente umfassen darf, erlaubt das 128K-System weitere Libraries, die zudem dank des %-Präfixes auf mehrere Disketten verteilt sein können, die jedoch alle während des Programmlaufs online sein müssen. Auch diese zusätzlichen Libraries dürfen nur 16 Segmente beinhalten. Will man nun von einem Programm aus nur eine solche weitere Library benutzen, so gibt man dieser Library denselben Namen wie dem Host-Programm und ersetzt den Suffix „.CODE" durch „.LIB". Eine solche Library heißt Program-Library. Man greift auf die Units der Program-Libraries zu, indem man diese in der uses-Klausel des Host-Programmes so deklariert, als ständen sie in der SYSTEM.LIBRARY. Das Betriebssystem sucht dann selbständig die Unit in der Program-Library, und wenn sie sie dort nicht findet, in der SYSTEM.LIBRARY. Will man nun mehrere Libraries von einem Programm aus benutzen, so muß man einen „Library-Name-File" anlegen. Dies ist ein Textfile (d.h. er wird im Editor erstellt), der die Namen aller Libraries enthält, die von dem Programm benutzt werden. Das Format dieses Textfiles soll hier an einem Beispiel gezeigt werden:

```
LIBRARY FILES:
DISK1:LIB1.LIB
DISK1:LIB2.LIB
DISK2:LIB3.LIB
DISK2:POWER.CODE
$$
```

Dieser Textfile muß zwingend ohne Rand und Leerzeilen eingegeben werden. Man kann auch ausführbare Programme wie Libraries behandeln, d.h. ein Zugriff auf Datenstrukturen eines anderen Programmes ist hier gewährleistet, ohne daß das Programm in eine Regular-Unit umgewandelt wird. Der Name des Library-Name-Files wird nach demselben Muster wie der Name einer Program-Library gebildet. Ein Programm kann also nicht gleichzeitig auf eine Program-Library und einen Library-Name-File zugreifen, da diese gleichnamig wären. Ein Beispiel: Ist HALLO.CODE unser Programm, so müssen auf der Diskette „DISK1:" folgende Files sein:
HALLO.CODE (Host-Programm)
LIB1.LIB (Library)
LIB2.LIB (Library)
HALLO.LIB (Library-Name-File)
Ferner müssen sich auf der „DISK2:" nachstehende Files befinden:
LIB3.LIB (Library)
POWER.CODE (ausführbares Programm)
Auch hier ist es wieder so, daß man die Units aus diesen Libraries einfach in die Uses-Klausel des Host-Programms einbauen kann, ohne Angabe, in welcher Library die Unit nun tatsächlich steht. Die richtige Library wird vom Betriebssystem selbständig gesucht.

**Kompatibilität:** Zu guter Letzt möchte ich noch auf die Kompatibilität Pascal 1.1 versus Pascal 1.2 eingehen. Für unter Pascal 1.1 erstellte Pascal- und Assembler-Programme gilt nach bisheriger Erfahrung uneingeschränkte Aufwärtskompatibilität, d.h. alle diese Programme laufen auch unter Pascal 1.2. Ebenso laufen alle unter Pascal 1.2 erstellten Programme, die keine der Neuerungen ausnutzen und nicht zuviel Speicherplatz verbrauchen (128K), unter Pascal 1.1. Man kann die System-Files beider Systeme mischen, sollte das aber niemals tun, z.B. kann ein Pascal-1.1-Linker nichts mit einem Segment mit der Nr. 33 anfangen, obwohl er sonst einwandfrei laufen würde. SYSTEM.APPLE und SYSTEM.PASCAL müssen allerdings immer aus der gleichen Version stammen, auch lassen sich keine 64K-SYSTEM.PASCAL-SYSTEM.-APPLE-Files mit 128K-SYSTEM.-PASCAL-SYSTEM.APPLE-Files mischen. Auch unter Pascal 1.1 erstellte Units laufen unter Pascal 1.2 normalerweise einwandfrei. Einzige (bisher bekannte) Ausnahme ist die CHAINSTUFF-Unit aus der Version 1.1. Sie arbeitet nicht unter Pascal 1.2. Es gibt aber auch keine Fehlermeldung.
Ein scheinbares Kuriosum ist die Tatsache, daß Pascal 1.2 beim Apple IIc auch von Laufwerk 2 bootet, nicht aber beim Apple IIe. Grund dafür sind die modifizierten ROMs des Apple IIc. Da aber der Apple IIe ab Mitte Juni 1985 mit denselben ROMs ausgeliefert wird, ist es dann auch möglich, Pascal von beliebigen Laufwerken und somit auch von Festplatten zu booten.
Unter dem Strich kann man sagen, daß Apple mit dem Pascal 1.2 zwar kein revolutionäres System, aber doch ein sehr ordentliches und professionelles Werk zur Software-Entwicklung geschaffen hat, dessen Möglichkeiten zur Erstellung komplexer Projekte keine Konkurrenz fürchten braucht.

# HELP-WARE!


Das

## Macintosh Anwenderhandbuch

**enthält:**

**• Einführung • Von Mäusen und Menüs • Fenster • Tastatur und Texteingabe • Dateien • Kleine Helfer • Zentralprozessor und Peripheriebausteine • Bild und Ton • Maus und Tastatur • Disketten, Drucker, Kommunikation • Textverarbeitung • Malen mit der Maus • Kommunikation • Kalkulation • Einblick in die Toolbox • Pascal • BASIC und FORTH • Macintosh und IBM PC im Vergleich u. v. m.**

**156 Seiten, DM 39,80**

**Fordern Sie unseren Gesamtprospekt an! Coupon ausschneiden und einsenden an: McGraw-Hill Book Company GmbH Lademannbogen 136, 2000 Hamburg 63**

.....

**Bitte senden Sie mir den Gesamtprospekt**

**Name** _____

**Anschrift** _____

_____

_____

# MACHEN SIE IHREM APPLE® EIN GESCHENK! FÜR NUR:

**59 DM**
(inkl. MWST)

**Großer MEMSOFT Wettbewerb:**
GEWINNEN SIE EINE REISE ZUR SONNENSEITE DES LEBENS!*

* Kostenloser Wettbewerb ohne jegliche Kaufverpflichtung. Teilnahmescheine erhalten Sie direkt bei MEMSOFT oder bei Ihrem Händler. Bei mehreren richtigen Einsendungen wird der Gewinner – unter Ausschluß des Rechtsweges – durch Los ermittelt. Die Verlosung erfolgt unter notarieller Aufsicht. Der Rechtsweg ist ausgeschlossen.

Entdecken Sie mit
**MEMDOS JUNIOR**
wie in Frankreich schon
8000 vor Ihnen die sagenhaften
Möglichkeiten von MEMDOS :
– den Maskengenerator
– den Index-Sequentiellen Zugriff auf Ihre Daten
– die Fehlerbehandlungsroutinen
– die Unterprogramme mit Parameterübergabe
und vieles mehr.
Für Apple II, IIe, IIc oder III.
Bestehend aus einer Diskette plus Handbuch.
(Apple II + mit Speicher-erweiterung 16 K)

MEMDOS
wurde von der Firma APPLE der "Goldene Apfel" für das beste Systemprogramm verliehen.

**BESTELLSCHEIN** zu schicken an **MEMSOFT GmbH**
Dreieichstr. 59 - 6000 Frankfurt/Main 70

☐ Ich bezahle DM 59.– + DM 3.00 Porto = DM 62.00 mit beiliegendem Verrechnungsscheck.
☐ Ich bezahle DM 59.– + DM 6.20 Nachnahmegebühr = DM 65.20 per Nachnahme.

NAME .....

FIRMA .....

ADRESSE .....

..... TEL.....

Ich nutze meinen Apple ☐ geschäftlich ☐ hobbymäßig.

Ich benutze einen Apple..... mit folgender Peripherie .....

☐ Ich möchte, ohne jegliche Kaufverpflichtung, an dem Wettbewerb teilnehmen, und fordere hiermit einen Teilnahmeschein an.

CP FRANCE CONSEIL

# Double-Hires-Grafik-Programme

## Doublestuff Plus

**getestet von Thomas Bühner**


Besitzer einer erweiterten 80-Zeichen-Karte für den Apple IIe (= 64K-Karte) können die gewohnte Grafik des Apple in zweierlei Hinsicht verbessern: Braucht man Farbgrafik, so kann man nun statt der bisherigen 6 Farben ein Spektrum von 15 Farben wählen. Bei der Schwarz-Weiß-Grafik stehen jetzt 560 horizontale Einzelpunkte statt 280 zur Verfügung. Zumindest in der Theorie. Denn bisher gibt es kaum Programme, die diese neuen Qualitäten unterstützen.

Als sich die Mehrzahl der Hobbyprogrammierer noch fragte, was denn nun eigentlich genau unter der Bezeichnung „doppelt hochauflösende Grafik" (Double Hires) beim Apple IIe zu verstehen sei, brachten zwei junge Amerikaner eine Erweiterung für Applesoft-Basic, **Doublestuff** genannt, auf den Markt, die es den Besitzern der 64K-Karte des Apple IIe (beim Apple IIc auf der Platine integriert) möglich macht, in eigenen Programmen die gewohnten Grafikbefehle wie HGR, HCOLOR, HPLOT etc. zu verwenden – nun aber mit 15 statt 6 Farben und 560 statt 280 horizontalen Bildpunkten. Da das Applesoft-Basic einfach verändert und in die 16K-Karte (Language Card) geschoben wird, geht im normalen Arbeitsspeicher nur wenig mehr als 1K an Platz verloren.

