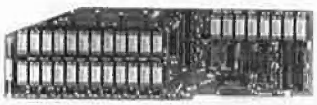
Nr. 5/85 Mai
DM 6,50, sfr 6,50, ÖS 50, Lit 5900, hfl 7,50

PEEKER

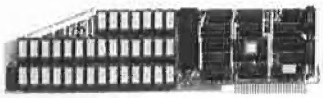
MAGAZIN FÜR APPLE-COMPUTER

DOS-File-Manager
Funktionsplotter
Geheime 6502-Opcodes
CP/M-Directory
Mac-Monitor
Kopierschutzverfahren
Apple-Kompatible

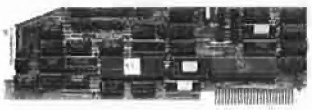
Hüthig
PUBLIKATION



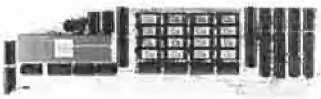
AP 13 und AP 17
RAM-Karten zum Einsatz als Pseudodisk unter CP/M, USCD und APPLE-DOS. Speichergröße von 64 kByte bis 256 kByte.
Bestell-Nr.: A 1013 a-b
A 1017 a-d



AP 33
RAMDISK der neuen Generation. Für besonders speicherintensive Arbeiten ist der Ausbau in Stufen von 64 kByte bis 1MByte möglich.
Bestell-Nr. A 1033

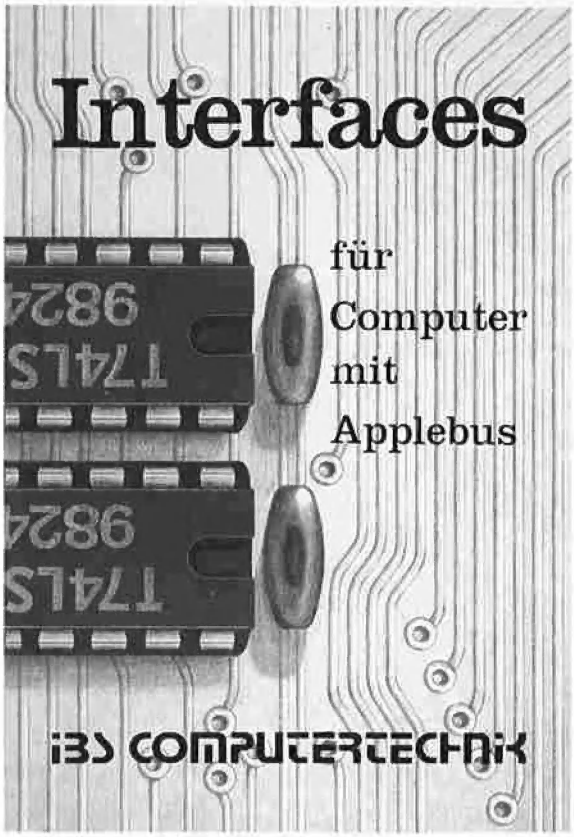


AP 14
Floppy-Controller für alle Anwendungsfälle. 10 Laufwerke können gleichzeitig angeschlossen werden. 4 × 8" DSDD, 4 × 5¼" DSDD und zwei Apple-Standardlaufwerke. Maximal ca. 10MByte im Direktzugriff.
Bestell-Nr.: A 1014

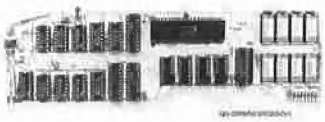


NEU! jetzt 512 k-RAM

AP 20
INTEMEX mit 68 000 CPU und 128 k-RAM. Diese Karte macht aus Ihrem Rechner mit „Applebus" einen echten 16 bit-Rechner. Eine Zusatzkarte (AP 26) ermöglicht einen Arbeitsspeicher bis zu einem MByte und an Software gibt es einiges. Z.B. stehen drei Betriebssysteme und die wichtigsten Hochsprachen zur Verfügung.
Bestell-Nr. A 1020



AP 19
12-Kanal AD-DA-Wandler mit 12 bit Auflösung und 25 μ sec Wandlungszeit. Eingangsspannung ±10 V. Ein schneller Wandler für extrem schnelle Anwendungen.
Bestell-Nr.: A 1019



NEU! 8 MHz Takt

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

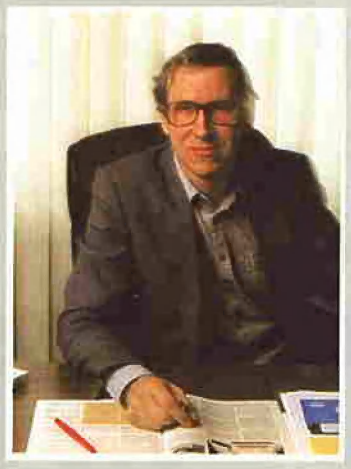
NEU!!!

Das Interface-Buch von IBS, ein Buch für Alle, die Ihren APPLE II oder Kompatiblen optimal nutzen wollen. Detaillierte Schaltpläne, Bauteilelisten und Benutzungshinweise zu allen IBS-Interfaces finden Sie jetzt in einem Buch vereint. Ausführliche Abhandlungen über Spezialschaltungen, über Anwendungsmöglichkeiten, über neue Softwarewelten aber auch über die Grenzen des APPLE II-Systems bestimmen den Wert dieses Buches.
Für nur DM 8,00 erhalten Sie dieses Buch ab sofort bei Ihrem Computerfachhändler oder für DM 8,00 + DM 2,00 Versandkosten bei IBS COMPUTERVERTRIEB.