Um dem Benutzer die Programmierarbeit abzunehmen, schoben die beiden Autoren wenig später ein System von Anwendungsprogrammen nach, mit dem in doppelt hochauflösender Grafik gezeichnet werden kann, und nannten es **Doublestuff Plus**. Als Ergänzung bietet es die Möglichkeit, doppelt hochauflösende Shapes zu erzeugen, die dann im System oder in eigenen Doublestuff-Programmen Verwendung finden.

Das Doublestuff-Plus-System kann mit Joystick, Paddle, Koala Pad oder Tastatur gesteuert werden. Um damit zu zeichnen, wählt man je eine Farbe für Hintergrund und Zeichenstift, sagen wir dunkelgrün auf hellblauem Grund, und kann dann schon beginnen. Um unregelmäßige Strukturen wie Wolken, Bäume, Gesichter usw. zu schaffen, wird freihändig gesteuert: Der Zeichenstift, dessen Aussehen und Größe Sie zuvor festgelegt haben, folgt allen Bewegungen Ihres Steuersystems. Beschreiben Sie mit dem Joystick einen Kreis, so sehen Sie ihn auf Ihrem Bildschirm nachgezeichnet.

Für gerade Gebilde wie Häuser und Brücken geben Sie einen Anfangs- und Endpunkt an. Dann wird eine Linie in der von Ihnen gewählten Farbe gezogen. Haben Sie vor, bestimmte Formen häufiger zu verwenden, so schaffen Sie Punkt für Punkt ein sogenanntes Shape, eine Figur also, die genau ihren Vorstellungen entspricht und die Sie nunmehr immer wieder in Ihren Kreationen verwenden können.

Müssen Sie Text zum Bild hinzufügen, so können Sie auch das tun. Ein amerikanischer Zeichensatz (also ohne Umlaute) ist im System inbegriffen. Dieser Zeichensatz kann von Ihnen auch jederzeit geändert werden.

Soweit das Handbuch und die Absicht der Programmautoren. Lassen Sie uns nun erste Versuche wagen. Noch bevor wir beginnen, ärgert es uns vielleicht ein wenig, daß wir die Apple-Maus hier nicht verwenden können, obwohl dieses Gerät zur Zeit der Veröffentlichung des Systems schon verfügbar war. Versuchen wir also etwas ganz Einfaches: Wir wollen mit dem Joystick einen braunen Kreis malen (siehe **Bild 1**, Mitte). Hoppla, das Ergebnis hat recht wenig mit unseren Absichten zu tun. Nun gut, unser Joystick hat zwar über DM 150,- gekostet, aber daß er nicht ganz linear ist, ist uns schon klar, nachdem er ausgiebig für Spiele verwendet wurde. Also leihen wir uns von einem Freund das gleiche Modell, allerdings nagelneu. Das Ergebnis ist leider nicht viel besser.

Da unser Freund sich auch ein Koala Pad zugelegt hat, wollen wir es einmal damit versuchen, denn unser gutes, altes Grafiktablett nützt uns bei diesem System ja nichts. Wieder ein Kreis. Nachdem der Zeichenstift auf dem Bildschirm auch hier wieder etliche Male auf- und abgesprungen ist, was ja nicht an der fehlenden Linearität des Koala Pad liegen kann, da das sonst ausgezeichnet funktioniert, erhalten wir als bestes Ergebnis das Gebilde links oben auf Bild 1. Die Geduld, mit den Pfeiltasten der Tastatur einen dritten Versuch zu starten, um einen runden Kreis zu erhalten, haben wir nicht.

Immerhin versuchen wir noch, eine Sonne (gelb) und ein Viereck (schwarz) mit Hilfe der Linienfunktion zu zeichnen, bevor wir unsere ersten Shapes anfertigen. Der Systemteil zur Shape-Erstellung ist wie alle anderen Teile leicht und durchschaubar zu bedienen. Nachdem wir festgestellt haben, daß ein Kommando, das zum Shape-Zeichnen erforderlich ist, nicht funktioniert, haben wir als Ergebnis einer halben Stunde Arbeit ein herrliches Raumschiff-Shape. Leider stellt sich der Computer plötzlich tot, als wir versuchen, das Shape auf unserer Diskette abzuspeichern und ist nur durch einen Neustart des Programms wieder zum Leben zu erwecken. Also auf ein Neues!

Schließlich haben wir unser Shape fertig und wollen es nun auf unserem Meisterbild verewigen. Als wir das Bild laden, stellen wir fest, daß nur ein kleiner Teil davon abgespeichert wurde, weil die Diskette dann voll war. Bemerkt haben wir das nicht, denn Doublestuff Plus hat versäumt, uns auf diesen Fehler aufmerksam zu machen. Die einzige Möglichkeit sicherzugehen, daß beim Abspeichern alles geklappt hat, besteht in der vorherigen Überprüfung der Sektorenzahl auf der Diskette.

Endlich haben wir das Bild noch einmal gemalt und setzen jetzt unser Raumschiff hinein. Fast hätte das auch geklappt, wenn das System uns nicht die lapidare Meldung gegeben hätte, daß ein irgendwie gearteter Fehler aufgetreten sei. Das ist uns nun inzwischen nichts mehr Neues. Um Ergebnisse wie das mitgelieferte **Bild 2** zu erzielen, muß man wohl jeden Punkt einzeln setzen.

Ich habe in der Zwischenzeit schon des öfteren mit dem Doublestuff Basic, das im Systempreis inbegriffen ist, programmiert, und die Ergebnisse können sich sehen lassen. Nie ist ein Fehler aufgetaucht, und die Grafikbefehle funktionieren tatsächlich wie gewohnt, nur eben mit mehr Farben oder doppelter horizontaler Auflösung. Das Compilieren dieser Programme klappt zwar, die Ergebnisse der compilierten Programme haben aber nur noch eine gewisse Ähnlichkeit mit dem uncompilierten Basic und können keine Grafikbefehle auf der rechten Bildschirmhälfte ausführen.

Bezugsquelle in den USA: Doublestuff Software Inc., in Deutschland Softline R. Alverdes, Schwarzwaldstraße 8a, 7602 Oberkirch, Preis DM 269,- (Das Testprogramm wurde uns von Softline zur Verfügung gestellt. Anm. d. Red.)

## Superplot

### Anmerkungen von U. Stiehl

Das Superplot-Programm ist von K.-W. Bott nach meinen Spezifikationen für unseren Hüthig Software Service geschrieben worden. Ich beschränke mich deshalb auf eine selbstkritische Aufzählung der technischen Daten. Mit Superplot sollte ein preiswertes, sehr kompaktes und möglichst schnelles Maschinenprogramm geschaffen werden, das ungeschützt ist und sich problemlos in Applesoft-Programme einbinden läßt und sowohl unter DOS wie auch ProDOS lauffähig ist. Der Anwendungszweck von Superplot sollte das wissenschaftliche Funktionsplotten sein.

Dieses Ziel kann als fast erreicht gelten (siehe unten). Superplot verschiebt sich wahlweise in die Bank 1 oder 2 der Language Card ab $D000 bis $DFFF, wobei der ASCII-Zeichensatz für die Beschriftung von Grafiken ebenfalls in diesem Bereich, und zwar ab $D600 liegt. (Man kann darüber hinaus auch eigene Shape-Tabellen in die LC schieben.) Ein kleiner Driver, der die LC an- und abschaltet, liegt ab $0300 in den unteren 48K. Superplot funktioniert auch dann, wenn DOS 3.3 in die Bank 2 der LC oder PRODOS in die Bank 1 der LC geschoben wurde. Dies ist bei Doublestuff nicht möglich, da es die gesamte LC belegt. Ferner hat man bei Superplot – wenn man von dem nur wenige Bytes umfassenden Driver ab $0300 absieht – die ganzen unteren 48K für das eigene Anwenderprogramm frei.

Superplot unterstützt alle HGR-Befehle mit Ausnahme von HCOLOR und ROT. Obgleich diese Befehle für wissenschaftliches Funktionsplotten wohl kaum gebraucht werden, dürften Sie von vielen Benutzern, insbesondere von Farbmonitor-Besitzern, vermißt werden. Diese Befehle fehlen, weil sie sich in dem Bereich $D000-$DFFF nicht mehr unterbringen ließen.

Bei den Superplot-Befehlen braucht man lediglich den Ampersand vor den normalen HGR-Befehl setzen, also z.B.

& HPLOT 10, 20 TO 30, 40

Einige wenige Befehle wurden zusätzlich implementiert. So aktiviert beispielsweise & H den Grafikbildschirm, ohne den Bildschirm selbst zu löschen (wie bei & HGR).

Superplot ist ein spartanisches Programm, das sich jedoch gerade deshalb leicht in eigene Applesoft-Programme einbinden läßt. Demgegenüber ist Doublestuff mehr als Stand-alone-Programmpaket gedacht, das jedoch über erheblich mehr Möglichkeiten verfügt. Der vorgesehene Anwendungszweck von Superplot wäre voll erreicht, wenn HCOLOR und ROT nicht fehlten und wenn Superplot-Applikationen auch compilierbar wären (wozu allerdings auch die Compiler wie TASC usw. modifiziert werden müßten). Wer umfangreiche Freihandzeichnungen anfertigen will, sollte nicht Superplot, sondern Doublestuff Plus erwerben.

Bezugsquelle: Unser Hüthig Software Service oder einschlägige Apple-Händler, Preis DM 48,-

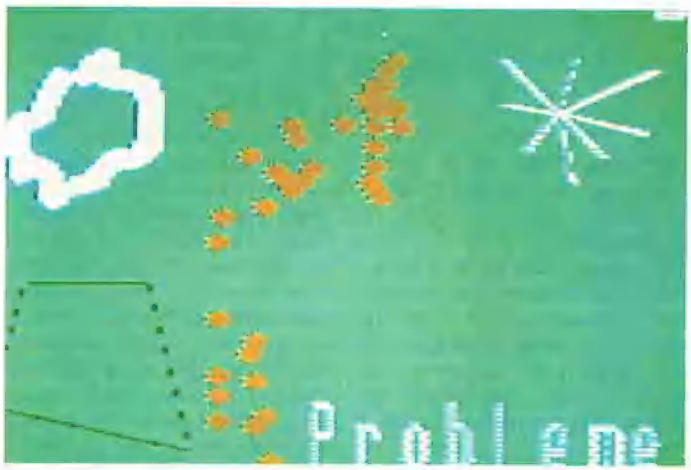



*Bild 1: Grafik mit Doublestuff erstellt.*


*Bild 2: Grafik mit Videokamera erstellt.*

### Eingegangene Testgeräte

Die nachfolgenden Testgeräte sind eingegangen und werden im nächsten Peeker-Heft ausführlich unter die Lupe genommen:

**Chinon-Laufwerk:** Preiswertes, laufruhiges 40-Spur-Slimline-Laufwerk mit Controller der Firma D.O.S. Computersysteme, Am Künbach 42, 7170 Schwäbisch Hall, Tel. 0791/51736

**Erphi-Controller:** Diesen Laufwerk-Controller mit den dazugehörigen Erphi-Drives (80 Spuren) werden wir detailliert beschreiben. Schon jetzt können wir Ihnen verraten, daß dieser innovative Controller mit praktisch jedem Betriebssystem einschließlich Pro-DOS (verschiedene Versionen) und Diversi-DOS läuft. Beachten Sie, daß die Firma Erphi nicht direkt, sondern nur über den Fachhandel liefert, weshalb wir hier auch keine Anschrift angeben.

**Besuchen Sie uns auf der Hobby-tronic!**

# Hardbreaker und Softbreaker oder wie man Programme „entschützt“

**von U. Stiehl**

Seriöse Anwender, die von rechtmäßig erworbenen Programmen Sicherungs- oder Arbeitskopien für den Eigengebrauch anfertigen wollen, haben es angesichts der immer raffinierteren Schutzverfahren zunehmend schwerer. Es soll an dieser Stelle über die rechtliche Seite des Schützens und „Knakkens“ nicht diskutiert werden, was Gegenstand eines gesonderten Peeker-Aufsatzes sein wird. Allerdings soll bereits hier darauf hingewiesen werden, daß einerseits das Anfertigen von „Vervielfältigungsstücken“ für den Eigengebrauch grundsätzlich zulässig ist, während die entgeltliche Weitergabe solcher „Vervielfältigungsstücke“ für den Fremdgebrauch grundsätzlich unzulässig ist. Gerade in der letzten Woche hat wieder einmal die Polizei an der Tür einer Studentenbude angeklopft und dann in einer Blitzaktion alles beschlagnahmt, was nicht niet- und nagelfest war (nicht nur die mutmaßlichen Raubkopien, sondern auch die gesamte Apple-Hardware als mutmaßliches Raubkopierer-Werkzeug). Deshalb hier der dringende Appell an alle diejenigen Jugendlichen und weniger Jugendlichen, die auf diese Weise leicht zu Geld kommen wollen: Warten Sie nicht, bis es an der Tür klopft! Und noch etwas: Ich weiß, daß die Produkte unseres Hüthig Software Service in einigen „Angebotslisten“ auftauchen. Unsere Produkte verkaufen wir so preiswert (zwischen DM 20,- und DM 98,-), daß schon von daher keine Notwendigkeit zu Schwarzkopien besteht. Helfen Sie uns, die niedrigen Preise zu halten und bringen Sie deshalb bitte nicht unsere Programme in Umlauf. Danke.

Es gibt z.Zt. prinzipiell 5 Möglichkeiten, geschützte Programme zu duplizieren:

**Boot-Trace-Methode:** Man macht die Programmdiskette ungeschützt, indem man die Schutzroutinen entfernt. Dieses Verfahren ist extrem zeitaufwendig, aber theoretisch fast immer erfolgreich, indem man die Stufen des Bootprozesses sukzessive nachvollzieht und dann geeignete Modifikationen („Patches“) vornimmt.

**Soft-Kopie-Methode:** Dieses Verfahren beruht darauf, daß man mit den *normalen* Diskettenlaufwerken unter Zuhilfenahme eines Spezialkopierprogramms, das üblicherweise ein „Nibble“-Kopierer ist (z. B. Locksmith usw.), ein Duplikat anzufertigen versucht. Der Erfolg dieses Verfahren hängt davon ab, ob das Kopierprogramm ein „State-of-the-Art“-Produkt ist oder nicht, denn veraltete Bit- oder Nibble-Kopierer können mit neuartigen Schutzverfahren nicht mithalten.

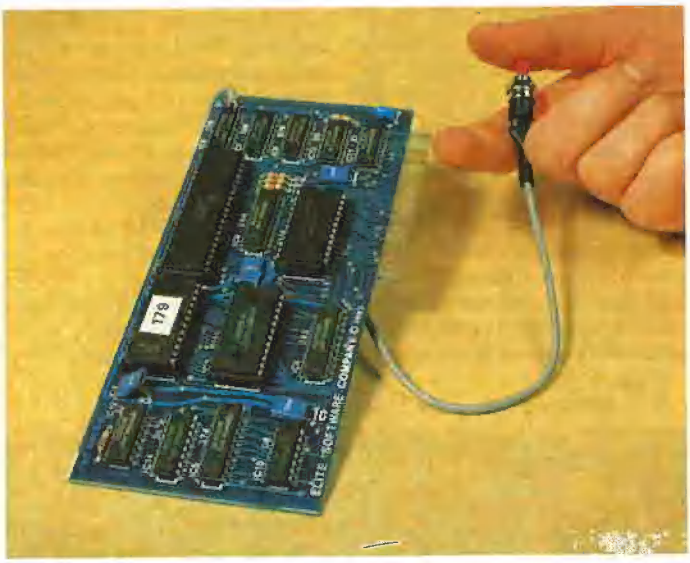
**Hard-Kopie-Methode:** Dieses Verfahren beruht darauf, daß man ein *spezielles* Diskettenkopiergerät benutzt. Im WDR-Fernsehen konnte gezeigt werden, daß sich mit einem solchen Spezialgerät praktisch alles „herüberholen“ läßt.

**Softbreaker-Methode:** Dieses von mir entwickelte Verfahren ist eine kombinierte Memory-Management- und Reset-Methode speziell für den Apple IIe und funktioniert theoretisch nur bei Apple-II-Plus-Programmen, die auf dem Apple IIe lauffähig sind.

**Hardbreaker-Methode:** Dieses Verfahren basiert auf einer *speziellen* Hardware-Karte mit eigenem Prozessor und eigenem RAM sowie der Fähigkeit, „unbemerkte“ Interrupts durchzuführen. Hierzu gehört z.B. die Wildcard Plus.

Nachfolgend beschränken wir uns auf meinen SOFTBREAKER sowie auf die WILDCARD, die beide gemeinsam haben, daß sie den momentanen Apple-RAM-Inhalt nach einer Reset/Interrupt-Unterbrechung auf einer Datendiskette auf Track-Sektor-Ebene sichern und später wieder einladen können. Der technisch interessierte Leser sei darauf hingewiesen, daß der Softbreaker die 64K ab Spur 34 rückwärts auf der Diskette speichert, während die Wildcard Plus die 64K bzw 128K spurmäßig aufsteigend auf der Diskette ablegt. Die „unteren“ 64K liegen vor und die „oberen“ 64K hinter der Catalog-Spur.

*Wildcard Plus*



**Softbreaker**

Beim Apple IIe muß man zwischen dem 64K-Motherboard und der 64K-Karte unterscheiden, die beide in denselben Adreßbereich „gemappt“ sind. Als ich vor anderthalb Jahren meinen alten Apple II Plus durch den Apple IIe ersetzte, kam ich auf den Gedanken, daß Programme eigentlich auch direkt von der 64K-Karte bootbar sein müßten. Dies ist tatsächlich möglich, und zwar erstens bei allen alten Apple-II-Plus-Programmen, die von der 64K-Karte „nichts wissen“, sowie darüber hinaus bei denjenigen neueren Apple-IIe-Programmen, die die 64K-Karte nicht benutzen. Da ein Hardware-Reset durch Drücken der Ctrl-Reset-Tastenkombination stets (wieder) das 64K-Motherboard aktiviert, läßt sich ein von der 64K-Karte gebootetes Programm immer unterbrechen. Ein Rücksprung auf die 64K-Karte ist über deren Stack möglich, weshalb der Softbreaker eigentlich ein „Stackjumper“-Programm ist, weil der Rücksprung stets über den 64K-Karte-Stack erfolgt. Da der Softbreaker von mir selbst stammt, zähle ich die Vor- und Nachteile so nüchtern wie möglich auf:

*Nachteile:* (1) Es können normalerweise nur die o.g. Altprogramme gesichert werden. (2) Eine Reset-Unterbrechnung ist nur an einer „geeigneten Stelle“ möglich, z.B. im Hauptmenü des „gebreakten“ Programms. (3) Prozessor-Register-Werte und Softswitches müssen notfalls manuell eingesetzt werden, was lästig und zudem bei der 80-Zeichenkarte nicht möglich ist. (Das letztere Problem hat auch die Wildcard auf dem Apple II Plus.) Bei 40-Z/Z-Applesoft-Programmen funktioniert der Softbreaker immer. (4) Der Softbreaker läuft nicht auf dem Apple IIc, weil der IIc einen Teil der 64K-Karte beim Booten löscht.