IBS COMPUTERTECHNIK

Olper Straße 10 · 4800 Bielefeld 14 · Tel.: 0521/444032 · W. Germany
1011 Rose Marie Lane 16 · Stockton CA 95207 · Tel. 209/473 – 7473 USA

EDITORIAL



Der Kopierrecht-Aufsatz im letzten Peeker hat eine Fülle telefonischer Anfragen ausgelöst. Insbesondere die 7-Exemplar-Regelung war vielen Lesern neu. Ich darf hier nochmals Prof. Nordemann zitieren: „Es dürfen auch hier nur einzelne Kopien, Aufnahmen usw. hergestellt werden... Das waren schon nach Ansicht der Bundesregierung etwa 6 bis 7 Exemplare (so die schriftliche Antwort des Bundesministers der Justiz vom 16. Juli 1968 auf eine kleine Anfrage). Aber es bedurfte erst einer bestätigenden höchstrichterlichen Entscheidung, um das Bundesland Bremen zur Beachtung des Gesetzes zu veranlassen" („Urheberrecht", 5. Aufl., S. 328).

Das Ausnutzen von Gesetzeslücken mag zwar moralisch verpönt sein, wird jedoch allenthalben praktiziert. Beispielsweise sind ganze Armeen von Steuerberatern damit befaßt, unter Anwendung aller gerade noch legalen Tricks auch die letzte Steuermark für ihre Klienten einzusparen. Das Urheberrechtsgesetz enthält bezüglich der „Computerwerke" (= Software) jedoch mehr Lücken als sinnvoll anwendbare Paragraphen, so daß man im Handumdrehen eine Fülle von Antinomien konstruieren kann, von denen wir eine herausgreifen wollen:

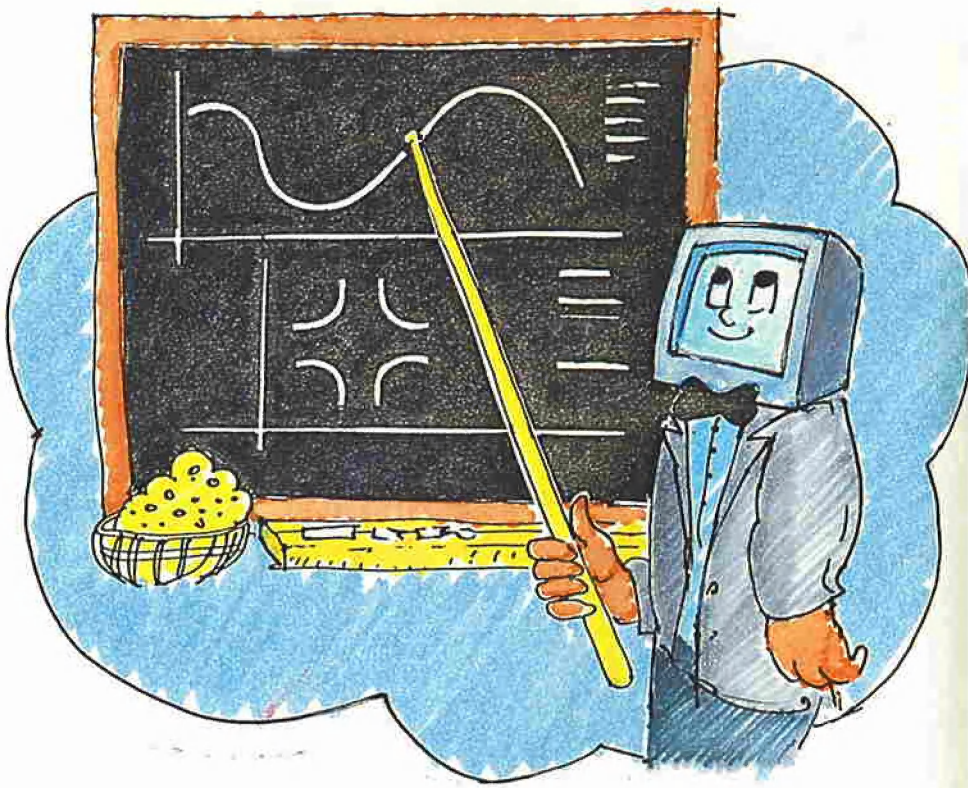
Als der Gesetzgeber den § 53 UrhG formulierte, dachte er nicht daran, daß von Betriebssystemen mehr als nur „einzelne" Vervielfältigungsstücke zum persönlichen oder sonstigen eigenen Gebrauch benötigt werden könnten. Peeker-Leser, die beispielsweise unter DOS 3.3 arbeiten, haben mit Gewißheit schon mehr als 7 Vervielfältigungsstücke von DOS 3.3 durch „INIT HELLO" hergestellt. Wäre DOS 3.3 ein Urheberwerk, könnten wohl die meisten Apple-Besitzer wegen Verletzung von § 53 als „Raubkopierer" belangt werden. Zwar kann es nicht im Interesse einer Softwarefirma liegen, sich mit Tausenden von Anzeigen zu befassen, weil sie dann letztlich ihre Kunden verlieren würde, doch möchte der Anwender nicht in der ständigen Ungewißheit leben, daß seine Disketten, die neben dem Betriebssystem für ihn wichtige eigene Daten enthalten (Briefe, Adressen, Meßwerte, Lagerbestände usw.), eines schönen Tages von der Polizei beschlagnahmt werden könnten. Wie kommt man aus diesem Dilemma heraus? Es bieten sich folgende Möglichkeiten an:

1. Das Urheberrechtsgesetz wird dahingehend novelliert, daß von einem Betriebssystem beliebig viele und nicht nur einzelne Vervielfältigungsstücke für den persönlichen (§ 53) und sonstigen eigenen Gebrauch (§ 54) hergestellt werden dürfen.
2. Das jeweilige Systemhaus erklärt beim Verkauf eindeutig, ob von dem Betriebssystem beliebig viele Vervielfältigungsstücke zum persönlichen oder sonstigen eigenen Gebrauch hergestellt werden dürfen. Dann weiß jeder, ob er im Falle der Herstellung von mehr als 7 Kopien mit zivil- und strafrechtlichen Konsequenzen rechnen muß.
3. Durch eine kollektive Feststellungsklage einer möglichst großen Zahl von Mikrocomputer-Besitzern wird die Beseitigung dieser Rechtsunsicherheit auf dem Prozeßwege erzwungen.

Zum Schluß noch ein Hinweis in eigener Sache: Das lange erwartete Hires-Grafik-Dump-Programm für den Imagewriter (s. Peeker, Heft 1/84) ist soeben fertiggestellt worden. Wir haben es deshalb bereits in die Sammeldisk #5 aufgenommen, obwohl es erst im nächsten Heft erläutert wird. Und noch eine „letzte Meldung": Das MOUSE.ASS von Disk #4 muß mit Pascal 1.1 assembliert werden, weil der 1.2-Assembler nicht mehr als 10 Prozeduren verkraftet.

Ulrich Stiehl

Hüthig
PUBLIKATION



INHALT

5/85

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

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

PEEKER

MAGAZIN FÜR APPLE-COMPUTER

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

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

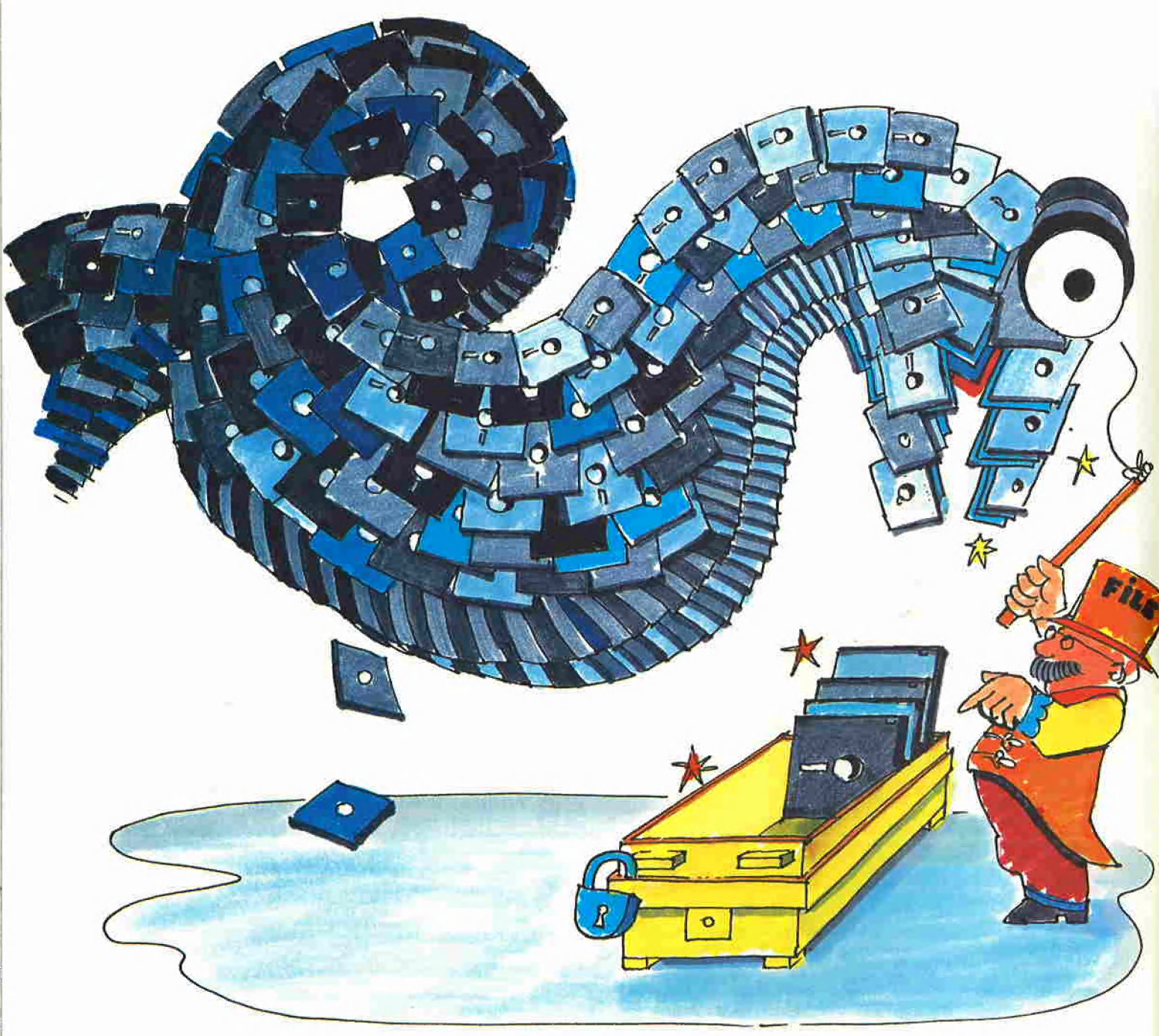
Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postscheckkonten: BRD: Karlsruhe 485 45-753; Österreich: Wien 7555888; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 47718; Belgien: Brüssel 723026; Dänemark: Kopenhagen 349 69; Norwegen: Oslo 994 24; Schweden: Stockholm 5477 76-5

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

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Der File-Manager des DOS 3.3

von Harald Grumser



Das schon etwas antiquierte Betriebssystem DOS 3.3, das im August 1980 erschien, ist trotz ProDOS auf dem besten Wege, noch zu Lebzeiten zur Legende zu werden. Obwohl die Firma Apple in ihrer Garagenzeit Akzente in der Transparenz von Computersystemen gesetzt hatte, wurde die innere Struktur des DOS mit Ausnahme der RWTS stets wie ein Staatsgeheimnis gehütet. In dem vorliegenden Beitrag soll deshalb der Schleier des File-Managers, eines bislang wenig bekannten Teils des DOS, gelüftet werden.