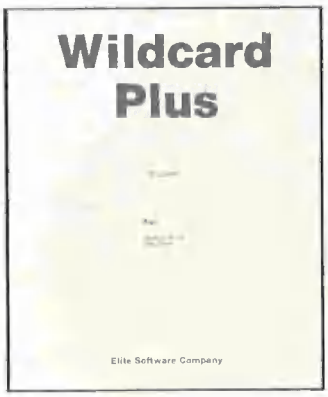
*Vorteile:* (1) Es ist keine besondere Zusatzkarte erforderlich. (2) Das Programm kostet nur DM 48,-.

Der Softbreaker ist somit das „Breaker“-Programm des armen Mannes, das zwar nicht immer, aber doch in vielen Fällen funktioniert.

(Bezugsquelle: Über unseren Hüthig Software Service oder über einschlägige Apple-Händler)

**Wildcard Plus**

Die neue Wildcard Plus (siehe **Bild**), die mir ein Freund zum Testen ausgeliehen hatte, ist dem Softbreaker weit überlegen, kostet dafür aber auch je nach Händler ca. DM 560,-. Nachdem man die Wildcard in einen freien Slot gesteckt




hat, läßt sich (wahrscheinlich) jedes RAM-residente Programm mit dem Interrupt-Auslöser (ein kleiner Druckknopf) unterbrechen. Die Wildcard Plus läuft zwar auch auf dem Apple II Plus, ist jedoch speziell auf den Apple IIe angepaßt und kann die kompletten 128K sichern. Anstelle von geschützten Programmen habe ich die Wildcard Plus mit Hilfe von zwei eigenen trickreichen Testprogrammen vergeblich „aufs Kreuz legen" wollen. Diese Programme WILDCARD-.TEST1 und WILDCARD.TEST2 können aus Platzgründen hier nicht abgedruckt werden und befinden sich auf der Peeker-Sammeldisk #4. Der WILDCARD-.TEST1 ergab im einzelnen folgendes:

1. Der Interrupt verändert nicht den Stack, weil die Wildcard eigenes RAM hat, wozu offenbar auch ein eigener Stack gehört.
2. Alle Prozessor-Register einschließlich des Statusregisters werden nicht verändert. Es wäre also zwecklos, das Interrupt-Bit zu setzen oder zurückzusetzen.
3. Alle Softswitch-Zustände werden (beim Apple IIe, nicht so beim Apple II Plus!) nach dem Interrupt automatisch „gerettet". Ich habe durch Modifikationen von WILDCARD.TEST1 die exotischsten Kombinationen ausprobiert. Die Wildcard Plus ließ sich einfach nicht „austricksen". Beispielsweise befand sich eines der Testprogramme in der Language Card, die HGR-Grafik war aktiv und der Motor des Disk-II-Laufwerks war eingeschaltet. Durch den Interrupt wurde nichts verändert: Die LC war weiterhin aktiv, der Stack wie gehabt, der Motor noch eingeschaltet usw.
4. Die Wildcard Plus, die in jeden beliebigen Slot gesteckt werden kann, hat weder Slot-RAM noch Slot-ROM. Folglich kann ein geschütztes Programm (wahrscheinlich) niemals ermitteln, ob die Karte existent ist oder nicht.

Die Wildcard Plus ist damit die ideale Interrupt-Karte des reichen Mannes. Allerdings gibt es – abgesehen vom Preis – auch einige technische Nachteile:

a) Die Karte hat einen eigenen 6502-Prozessor, läuft also nicht mit 65C02-Programmen.
b) Ferner schließt die Wildcard Plus jede andere Prozessor-Karte aus. Beispielsweise funktioniert sie nicht in Verbindung mit der Accelerator-Karte.
c) Es kann wie beim Softbreaker nur der momentane RAM-Inhalt gesichert werden. Damit ist die Karte bei Programmpaketen, die aus mehreren Modulen bestehen, nur begrenzt einsetzbar.
d) Man muß für die Wildcard Plus einen Slot reservieren, was dann von Nachteil ist, wenn alle Slots bereits bestückt sind. Sinngemäß funktioniert die Wildcard natürlich nicht auf dem Apple IIc, denn dieser hat gar keine Slots. Aufgrund einer cleveren Bootroutine ist jedoch ein auf der Diskette gesichertes Programm auch *ohne* die Karte korrekt bootfähig, wenn dieses Programm in den unteren 48K mindestens 1024 Bytes während des gesamten Ladevorgangs nicht benutzt. Dies wurde experimentell mit dem WILDCARD.TEST2 ermittelt.

Abschließend sei darauf hingewiesen, daß sich die Wildcard Plus auch zum Debuggen von eigenen Programmen eignet, da man sich nach dem Interrupt die Registerinhalte sowie den Program Counter ansehen kann.

(Bezugsquelle: Firma Karl-Heinz Weiß, Am Wiesenhof 17, 2940 Wilhelmshaven, Tel. 04421/83179)

# Lieber gescheite Sprachen als dumme Sprüche

(Aufgepeekter Microsoft-Werbeslogan)


*Spiele-Auswahl (Fa. Lucius)*

## Beliebte Apple-Spiele

Die Firma Klaus-Peter Lucius, Theodor-Körner-Straße 5, 4220 Dinslaken 1, Tel. 02134/52782, die neben Anwenderprogrammen und Utilities über ein großes Sortiment an Apple-Spielprogrammen verfügt, hat uns freundlicherweise eine Auswahl der meistbegehrten unter den „klassischen" Spielen mit Kurzcharakteristika zur Verfügung gestellt. Es handelt sich also hier nicht etwa um Testberichte, denn Spiele sind ohnehin mehr Geschmacksache. Im nächsten Peeker-Heft bringen wir dann eine Auswahl der neuesten Spiele aus den USA. Bitte geben Sie bei Ihrer Preisanfrage oder Bestellung den Gerätetyp an, da wir feststellen mußten, daß einige Spiele beispielsweise nicht auf dem IIc laufen. Ich hoffe, daß es Ihnen, lieber Leser, nicht so ergeht wie mir. Ich habe nämlich leider nie Zeit für Spiele.

us

**Cyclod**
Der einäugige Cyclod muß sich durch ein Labyrinth kämpfen, in dem allerlei Gefahren auf ihn lauern. Schlangen, Monster und andere Fallen machen ihm das Leben schwer. Helfen Sie ihm, aus diesem Labyrinth zu entkommen. Mauern Sie die Schlangen in ihrer Grube ein oder werfen Sie mit Steinen nach ihnen. 1 Spieler, 20 Levels, Sound, Keyboard, Paddles, Joystick.

**Fly Wars**
Die von Ihnen gesteuerte Spinne hat die Aufgabe, Fliegen zu fangen. Das geschickt aufgebaute Spinnennetz wird vielen Fliegen zum Verhängnis. Allerdings sollten Sie sich vor der gelegentlich auftauchenden Giftspritze hüten. Denn wenn Sie in diese Giftwolke geraten, ist Ihr Spiel beendet. 1 Spieler, 8 Levels, Sound, Keyboard, Paddles, Joystick.

**Minotaur**
Dieses Labyrinthspiel basiert auf der griechischen Mythologie. Minotaur, halb Mensch, halb Stier, ist in einem Labyrinth gefangen und versucht zu fliehen. Dabei stellen sich ihm mehr als zwanzig verschiedene Gestalten in den Weg. Unterstützen Sie Minotaur in diesem interessanten Hires-Spiel. 1 Spieler, 32 Levels, Sound, Keyboard, Paddles, Joystick.

**Outpost**
Als Kapitän einer großen Raumstation müssen Sie Ihr Schiff gegen feindliche Raumschiffe schützen. Neben vier großen Laserkanonen stehen Ihnen auch die entsprechenden Schutzschilder zur Verfügung. Die Möglichkeit des Gegners, aus verschiedenen Richtungen anzugreifen, macht dieses Spiel besonders abwechslungsreich. 1 + 2 Spieler, 8 Levels, Sound, Keyboard, digitaler Joystick.

**Pulsar**
Sie haben die Aufgabe, eine feindliche Raumstation, die durch mehrere, rotierende Schutzschilder gesichert ist, zu zerstören. Im zweiten Spiel müssen Sie durch Lükken im System zur Befehlszentrale gelangen. Die besonders schwierige Kombination aus beiden Aufgaben ist als dritte Variante vorgesehen. 1 Spieler, 8 Levels, Sound, Keyboard.