Das Disk-Operating-System des Apple II läßt sich in 3 Teile gliedern. Der am besten dokumentierte Teil ist ohne Zweifel der 3. Teil, d.h. die Read/Write-Track/Sektor-Routine oder RWTS, die im Benutzerhandbuch kurz beschrieben wird. Ihre Aufgabe ist der Transfer von Daten zwischen RAM und Diskette; sie bildet somit die Schnittstelle zwischen Speicher und Laufwerk und umfaßt den Bereich von $B800 bis $BFFF. Der 1. Teil ($9D00–$AAC0) des Betriebssystems wird vom Command-Interpreter eingenommen, der für die Befehlserkennung, die Parameterübergabe, die Formate der einzelnen Dateitypen und die Bereitstellung der DOS-Puffer verantwortlich ist. Die eigentliche Verwaltung der Dateien ist Aufgabe des 2. Teils, des File-Managers (= FM), der zwischen dem Command-Interpreter und der RWTS liegt. Der File-Manager verfügt über eine eigens für den Assemblerprogrammierer eingerichtete Schnittstelle, die übrigens vom Command-Interpreter nicht benutzt wird und die über einen der Page-3-Vektoren indirekt aufgerufen werden kann. Alle für den Aufruf benötigten Werte werden in der File-Manager-Parameter-Liste übergeben, deren Adresse durch Aufruf einer kurzen Page-3-Routine ermittelt werden kann. Durch dieses Verfahren ist es möglich, den FM auch in verschobenen und gepatchten DOS-Versionen ausfindig zu machen.

Die Organisation einer Diskette

Vor der näheren Analyse dieser Schnittstelle sei noch einmal kurz die Organisation einer DOS-3.3-Diskette beschrieben. Eine Diskette ist im allgemeinen in 35 Spuren zu je 16 Sektoren mit jeweils 256 Bytes aufgeteilt: Dies macht zusammen 560 Sektoren oder 143.360 Bytes. Bei diesen 560 Sektoren sind 5 verschiedene Typen zu unterscheiden:

DOS-Image – Die Spuren 0 bis 2 beinhalten das Image des Systemprogramms, das beim Booten in mehreren Schritten in den Speicher geschrieben wird. Von diesen 3 mal 16 Sektoren werden jedoch nur 37 Sektoren benutzt. Die verbleibenden 11 Sektoren auf Spur 2 werden beim Initialisieren einer Diskette dennoch als belegt gekennzeichnet, es verbleiben also 560 - 46 = 512 Sektoren.
VTOC – Die Volume Table of Contents stellt den Haupteintrag im Inhaltsverzeichnis der Diskette dar und umfaßt den Sektor 0 der Spur 17. In ihr sind unter anderem die Position des Catalogs und die Track-Bit-Map enthalten. In dieser Spur-Bit-Tabelle sind für jeden Sektor 4 Bytes als Bitmuster angelegt, wobei ein gesetztes Bit den entsprechenden Sektor freigibt und ein gelöschtes Bit den Sektor als benutzt kennzeichnet.
Catalog-Sektoren – Das Inhaltsverzeichnis belegt die 15 verbleibenden Sektoren der Spur 17. Dabei beinhaltet jeder Sektor ein Linkfeld zum nächsten Catalog-Sektor und 7 Felder für Dateieintragungen. Somit sind 7 mal 15 = 105 Dateieinträge möglich. Eine Erweiterung des Catalogs ist nicht vorgesehen. Jeder Catalog-Eintrag enthält außer Name, Dateityp und -länge ein Linkfeld zum ersten Sektor der Track/Sektor-Liste (= TSL) einer Datei. Subtrahiert man die Spur 17 von der Nettospeicherkapazität, so verbleiben 496 Sektoren oder 126.976 Bytes zur Speicherung von Daten.
Track/Sektor-Liste – Die TSL enthält außer einem Linkfeld zur nächsten TSL eine Tabelle von 122 T/S-Paaren, die auf die Datensektoren einer Datei verweisen. Umfaßt eine Datei mehr als 122 Datensektoren, so wird ein weiterer TSL-Sektor angelegt.
Datensektoren – Die für den Benutzer letztlich einzig bedeutungsvollen Sektoren sind die Datensektoren, die entsprechend dem Dateityp außer den Informationen über Position und Länge im allgemeinen ein Abbild des Speichers darstellen. Auf die Sonderstellung von Textdateien soll hier nicht weiter eingegangen werden.

Der Aufbau eines DOS-Puffers

Es erweist sich zum Verständnis des FM vorteilhaft, sich noch einmal den Aufbau eines DOS-Puffers vor Augen zu führen. Der Command-Interpreter richtet beim Kaltstart oder der Angabe von MAXFILES eine DOS-Buffer-Chain ein, die meist aus 3 Puffern besteht. Jeder geöffneten Datei (dieser Begriff wird später noch einmal erläutert) wird ein Puffer zugewiesen. Ein solcher Puffer umfaßt 595 Bytes und ist in 5 Felder einteilbar:

Datenpuffer – $000-$0FF (000-255). In diesem Bereich sammelt der FM die zu lesenden oder zu schreibenden Informationen, um ausschließlich ganze Sektoren zu transferieren.
TSL-Puffer – $100-$1FF (256-511). Dieser Bereich nimmt den gerade aktuellen Sektor der TSL auf.
Puffer des FM-Arbeitsbereichs – $200-$22C (512-556). Der FM sichert hier nach jedem Aufruf alle im Zusammenhang mit dieser Datei wichtigen Werte und liest sie bei einem erneuten Aufruf wieder ein.
Puffer des Dateinamens – $22D-$24A (557-586) Ein Dateiname hat stets 30 Zeichen (Bit 7 = 1), wobei der letzte Teil mit Blanks ($A0) aufgefüllt wird. Ist dieser DOS-Puffer unbenutzt, so erscheint eine Null im ersten Byte.
Adreßliste – $24B-$252 (587-594) Diese Adreßliste besteht aus 4 Adressen im LSB/MSB-Format und verweist auf die ersten 3 Felder dieses Puffers und das Namensfeld des nächsten Puffers. Beim letzten Puffer enthält die letzte Adresse Nullen.

Der Aufruf des File-Managers

Wie bereits oben erwähnt, kann der FM über einen der Page-3-Vektoren, und zwar durch JSR $03D6 aufgerufen werden. Bei normalem DOS 3.3 ist dies auch durch JSR $AAFD möglich. In jedem Fall muß vor dem Aufruf die Parameter-Liste entsprechend vorbereitet werden. Die Anfangsadresse dieser Liste läßt sich durch folgende kleine Routine ermitteln:

```
JSR $03DC
STY $ED
STA $EE
```

Danach kann mittels ($ED),Y auf die Parameter-Liste zugegriffen werden. Bei normalem DOS ist deren Adresse $B5BB.
Die zu übergebenden Parameter sind von dem auszuführenden Befehl abhängig und können der **Tabelle 3** entnommen werden. Es ist ersichtlich, daß nicht für alle Befehle alle Parameter benötigt werden. Bevor nun im folgenden auf die einzelnen Befehle eingegangen wird, noch einige allgemeine Bemerkungen.

Der FM kann stets nur eine Datei bearbeiten. Dabei muß jede Datei zunächst einmal eröffnet werden. Dieses Eröffnen ist nicht mit dem OPEN-Befehl bei Textdateien zu

verwechseln. Alle für die Bearbeitung der Datei benötigten Variablen legt der FM in seinem Arbeitsbereich, der sog. File-Manager-Work-Area, ab. Um bei einem späteren Aufruf diese Variablen wieder parat zu haben, kopiert er diese nach getaner Arbeit in einen Puffer, dessen Adresse vom aufrufenden Programm in der Parameter-Liste auf Position 12 und 13 bereitgestellt werden muß. Das gleiche gilt für die TSL der gerade in Arbeit befindlichen Datei. Auch dieser Bereich muß vom aufrufenden Programm bereitgestellt und in Position 14 und 15 der Parameter-Liste eingetragen werden. Benötigt der FM einen Datenpuffer, so ist dieser auf Position 16 und 17 zu übergeben.

Hier zeigt sich der Sinn und Zweck der DOS-Puffer; sie enthalten nämlich genau die geforderten Bereiche, d.h. daß bei der richtigen Zuordnung des richtigen Puffers beliebig viele Dateien gleichzeitig bearbeitet werden können. Es ist nur sicherzustellen, daß bei jedem Aufruf die entsprechenden Pufferbereiche übergeben werden.

Das 10. Byte der Parameter-Liste hat eine Sonderstellung. Es dient zur Rückgabe eines Fehlercodes. Trat während der Arbeit des FM ein Fehler auf, wird das Prozessor-Carryflag auf 1 gesetzt und ein entsprechender Code im 10. Byte eingetragen, ansonsten werden das Carryflag und der Fehlercode zu 0. Die Bedeutung der einzelnen Fehlernummern entnehme man der **Tabelle 1**.

Die letzten beiden allgemeingültigen Bytes sind das Byte auf Position 11, das ungenutzt ist, und das Byte auf Position 0, das zur Übergabe der Befehlsnummer dient.

Übersicht über die FM-Befehle

OPEN – Der Open-Befehl dient zur Eröffnung einer Datei. Dabei müssen die Laufwerksparameter Slot und Drive sowie die Volume-Nummer (0 deckt alle Volume-Nummern ab) übergeben werden. Die Adresse des Dateinamens muß in Position 8 und 9 eingetragen werden, der Dateityp in Position 7, wobei Bit 7, das Lockbit, stets 0 sein sollte (siehe hierzu **Tabelle 2**). Soll später eine Positionierung innerhalb der Datei erfolgen, so ist die Recordlänge festzulegen; Recordlänge 0 wird intern zu 1 geändert. Der FM sucht im Inhaltsverzeichnis der Diskette eine Datei gleichen Namens und trägt den vorgefundenen Dateityp in Position 7 ein. Das aufrufende Programm kann durch Setzen des X-Registers beeinflussen, ob eine neue Datei angelegt wird, falls eine Datei gleichen Namens noch nicht existiert. Bei X ungleich 0 erfolgt bei Nichtbestehen die Fehlermeldung „Datei nicht gefunden", bei X = 0 wird eine neue angelegt. Der erste TSL-Sektor wird automatisch eingelesen. Einem Open-Befehl sollte stets eine Positionierung folgen.

POSITION – Durch diesen Befehl kann der Dateizeiger auf eine beliebige Position innerhalb der Datei gesetzt werden. Dieser Zeiger wird durch einfache Multiplikation der Recordlänge mit der Recordnummer ermittelt. Durch Addition des Record-Offsets kann dabei auch eine Position innerhalb eines Feldes eingestellt werden. Dieser Befehl wurde ursprünglich für die Bearbeitung von Random-Access-Dateien implementiert, kann aber bei jedem anderen Dateityp angewandt werden.

CLOSE – Der Close-Befehl schließt die Bearbeitung einer Datei ab. Dabei wird der letzte Datensektor auf die Diskette geschrieben und die VTOC sowie die TSL aktualisiert. Ein bereitgestellter DOS-Puffer wird nicht freigegeben.

Tabelle 3: File-Manager-Parameter-Liste

	$00	$01	$02 | $03	$04	$05	$06	$07	$08 | $09	$0A	$0B	$0C | $0D	$0E | $0F	$10 | $11
OPEN	$1		Rekordlänge	V	D	S	Typ	1. Namensadresse	RETURN CODE	nicht benutzt	FMWA-Adresse	TSL-Adresse	
POSITION	$A		Recordnummer	Recordoffset							–"–		
CLOSE	$2										–"–	TSL-Adresse	Datenadresse
READ/WRITE	$3/$4	$1						Datenbyte			–"–	–"–	–"–
		$2				Pufferlänge		Pufferadresse			–"–	–"–	–"–
		$3	Recordnummer	Recordoffset				Datenbyte			–"–	–"–	–"–
		$4	–"–	–"–		Pufferlänge		Pufferadresse			–"–	–"–	–"–
DELETE	$5			V	D	S		1. Namensadresse			–"–	–"–	
CATALOG	$6				D	S					–"–		
LOCK	$7			V	D	S		1. Namensadresse			–"–	TSL-Adresse	
UNLOCK	$8			V	D	S		–"–			–"–	–"–	
RENAME	$9		2. Namensadresse	V	D	S		–"–			–"–	–"–	
INIT	$B	DOS-seite		V	D	S					–"–		
VERIFY	$C			V	D	S		1. Namensadresse			–"–	TSL-Adresse	Datenadresse