**Snake Byte**
In einem Labyrinth sind Äpfel verteilt, die von Ihrer Schlange gefressen werden sollen. Mit jedem Apfel wird die Schlange länger und das Spiel schwieriger. Sobald alle Äpfel gefressen sind, landet die Schlange im nächsten Labyrinth. Versuchen Sie einmal, in das achtundzwanzigste Labyrinth vorzudringen. 1 Spieler, 28 Levels, Sound, Keyboard, digitaler Joystick.

**Wavy Navy**
In diesem sehr bewegten Spiel wehren Sie sich gegen Flugzeuge, Bomben, Hubschrauber, Treibminen, U-Boote und andere Gefahren. Dabei müssen Sie gleichzeitig beobachten, was über Ihnen und unter Ihnen passiert. Und das alles bei einem unwahrscheinlich hohen Seegang, der das Spiel besonders gefährlich macht. 1-4 Spieler, 3 Levels, Sound, Keyboard, Paddles, Joystick.

**Federation**
Helfen Sie mit Ihrem Raumschiff, das friedliche Weltreich zu erhalten und schützen Sie die Erde vor unmenschlichen Feinden. In der Nähe von DENEB VII treffen Sie auf gegnerische Raumschiffe, die mit ihren intelligenten Waffensystemen versuchen, Sie an der Ausführung Ihres Auftrages zu hindern. 1 Spieler, 3 Levels, Sound, Keyboard.

**Reversal**
Das klassische Brettspiel in Computerversion. Sie spielen gegen den Computer und müssen versuchen, durch kluges Setzen Ihrer Spielsteine eine möglichst große Spielfläche zu belegen. Der aktuelle Spielstand wird u.a. duch fröhliche oder traurige Gesicher auf den Spielsteinen dargestellt. 1 + 2 Spieler, 27 Levels, Sound, Keyboard.

**Juggler**
Im Zirkus werden dem Jongleur verschiedene Gegenstände zugeworfen, die er möglichst lange in der Luft halten muß. Geschicktes Hantieren mit vielen Gegenständen ergibt eine hohe Punktzahl. Je nach Geschicklichkeit können sieben Spielstufen erreicht werden, die den Jongleur in Bewegung halten. 1 Spieler, 7 Levels, Sound, Keyboard, Paddles, Joystick.

**Shuffleboard**
Eigentlich wird Shuffelboard während einer Kreuzfahrt auf dem Sonnendeck des Passagierdampfers gespielt. Dieses mit Billard verwandte Spiel können Sie nun auch zu Hause gegen den Computer oder einen anderen Shuffler spielen. Nach internationalen Regeln müssen Sie Ihren Puck im Ziel plazieren. 1 + 2 Spieler, 2 Level, Sound, Keyboard.

**The Cube Solution**
Rubik's Cube, der Zauberwürfel, kann nur per Computer gelöst werden. Überlassen Sie Ihrem Apple das Problem. Jeder einzelne Schritt wird in der Perspektive farbig auf dem Bildschirm dargestellt. Beliebige Aufgabenstellungen und automatische Lösungen sind schnell und einfach abzurufen. 1 Spieler, Keyboard.

**Roadblock**
Als Sheriff einer Kleinstadt müssen Sie Ihre vier Polizeiwagen so geschickt plazieren, daß die Bankräuber, die gerade eine Bank überfallen haben, nicht entwischen können. Diese Aufgabe ist gar nicht so einfach zu lösen, obwohl Ihnen auf dem farbigen Stadtplan die Positionen von Räuber und Gendarm angezeigt werden. 1 Spieler, 3 Level, Sound, Keyboard.

**Escape**
Sie versuchen, aus dem Staatsgefängnis zu fliehen. Dabei dürfen Sie sich natürlich nicht von den Wachen, die überall auftauchen, erwischen lassen. Der elektrische Zaun hat nur zu ganz bestimmten Zeiten einige Lücken, die Sie nutzen können. Das Spiel gegen die Zeit erfordert viel Überlegung und Aufmerksamkeit. 1 Spieler, Keyboard.

**Flight Simulator**
Diese Real-Time-Simulation eines einmotorigen Sportflugzeugs bietet auf dem Hires-Bildschirm nicht nur alle Instrumente, die für den Blindflug vorgeschrieben sind, sondern stellt auch bewegte Landschaft, Seen, Flüsse, Straßen, einige Gebäude und über 80 verschiedene Flugplätze zu jeder Tages- und Nachtzeit dar. Wind- und Wetterlage, Sichtverhältnisse, und viele andere Flugparameter sind frei programmierbar. In vier Flugkarten sind alle erforderlichen Angaben für die Funknavigation eingetragen. Die 180seitige Dokumentation macht neben der Flugphysik auch mit dem Programm und den Kunstflugmöglichkeiten vertraut. Einige Flugschulen und Flugzeughersteller nutzen dieses Programm bereits für Anfänger und Fortgeschrittene. (II+, IIc, IIe).

**Sargon III**
Etwa 680 verschiedene Eröffnungen sind auf der Programm-Diskette gespeichert. Neun Spielstärken und ca. 45 Schachprobleme können frei gewählt werden. Über 100 klassische Schachpartien können geladen und nachgespielt werden. Neu ist die Möglichkeit, das Spielfeld und alle gemachten Züge auszudrucken und Spiele auf der Diskette zu speichern. Das 80seitige Handbuch bringt neben einigen Grundregeln auch alles Wissenwerte über das Programm. In einem eigenen Kapitel sind insgesamt 22 „Special Features" erklärt, die das Programm interessant und vielseitig machen. (II+, IIc, IIe, mac).

**Summer Games**
Dieses Hires-Action-Programm bringt die Olympischen Spiele ins Wohnzimmer. Nach einer Eröffnungszeremonie können die Spieler zwischen achtzehn Nationen wählen, für die sie antreten wollen. In den meisten olympischen Disziplinen kann man gegen den Computer oder gegen andere Mitspieler antreten. Schwimmen, Leichtathletik, Turnen und viele andere Sportarten werden durch den Joystick oder die Tastatur gesteuert. Alle Ergebnisse werden nach internationalen Richtlinien ausgewertet und nach jedem Wettkampf angezeigt. Neben den Wettkämpfen können auch einzelne Disziplinen ausgewählt werden, falls der Spieler in der einen oder anderen Sportart noch etwas trainieren möchte. Die schnelle, hochauflösende Farbgrafik macht Summer Games zu einem Programm mit hohem Spielwert. (II+, IIc, IIe).

**Der nächste Peeker**
**Heft 4/1985**
**erscheint am**
**25.03.85**


*Memdos-Karte*

zu verknüpfen oder auszugeben. Bis zu 10 Schlüssel, jeder bis zu 255 Zeichen lang und sogar verschachtelt, können für den indexsequentiellen Filezugriff benutzt werden. Die einzelnen Datensätze können unterschiedlich lang sein, wodurch ein platzsparender Umgang mit dem Speichermedium gewährleistet ist. Zur einfacheren Datenerfassung ist ein Maskengenerator eingebaut, der durch direkte Eingabe auf den Bildschirm gesteuert wird.
Eine wesentliche Erweiterung zum Basic ist die Möglichkeit, Unterprogramme mit „DEF FN" zu definieren und mit einem „CALL FN" aufzurufen. Dabei können Sie gezielt Variablen aus dem Hauptprogramm ins Unterprogramm übergeben. Die Verwendung von globalen und lokalen Variablen erinnert etwas an Pascal. So erreichen Sie mit Memdos ein strukturierteres Basic als mit Applesoft allein. Trotzdem müssen sie immer noch viel programmieren, um an die Leistungsfähigkeit kommerzieller Datenbanken (wie z.B. dBase II) auch nur annähernd heranzukommen. Bei kleineren Anwendungen sind Sie jedoch schnell am Ziel, wenn es z.B. nur Adressenlisten sein sollen.
Bezugsquelle: Memsoft GmbH, Dreieichstraße 59, 6000 Frankfurt 70, Tel. 069/623083

## Memdos Junior

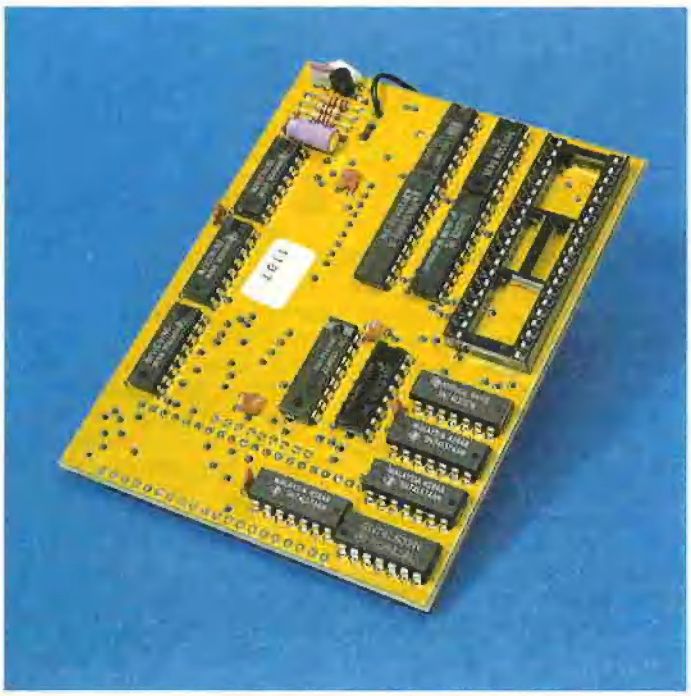
**Ein neues Betriebssystem für den Apple II**
**getestet von Dr. Jürgen B. Kehrel**