READ/WRITE – Read und Write unterscheiden sich außer in der Befehlsnummer (Read = 3, Write = 4) kaum in den Parametern. Beide Befehle können durch den Subcode auf Position 1 nochmals näher spezifiziert werden. Bei Subcode 1 und 2 wird an der Stelle fortgefahren, an der der letzte Schreib/Lese-Vorgang endete, bei Subcode 3 und 4 wird entsprechend den zu übergebenden Recordwerten vorher positioniert. Mit Subcode 1 und 3 wird pro Aufruf immer nur 1 Byte gelesen/geschrieben, das auf Position 8 eingetragen bzw. von dort entnommen wird. Subcode 2 und 4 ermöglichen den Transfer ganzer Blöcke. Dabei muß in Position 8 und 9 die Lage dieses Blocks eingetragen werden. Die Werte auf Position 6 und 7 dienen dabei als Blockzähler (= Länge des Blocks), wobei darauf zu achten ist, daß beim Schreiben diese Länge um 1 vermindert angegeben werden muß.

DELETE – Dieser Befehl entfernt einen Catalog-Eintrag aus dem Inhaltsverzeichnis der Diskette. Entfernen bedeutet hierbei kein Löschen, sondern eine spezielle Markierung. In einem weiteren Zug werden dann alle durch diese Datei belegten Sektoren freigegeben. Dieser Befehl führt implizit ein OPEN durch. Das X-Register sollte daher, um dem FM unnötige Arbeit zu sparen, beim Aufruf stets ungleich 0 sein.

CATALOG – Dieser Befehl listet das Inhaltsverzeichnis der Diskette und entspricht exakt dem normalen CATALOG-Befehl. Es sei in diesem Zusammenhang erwähnt, daß der FM bei jedem Aufruf den in Position 12 und 13 angegebenen Arbeitsbereich einliest, obwohl dies bei einigen Befehlen nicht notwendig wäre. Es kann bei diesem Aufruf also durchaus auf die Benutzung eines DOS-Puffers verzichtet und statt dessen ein beliebiger 45 Bytes großer Bereich angegeben werden.

LOCK/UNLOCK – Diese beiden Befehle verändern das Bit 7 des Dateitypbytes im Catalog-Eintrag. Da auch diese Befehle ein implizites OPEN durchführen, sollte auch bei diesem Aufruf das X-Register ungleich 0 sein.

RENAME – Hierbei wird das Namensfeld in einem Catalog-Eintrag geändert. Die Adresse des zweiten Namens ist auf Position 2 und 3 zu übergeben. Auch hier wird ein Open durchgeführt.

INIT – Dieser Befehl initialisiert durch Aufruf der RWTS die Diskette und schreibt dann das DOS-Image auf die Spuren 0 bis 2. Anschließend wird die VTOC und der Catalog angelegt. Auf Position 1 ist die

Tabelle 1: File-Manager-Fehlermeldungen

```
 Ø – kein Fehler
 1 – Sprache nicht verfügbar
     (beim Laden von RAM-A/S)
 2 – ungültige Befehlsnummer
 3 – ungültiger Subcode
 4 – Diskette schreibgeschützt
 5 – keine weiteren Daten
 6 – Datei nicht gefunden
     (bei X = Ø dennoch Neuzuweisung)
 7 – falsche Volumenummer
 8 – I/O-Fehler
 9 – Diskette voll
1Ø – Datei gesperrt (locked)
```

Tabelle 2: Dateitypen

```
Bit 7 = 1 bei gesperrten (locked) Dateien
Bit 7 = Ø bei nicht gesperrten Dateien

$ØØ – Textdatei
$Ø1 – Integer-Basic-Programmdatei
$Ø2 – Applesoft-Programmdatei
$Ø4 – Binärdatei
$Ø8 – unbenutzt (Typ S)
$1Ø – verschiebbares Assemblermodul (Typ R)
$2Ø – unbenutzt (Typ A)
$4Ø – unbenutzt (Typ B)
```