Programme aus Frankreich sind bei uns bisher recht unbekannt. Eine Firma versucht das nun zu ändern: Memsoft S.A. aus Nizza. Zur Eroberung des deutschen Marktes setzt sie auf Memdos, das es in einer vollständigen Version und einer Testversion Memdos Junior gibt. Wir haben uns letztere angesehen.
Memdos Junior kostet DM 59,- und wird auf einer Diskette geliefert. Das deutsche Handbuch umfaßt 90 Seiten und hat mindestens ebensoviele Fehler wie Seiten, ganz abgesehen von einigen lustigen Wendungen wie „Pisten-Zuweisung" (S. 84) für „track allocation" (Spurzuordnung). Einiges wird der Benutzer schon zweimal lesen müssen, um hinter die Geheimnisse von Memdos zu kommen. Wie wirbt Memsoft doch so schön: „Auf ihre Bedarfe antworten ist eine einfache Sache, indem man mit dem richtigen Werkzeug eine Anwendunvg erschafft." Aber auf der Diskette sind viele Beispielprogramme, die Sie durcharbeiten sollten, um einen richtigen Eindruck von den Fähigkeiten des Programms zu bekommen.

Memdos ersetzt als Betriebssystem DOS 3.3. In der vollen Version ist es in der Lage, Festplatten mit bis zu 120M zu verwalten. Es ist mehrplatzfähig für bis zu 16 Rechner, die über serielle Schnittstellen verbunden werden. In der Junior-Version ist nicht viel davon zu merken: Sie simuliert im Hauptspeicher des Apple eine RAM-Floppy mit 28K, die sich Daten und Programme teilen müssen. Da sind Sie schnell an den Grenzen angelangt. Im Handbuch heißt es auf Seite 1 dann auch richtig: „Allerdings bleibt seine Benutzung auf den pädagogischen Zweck beschränkt." Andere Firmen nennen so etwas offen eine Demo-Version, und es stände auch Memsoft gut an, in ihren Anzeigen darauf hinzuweisen. Immerhin bekommen Sie die DM 59,- auf den Kaufpreis des vollen Systems angerechnet (DM 349,- Diskettenversion, DM 499,- Kartenversion, die keinen Platz im Hauptspeicher belegt). Wenn Sie nicht „auf den Geschmack" kommen, war Ihre Ausgabe allerdings recht nutzlos.

Der zweite Teil von Memdos ist ein Verwaltungssystem für Basicdateien. Dazu besitzt es sehr leistungsfähige Makrobefehle, um Dateien anzulegen, zu lesen, zu sortieren,

*CP/M-Karte*



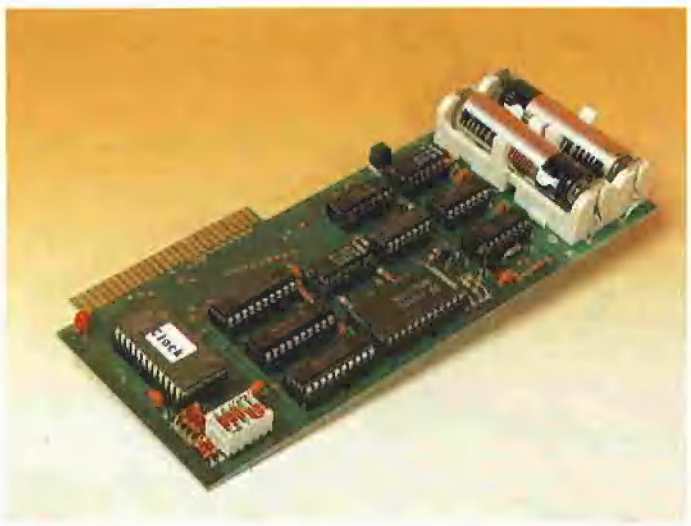
## CP/M-Karte für den Apple IIc

**getestet von Dr. Jürgen B. Kehrel**

Eigentlich hatte die Firma Apple ihren neuen Rechner IIc so ausgelegt, daß keine Zusatzprozessoren benutzt werden können, denn dem IIc fehlen die Erweiterungssteckplätze. Deshalb mußte bisher die Frage, ob auch Wordstar, dBase II, Multiplan oder Turbo-Pascal auf dem IIc laufen, mit nein beantwortet werden. Das ist jetzt anders, denn inzwischen wurde eine Methode entwickelt, wie doch ein Z80-Prozessor anzuschließen ist.
Ganz so leicht und billig wie beim Apple IIe ist die Umrüstung zwar nicht, aber immerhin haben Sie jetzt die Möglichkeit, aus Ihrem IIc eine CP/M-Maschine zu machen. Mir lag das IIc-CP/M-Modul (siehe **Bild**) der amerikanischen Firma CIRTECH vor, das in Deutschland von Computerware in Frankfurt vertrieben wird.
Da der Endbenutzer den IIc nicht öffnen darf, ohne seine Garantie zu verlieren, muß der Umbau von einem autorisierten Händler vorgenommen werden. Das ist aber keine große Aktion: Nach dem Öffnen des Gehäuses wird die Tastatur

beiseite gelegt. Unter ihr befindet sich der 65C02-Prozessor des Apple, der aus seiner Fassung gezogen und auf die CP/M-Zusatzplatine gesteckt wird. Diese enthält den Z80-Prozessor und einige Steuerbausteine. Alles zusammen wird wieder zurück auf die Appleplatine gesteckt, wo erstaunlicherweise noch genügend Raum zwischen ICs und Tastatur vorhanden ist. Nur die IC-Fassung und die Steckbeinchen können Sie beim Einsetzen nicht sehen, also Vorsicht! Ein Draht wird zum Schluß noch an den TMG-Baustein geklemmt. Zuvor muß allerdings das Laufwerk abgeschraubt werden.
Auf diese wenigen Handgriffe beläuft sich der ganze Umbau, der ebensoleicht wieder rückgängig zu machen ist.

Ein kurzer Test mit CP/M 2.23 60K von Microsoft zeigte, daß alle gewohnten Funktionen arbeiten. Auch der Drucker wird über die serielle Schnittstelle richtig angesprochen, ohne daß das CP/M umkonfiguriert werden mußte. Zur Testversion des Autors gehörte ein RAM-Disk-Treiber für die zweite 64K-Bank, jedoch kein CP/M-System und kein Handbuch. Diese beiden müssen Sie getrennt erwerben. Die endgültigen Preise standen Anfang Januar noch nicht fest, aber mit rund DM 650,- für die Hardware *ohne* CP/M-System müssen Sie zur Zeit wohl noch rechnen.
Bezugsquelle: Computerware GmbH, Wilhelm-Leuschner-Str. 34, 6000 Frankfurt/Main, Tel. 069/ 236713



*HOCO-Masterclock*

# HOCO-Masterclock

**getestet von Arne Schäpers**

Diese Uhrenkarte kommt zusammen mit einem kleinen Manual und einer Begleitdiskette mit einigen Demoprogrammen. Von der Karte gibt es laut Hersteller zwei Versionen, die sich nur hinsichtlich der Stromversorgung unterscheiden, einmal mit zwei Monozellen, also mit normalen Batterien, und zum anderen mit zwei NiCd-Akkus (optional). Ich hatte die Version mit normalen Batterien und deshalb bereits beim Auspacken Bedenken, wo sich diese Karte wohl unterbringen läßt. Beim Einsetzen zeigte sich dann leider auch, daß die Karte durch die Batterien eine solche Dicke bekommt, daß sie entweder einen zweiten Slot blokkiert oder in Slot 7 eingesetzt werden muß. Nun, es wird nicht jeder einen Apple haben, bei dem alle Slots „vollgestopft" sind. Trotzdem ist das – speziell für Benutzer eines PAL-Interfaces, bei denen Slot 7 belegt ist – etwas ärgerlich, und es bleibt zu hoffen, daß die Version mit Akkus bald und ohne allzu hohen Aufpreis verfügbar ist. Nach dieser Randbemerkung aber nun zu den Eigenschaften der Karte selbst.
*Preis:* ca. 330,- DM, also ausgesprochen günstig im Vergleich zu der „offiziellen" Uhrenkarte (Thunderclock).

*Hardware:* Die Uhrenkarte funktioniert in jedem Slot (außer Slot 0). Auf der Karte befinden sich 5 DIL-Schalter, mit denen die Karte für das entsprechende Betriebssystem konfiguriert werden muß. Diese Schalterstellungen sind softwaremäßig nicht änderbar. Die zeitliche Auflösung beträgt eine Millisekunde, der Kalender ist (inklusive Schaltjahre) mit auf dem Uhrenchip vorprogrammiert. Die Gangabweichnung lag bei meinem Exemplar bei einer Minute im Monat.
*„Paper"ware:* Es werden 15 Seiten Informationen mitgeliefert, die die diversen Konfigurationsmöglichkeiten beschreiben. Dieses Manual besteht aus photokopierten Blättern und ist einigermaßen übersichtlich gestaltet.
*Software:* Hier ist (wieder einmal) ein Hoch auf ProDOS fällig. Die entsprechenden DIL-Schalter werden gesetzt, die Karte wird eingesteckt und nach dem Einstellen der Uhrzeit kann man sich von nun an über CREATION/MODIFIKATION DATE/TIME im Directory freuen und die Karte vollständig vergessen.
Für die anderen Betriebssysteme auf dem Apple sind die üblichen Verrenkungen nötig. Allerdings hat HOCO hier mit reichlich Software vorgesorgt, die sich fast vollständig auf der Karte in einem EPROM des Typs 2732 (!) befindet.