FM.BSP

```
                          1  *-------------------------------------------------*
                          2  *          File-Manager Anwendungsbeispiele       *
                          3  *-------------------------------------------------*
                          4
                          5           ORG  $8ØØØ
                          7
                          8  FMPL     EQU  $EB        ;Adresse der F/M-Parameter-Liste
                          9  BUFADR   EQU  $ED        ;Adresse des DOS-Puffers
                          1Ø DOSWRM   EQU  $3DØ       ;Sprung zum DOS-Warmstart
                          11 FLEMGR   EQU  $3D6       ;Sprung zum File-Manager
                          12 GETFMPL  EQU  $3DC       ;FMPL ermitteln
                          13 CROUT    EQU  $FD8E      ;CR ausgeben
                          14 PRHEX    EQU  $FDE3      ;Hexzahl ausgeben
                          15 PRERR    EQU  $FF2D      ;"ERR" ausgeben
                          16
                          17 *---------------------- DELETE ----------------------*
                          18
8ØØØ: 2Ø 51 8Ø            19          JSR  STARTUP
8ØØ3: 2Ø 6D 8Ø            2Ø          JSR  SETFLN1    ;Dateiname eintragen
8ØØ6: A9 Ø5               21          LDA  #5         ;DELETE-Befehl
8ØØ8: 4C 82 8Ø            22          JMP  CALLFM
                          23
                          24 *---------------------- CATALOG ---------------------*
                          25
8ØØB: 2Ø 51 8Ø            26          JSR  STARTUP
8ØØE: A9 Ø6               27          LDA  #6         ;CATALOG-Befehl
8Ø1Ø: 4C 82 8Ø            28          JMP  CALLFM
                          29
                          3Ø *----------------------- LOCK -----------------------*
                          31
8Ø13: 2Ø 51 8Ø            32          JSR  STARTUP
8Ø16: 2Ø 6D 8Ø            33          JSR  SETFLN1    ;Dateiname eintragen
8Ø19: A9 Ø7               34          LDA  #7         ;LOCK-Befehl
8Ø1B: 4C 82 8Ø            35          JMP  CALLFM
                          36
                          37 *---------------------- UNLOCK ----------------------*
                          38
8Ø1E: 2Ø 51 8Ø            39          JSR  STARTUP
8Ø21: 2Ø 6D 8Ø            4Ø          JSR  SETFLN1    ;Dateiname eintragen
8Ø24: A9 Ø8               41          LDA  #8         ;UNLOCK-Befehl
8Ø26: 4C 82 8Ø            42          JMP  CALLFM
                          43
                          44 *---------------------- RENAME ----------------------*
                          45
8Ø29: 2Ø 51 8Ø            46          JSR  STARTUP
8Ø2C: 2Ø 75 8Ø            47          JSR  SETFLN2    ;RENAME-Dateiname eintragen
8Ø2F: 2Ø 6D 8Ø            48          JSR  SETFLN1    ;alten Dateiname eintragen
8Ø32: A9 Ø9               49          LDA  #9         ;RENAME-Befehl
8Ø34: 4C 82 8Ø            5Ø          JMP  CALLFM
                          51
                          52 *----------------------- INIT -----------------------*
                          53
8Ø37: 2Ø 51 8Ø            54          JSR  STARTUP
8Ø3A: AD D2 Ø3            55          LDA  DOSWRM+2   ;erste Seite des DOS-Image
8Ø3D: AØ Ø1               56          LDY  #1         ; ermitteln und
8Ø3F: 91 EB               57          STA  (FMPL),Y   ; eintragen
8Ø41: A9 ØB               58          LDA  #11        ;INIT-Befehl
8Ø43: 4C 82 8Ø            59          JMP  CALLFM
                          6Ø
                          61 *---------------------- VERIFY ----------------------*
                          62
8Ø46: 2Ø 51 8Ø            63          JSR  STARTUP
8Ø49: 2Ø 7B 8Ø            64          JSR  SETFLN1    ;Dateiname eintragen
```

erste Seite des sich im Speicher befindlichen DOS anzugeben (normalerweise $9D00), auf Position 4 die Volume-Nummer.
VERIFY – Es handelt sich hier um denselben Befehl, wie er auch unter Basic Anwendung findet. Praktisch geschieht nichts anderes als ein komplettes Lesen der Datei, wobei ausschließlich RWTS-Fehler auftreten können.

Um die Verwendung des FM noch einmal zu verdeutlichen, ist diesem Beitrag ein kurzes Beispielprogramm **FM.BSP** angefügt, das den Aufruf der einzelnen Befehle näher erläutern soll. Die dort nicht aufgeführten Befehle werden in einem späteren Beitrag anhand einer FID-ähnlichen File-Copy-Utility ausgeführt.