Unter DOS 3.3 sind folgende Betriebsarten möglich:

0 – „Screen"-Interrupt 1: In der rechten oberen Ecke des Bildschirms wird die Zeit fortlaufend angezeigt, bei einem IIe auch bei 80 Zeichen.

1 – „TI$"-Interrupt: Ein in Applesoft definierter String wird im Sekundentakt verändert und enthält Stunde/Minute/Sekunde.

2 – „Screen"-Interrupt 2: Dieser ist identisch mit Interrupt 1; die Anzeige erfolgt in der unteren rechten Ecke des Bildschirms.

3 – „HOCO-Format": Ein eigenes Format, auf das die Demoprogramme auf der mitgelieferten Diskette abgestimmt sind.

4 – Emulation der Uhrenkarte von California Computer Systems.

5 – Emulation der Mountain Hardware Clock.

Für Pascal wird eine UNIT auf der Demodisk mitgeliefert, die in die SYSTEM.LIBRARY eingebunden wird und danach mit „USES UHR" von Anwenderprogrammen aus angesprochen werden kann. Des weiteren wird ein STARTUP mitgeliefert, mit dem die Uhrzeit gesetzt werden kann.

Für CP/M 2.2 existiert keine Software von HOCO. Man hätte hier eventuell an etwas ähnliches wie TI$ für MBASIC denken können, das Betriebssystem macht hier aber ziemliche Schwierigkeiten, weil ein direktes Ansprechen der Slots auch von MBASIC aus nicht möglich ist (bestenfalls via „UHR = &HC100: CALL %UHR") und Interrupts, bedingt durch die Umschalterei zwischen 6502 und Z80, zum Systemabsturz führen. Das passiert leider auch, wenn man unter DOS einen „Screen"-Interrupt aktiviert hat und vor dem Booten von CP/M vergißt, diesen wieder abzuschalten.

Für CP/M 3.0 hat HOCO entsprechende Software angekündigt.

Zusammenfassend läßt sich sagen, daß die HOCO-Masterclock wohl ihr Geld wert ist, wenn man noch genügend Platz in seinem Apple oder die Akku-Version hat. Zu meiner „Traum"-Uhr gehört allerdings eine Konfigurierbarkeit per Programm, damit man nicht jedesmal das Rechnergehäuse aufmachen muß, wenn man das Betriebssystem wechselt.

Bezugsquelle: HOCO Homecomputer Vertriebs-GmbH, Flügelstraße 47, 4000 Düsseldorf, Tel 0211/ 776270

# Leserbriefe

**1-Drive-Kopierprogramm**
Hiermit möchte ich Ihr Angebot in dem Artikel „Quickcopy", Jan/Feb '85 Seite 22, annehmen und Sie bitten, mir ein Listing für das 1-Drive-Kopierprogramm zu übersenden. Außerordentlich gut finde ich Ihre Programme für 80-Spur-TEAC-Laufwerke (z.B. 55 F). *Jürgen Schulz, Bremen*
Kommt im nächsten Heft. Anm. der Red.

**Assemblereinführung**
Können Sie eine Einführungsreihe über Maschinensprache abdrukken? *Harald Lux, Bonn*
Kommt als 16seitiger Sonderteil im übernächsten Heft. Anm. der Red.

**Turbo-Pascal**
Von den ersten beiden Ausgaben des „Peekers" bin ich begeistert. Sie könnten jedoch meinen Enthusiasmus noch steigern, wenn Sie sich auch einmal mit dem Turbo-Pascal befassen würden, das wohl immer mehr Verbreitung findet und mir wegen seiner Schnelligkeit, numerischen Genauigkeit und leichten Handhabung sehr gut gefällt. *Albert Stimpel, Kassel*
Zwei weitere Beiträge sind in Arbeit. Anm. der Red.

**64K-Karte-Softswitches**
Mit großem Interesse habe ich in Ihrer letzten Ausgabe den Artikel Double-Hires-Grafik verfolgt, der für mich zum ersten Mal eine verständliche Information über die Grafikmöglichkeiten der erweiterten 80-Zeichenkarte darstellte. Mir ist jedoch nicht ganz klar, wie man von einem Applesoft-Programm im Text-Modus schreibend und lesend auf die zusätzlichen 64K der Karte zugreifen kann. Ich wäre Ihnen dankbar, wenn Sie in Ihrer nächsten Ausgabe diese Schalterkombinationen veröffentlichen würden. *Friedhelm Klein, Bochum*
Beachten Sie die diversen RAM-Disk-Peeker-Beiträge. Außerdem ist eine gesonderte Softswitch-Tabelle unter Berücksichtigung des IIc in Arbeit. Anm. der Red.

**Lobeshymne**
Ich möchte meiner Freude Ausdruck geben, daß Sie das Unternehmen „Peeker" begonnen haben. Mir ist egal, wie oft Ihr Name als Autor oder Anbieter erscheint, solange Sie das nur (hoffentlich) durchhalten. Das ist einmal eine deutsche Zeitschrift mit Substanz, was Neues nicht nur auf dem Markt der Apple-Spezialisten. Bei letzteren bin ich nicht einmal sicher, daß „Nibble" mit Ihnen konkurieren kann, oder sonstwer. Auch ich habe die A.U.G.E. entsetzt wieder verlassen und kann mich nur wundern, wie lange Sie Ihre Fähigkeiten dort hergeschenkt haben. Nachdem nun auch Call-A.P.P.L.E. zu einem Regenbogenblättchen entartet ist (woran selbst Autor Kashmarek nichts ändern kann), liefern Sie hier in Deutschland einen so herrlichen Ersatz. Der Drang nach Sauberkeit der Sprache, der gerade im neuen Heft 1/2 mehrfach anklingt, erfreut mich besonders. Früher mußte man sich gegen die Kulturapostel zur Wehr setzen, nach deren Meinung Technik-Ignoranz allein schon zum Kulturträger qualifizierte. Heute gibt es leider umgekehrt den Elektroniker, der den Lötkolben aus der Hand gelegt hat und nun Artikel schreibt in der Annahme, daß die deutsche Sprache mit ihm in die Welt gekommen sei. Beispiel: Das seit Jahrhunderten übliche „edieren", englisch „to edit", heißt nun rückübersetzt auf einmal „editieren", nur weil die Proponenten der deutschen Fachsprache in der Schule nicht aufgepaßt haben. Solche Dinge werden sich allerdings nicht mehr ändern lassen. *Dr.-Ing. Hansjochen Benda, Ismaning*

**Z80H bei Prometric?**
Bezüglich der im Beitrag über Prometric, in Heft 2/84, Seite 64 erwähnten Z80H-Zusatzkarte: Eine solche Karte wird es vorerst nicht geben, wie eine Anfrage bei E. Böhmer, Dreiech ergab, da es augenblicklich nicht zu überwindende Timing-Probleme gegeben hätte. Man wollte eine neue Z80-Version abwarten und es nochmals versuchen. *Martin Lukas, Eschelbronn*

**Testbericht „Exbasic Level II"**
Ende November '84 habe auch ich die Basic-Erweiterung „Exbasic Level II" für den Apple IIe (80 Zeichen) direkt bei der Vertriebsfirma bestellt, die auch prompt geliefert wurde. Daher habe ich mit großem Interesse Ihren kritischen – aber auch objektiven – Testbericht gelesen, zumal mir auch der im Apple's Magazin veröffentlichte „Test" bekannt war. Von den positiven Aspekten der Software konnte ich mich bisher leider nicht überzeugen, da der Computer nach der ordnungsgemäßen Installation der Platine nicht funktionierte. Grund war ein Kurzschluß auf der Platine, der mich zu einer Reklamation veranlaßte. Dagegen muß ich allen von Ihnen erwähnten negativen Punkten in vollem Umfang zustimmen, obwohl zum Zeitpunkt der Bestellung die seitenverkehrte Platinen-Konstruktion von mir noch toleriert wurde. Anstatt schnellsten eine neue Platine zu liefern, wurde von der Vertriebsfirma auf rechtlich nicht haltbare Haftungsausschlüsse – wie z.B. wegen einer nicht auszuschließenden Computer-Beschädigung –, Lieferbedingungen, Eigentumsvorbehalte usw. verwiesen. Angeblich war ich auch der erste Kunde, der mit dem dürftigen Handbuch nicht zufrieden war. Die ganze Behandlung des Sachverhalts ließ eine Gleichstellung eines ehrlichen, aber unvorsichtigen, Käufers mit einem Software-Kopierer vermuten. Schließlich sah ich mich gezwungen, die defekte Platine zurückzusenden und den Kaufvertrag, entsprechend den bei Mängeln gültigen gesetzlichen Regeln, rückgängig zu machen. Um nicht ähnliche negative Überraschungen zu erleben, wie sie auch in diesem Testbericht erwähnt wurden, sollte sich jeder potentielle Käufer von Exbasic Level II vor einer Bestellung davon überzeugen, ob das Produkt auch seinen Vorstellungen entspricht. *Gerhard Bletzer, Hirschberg*

**Apple in der Schule**
Mit großem Interesse habe ich Ihre Zeitschrift gelesen. Ich finde es hervorragend, daß sich Peeker – in der großen Flut der Computer-Zeitschriften – nur den Apple-Computern widmet. Die Informationen können dadurch spezifisch und wirklich hilfreich sein.

Wir haben erst vor einigen Wochen mit der Arbeit an Computern begonnen und besitzen vorerst nur einen Apple IIc mit Monitor und einem Drucker. Die vorhandene Software beschränkt sich im Augenblick nur auf Appleworks, mit dem wir auch erste Versuche in der Schulverwaltung machen. Da Apple-Computer gerade auch an Schulen sehr verbreitet sind, finden Sie sicherlich auch dort sehr viele interessierte Leser. Es wäre daher sehr wichtig, wenn Ihre Redaktion die Interessen und Bedürfnisse der Schulen einbeziehen würde. *Walter Gehrmann, Mainz-Gonsenheim*
Tun wir, siehe z.B. Legasthenie-Aufsatz in diesem Heft. Anm. der Red.

**MS-BASIC für Macintosh**
Es hat mich sehr gefreut, daß Sie in Ihrer Zeitschrift die Serie MS-BASIC für den Macintosh begonnen haben! Ich würde es sehr begrüßen, wenn Sie nach dem Abschluß dieser Serie eine entsprechende über Macintosh-PASCAL folgen lassen würden! Auch wenn der MAC-Anteil derzeit weniger als 1% aller Apple-Geräte ausmacht, meine ich, daß Sie in Ihrer vorzüglichen Zeitschrift ruhig mehr als 1% Anteil dem MAC widmen sollten. *Herbert Schmidt, Hamburg*

**Ohne „Freaks" geht nichts**
Ich habe das erste Mal Ihre Zeitschrift gelesen und muß sagen, daß sie sich in angenehmer Weise von anderen Puplikationen unterscheidet. Wobei ich auch das „apple's"-Magazin nicht ausnehmen will. Man könnte manchmal glauben, daß die meisten Zeitschriften schon Macintosh-verrückt sind. Wobei man annehmen könnte, daß die Herren von Apple druckreife Manuskripte gleich mitliefern.
Aber wie man in Ihrer letzten Ausgabe lesen kann, kochen die Leutchen auch nur mit Wasser (siehe Mecki). Im übrigen bin ich der Meinung, daß die allerorts hochgelobten Computerhersteller ohne die „Freaks" ganz schön aufgeschmissen wären. Ich kenne durch mein Hobby einige junge Leute, die sich mit traumwandlerischer Sicherheit in der Hard- und Software bewegen, so daß Angestellte, die heute noch glauben, das warme Wasser erfunden zu haben, sich Gedanken um ihren Arbeitsplatz machen sollten. Was aber wohl nicht der Fall ist, ich verweise wieder auf Mecki, und mit ganzer Kraft wird ein einmal erkorenes Konzept bis zum bitteren Ende als „das Gelbe vom Ei" dargestellt. Wohl dem, der eine gute Werbeabteilung hat. Und im übrigen schläft die Konkurenz ja auch nicht, von den Kompatiblen einmal ganz abgesehen. Ich hoffe auch weiterhin Positives von Ihnen zu lesen. *Friedhelm Groß, Wunstorf*

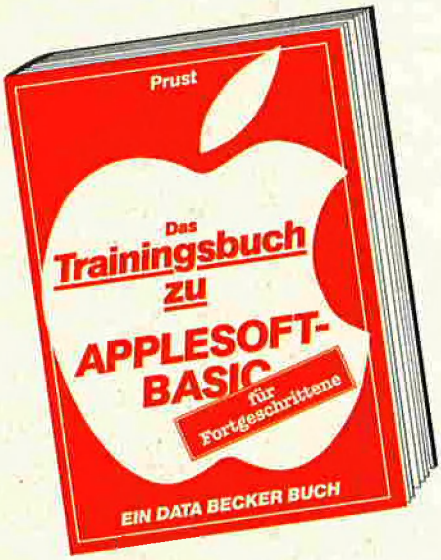
# Große Peeker-Umfrage

Haben Sie schon einmal die Hotline der Firma Apple in München schriftlich oder telefonisch (089 / 35 03 40) in Anspruch genommen?

Schreiben Sie uns, ob Ihre Probleme optimal gelöst wurden.
Wir werden Ihre Erfahrungen auswerten und darüber berichten.

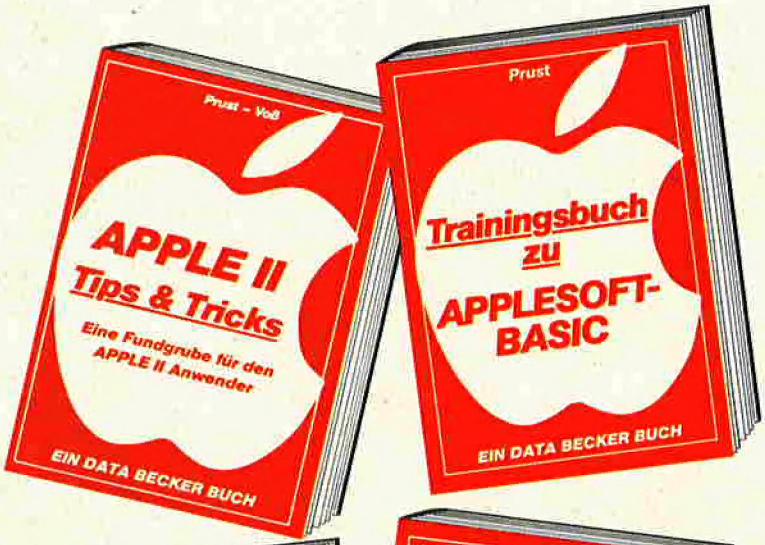
Ihre Peeker-Redaktion

# Neue APPLE Buchhits




Das TRAININGSBUCH ZU APPLESOFT-BASIC für FORTGESCHRITTENE ist besonders für denjenigen geschrieben worden, der schon mit der einfachen BASIC-Programmierung vertraut ist. Hier lernt man fortgeschrittenes Programmieren von der hochauflösenden Grafik bis zum direkten Speicherzugriff. Aus dem Inhalt: Gestaltung der Bildschirmausgabe, Stringoperationen, Hochauflösende Grafik, Systemmonitor, Formdefinitionen, EXEC-Files, Binärfiles, Speichergrenzen, Unterprogramme in Maschinensprache, Speichereinteilung. Ein Spitzenbuch! **Das Trainingsbuch zu APPLESOFT-BASIC für FORTGESCHRITTENE, 1984, 419 Seiten, nur DM 39,– (!)**


Ob Sie nun einen APPLE II+ oder einen APPLE IIc haben, in jedem Falle sollten Sie zur Diskettenprogrammierung und Dateiverwaltung mit dem großen FLOPPY-BUCH zum APPLE II arbeiten. Aus dem Inhalt: Eigenschaften und Anwendungen einer Floppy, unsortierte und sortierte sequentielle Dateien, Random-Dateien, Such- und Sortierverfahren, Index-sequentielle Dateien, weitere DOS-Kommandos und Besonderheiten, Erstellen eines Turnkey-Systems, CHAIN, Simulation des EOF-Befehls. Das Buch berücksichtigt nicht nur DOS 3.3, sondern auch das ProDos Betriebssystem. **Das Floppybuch zum APPLE II, 1985, ca. 250 Seiten, DM 49,–**




APPLE II TIPS & TRICKS bietet viele nützliche Informationen. Von der Menuetechnik und dem Mischen von Programmen bis zum direkten Speicherzugriff und Unterprogrammen in Maschinensprache, Hochauflösende Grafik, Sortieren, Speichereinteilung und vieles mehr. **APPLE II TIPS & TRICKS, 1984, 405 Seiten, DM 49,–**

Damit lernen Sie das APPLESOFT-BASIC und einen vernünftigen Programmierstil von Grund auf. Von der Beherrschung der Tastatur bis zum Arbeiten mit mehrdimensionalen Feldern. Für jeden, der solide und sicher in die Programmierung seines APPLE II, einsteigen will. **Trainingsbuch zu APPLESOFT-BASIC, 1984, 350 Seiten, DM 39,–**

So etwas haben Sie gesucht: Umfassendes Nachschlagewerk zum APPLE II und seiner Programmierung. Allgemeines Computerlexikon mit Fachwissen von A–Z und Fachwörterbuch mit Übersetzungen wichtiger englischer Fachbegriffe. Viele Abbildungen und Beispiele ergänzen den Text. **DATA BECKER LEXIKON ZUM APPLE II, ca. 400 S., DM 49,–**


APPLE II für Technik und Wissenschaft bietet ein faszinierendes und vielfältiges Spektrum naturwissenschaftlicher Aufgabestellungen, dokumentiert mit vielen interessanten Listings von der Statistik bis zur Atomphysik. So wird der APPLE II zur wissenschaftlichen Hilfskraft. **APPLE II für Wissenschaft und Technik, 1984, 268 Seiten, DM 49,–**

**BESTELL-COUPON**

Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

zzgl. DM 5,– Versandkosten
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name und Adresse bitte deutlich schreiben

# DATA BECKER

**Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010**