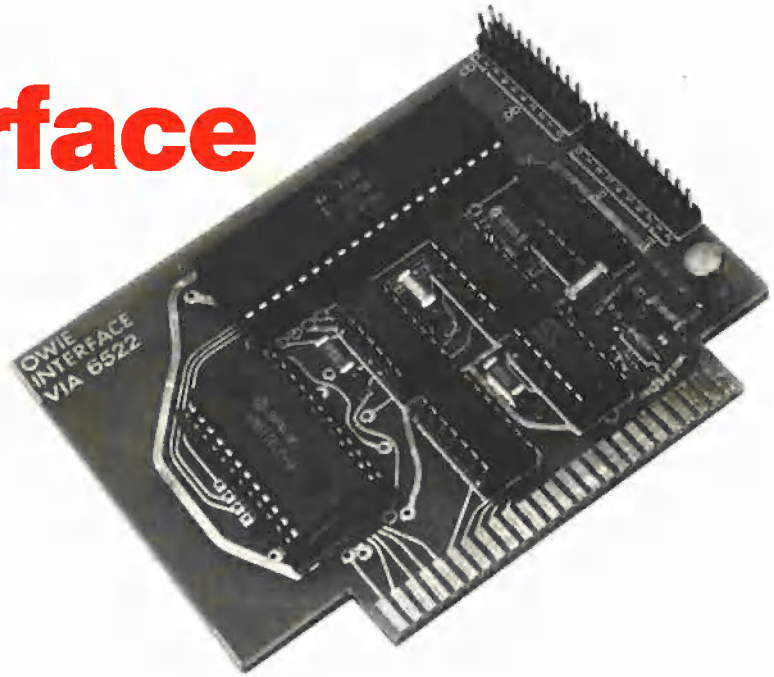
Leider läßt sich der FM nicht in jeder DOS-Version ohne weiteres aufrufen. Das von Glen Bredon stark gepatchte DOS des Merlin-Assemblers verweigert die Zuordnung neuer Dateien. Der Schlüssel zu diesem Problem liegt in einem Byte (oder besser noch in einem Bit) des Command-Interpreters, der eine Tabelle zur Überprüfung der Gültigkeit der verschiedenen Parameter jedes Befehls enthält. In dieser Tabelle ist je 1 Bit für die Möglichkeit einer Dateineuzuweisung angesetzt. Der alte, jetzt geänderte INIT-Befehl ließ dies zu. Da daraus der ASSEM-Befehl wurde, hat Bredon dieses Bit zurückgesetzt. Beim Aufruf des FM wird durch das X-Register der INIT- oder LOAD-Befehl simuliert, d.h. in dem einen Fall ist eine Neuzuweisung möglich, in dem anderen nicht. Nach Bredons Eingriff war in beiden Fällen keine mehr möglich. Eine schnelle Lösung für dieses Problem: $A909 wird wieder zu $21. Dies tut dem Merlin keinen Abbruch und der FM arbeitet wieder, wie er soll.

```
804C: A9 0C     65            LDA  #12         ;VERIFY-Befehl
804E: 4C 82 80  66            JMP  CALLFM
                67
                68  *-------------------------------------------------*
                69
8051: 20 DC 03  70  STARTUP   JSR  GETFMPL     ;Adresse der File-Manager-
8054: 84 EB     71            STY  FMPL        ; Parameter-Liste
8056: 85 EC     72            STA  FMPL+1      ; ermitteln
                73
8058: AD D2 03  74            LDA  DOSWRM+2    ;Adresse des 1. DOS-Puffers
805B: 85 EE     75            STA  BUFADR+1    ; ermitteln
805D: A0 00     76            LDY  #0
805F: 84 ED     77            STY  BUFADR      ; (1. Eintrag im DOS-Image
8061: B1 ED     78            LDA  (BUFADR),Y  ; verweist auf den 1. Puffer
8063: AA        79            TAX              ; der DOS-Buffer-Chain)
8064: C8        80            INY
8065: B1 ED     81            LDA  (BUFADR),Y
8067: 85 EE     82            STA  BUFADR+1
8069: 8A        83            TXA
806A: 85 ED     84            STA  BUFADR
806C: 60        85            RTS
                86
806D: A9 BD     87  SETFLN1   LDA  #<FLENME1   ;erster Dateiname
806F: A2 80     88            LDX  #>FLENME1
8071: A0 08     89            LDY  #8          ;Offset zum 1. Dateinamen
8073: D0 06     90            BNE  SETFLN      ;unbedingt
8075: A9 DB     91  SETFLN2   LDA  #<FLENME2   ;RENAME-Dateiname
8077: A2 80     92            LDX  #>FLENME2
8079: A0 02     93            LDY  #2          ;Offset zum 2. Dateiname
807B: 91 EB     94  SETFLN    STA  (FMPL),Y    ;LSB der Adresse eintragen
807D: C8        95            INY
807E: 8A        96            TXA
807F: 91 EB     97            STA  (FMPL),Y    ;MSB der Adresse eintragen
8081: 60        98            RTS
                99
                100 * File-Manager aufrufen
                101
8082: A2 01     102 CALLFM    LDX  #1          ;keine Neuzuweisung
8084: A0 00     103           LDY  #0          ;CALLTYP in Parameter-
8086: 91 EB     104           STA  (FMPL),Y    ; Liste eintragen
8088: A0 1E     105           LDY  #30
808A: B1 ED     106 GETADRS   LDA  (BUFADR),Y ;Pufferadressen aus DOS-
808C: 48        107           PHA              ; Puffer einlesen
808D: C8        108           INY
808E: C0 24     109           CPY  #36         ;  (FM-Arbeitspuffer)
8090: 90 F8     110           BCC  GETADRS     ;  (TSL-Puffer)
8092: A0 11     111           LDY  #17         ;  (Datenpuffer)
8094: 68        112 SETADRS   PLA
8095: 91 EB     113           STA  (FMPL),Y    ;und in die Parameterliste
8097: 88        114           DEY              ; eintragen
8098: C0 0C     115           CPY  #12
809A: B0 F8     116           BCS  SETADRS
809C: A0 04     117           LDY  #4          ;Volume
809E: B9 F5 80  118 SETVDS    LDA  VODRSL-4,Y ;  Drive
80A1: 91 EB     119           STA  (FMPL),Y    ;    Slot
80A3: C8        120           INY
80A4: C0 07     121           CPY  #7          ; in Parameterliste
80A6: 90 F6     122           BCC  SETVDS      ; eintragen
80A8: 20 D6 03  123           JSR  FLEMGR      ;File-Manager via Page 3
80AB: A0 0A     124           LDY  #10
80AD: B1 EB     125           LDA  (FMPL),Y    ;Return-Code
80AF: 90 0B     126           BCC  NOERR
80B1: 48        127           PHA              ;Return-Code retten
80B2: 20 8E FD  128           JSR  CROUT
80B5: 20 2D FF  129           JSR  PRERR       ;"ERR" drucken
80B8: 68        130           PLA
80B9: 4C E3 FD  131           JMP  PRHEX       ;Fehlernummer ausgeben
80BC: 60        132 NOERR     RTS
                133
80BD: C4 C1 D4  134 FLENME1   ASC  "DATEI1"
80C3: A0 A0 A0  135           HEX  A0A0A0A0A0A0
80C9: A0 A0 A0  136           HEX  A0A0A0A0A0A0
80CF: A0 A0 A0  137           HEX  A0A0A0A0A0A0
80D5: A0 A0 A0  138           HEX  A0A0A0A0A0A0
80DB: C4 C1 D4  139 FLENME2   ASC  "DATEI2"
80E1: A0 A0 A0  140           HEX  A0A0A0A0A0A0
80E7: A0 A0 A0  141           HEX  A0A0A0A0A0A0
80ED: A0 A0 A0  142           HEX  A0A0A0A0A0A0
80F3: A0 A0 A0  143           HEX  A0A0A0A0A0A0
                144
80F9: 00        145 VODRSL    DFB  0           ;Volume
80FA: 01        146           DFB  1           ;Drive
80FB: 06        147           DFB  6           ;Slot

252 Bytes
```

Balfer-Interface

RAM-Disk-Driver für Apple IIe



von Ulrich Stiehl

Die Firma Digitanalog Ewald Balfer hat seit einiger Zeit eine preiswerte Interface-Karte im Produktionsprogramm, die als durchkontaktete Leiterplatte (DM 35,–), als Bausatz VIA 6522, CMOS-RAM und Sockel (DM 128,–) sowie als geprüfte Fertigkarte (DM 158,–) geliefert werden kann. Auf der uns zur Verfügung gestellten Komplettplatine befindet sich ein 2K-RAM, das jedoch auch durch ein EPROM 2716 ersetzt werden kann. Für die RAM-Version gelten folgende Adreßbereiche:

– CX00-CXFF (X = 1-7)
– C800-CEFF

Eigentlich ist die Karte als gewöhnliche Kommunikationsschnittstelle gedacht, z.B. als Parallel-Interface, serielles Interface (RS 232), Modem-Interface usw., und es werden von uns auch entsprechende Utilities für diesen Anwendungszweck vorbereitet, z.B. ein Drucker-Dump-Programm für Double-Hires-Grafik.

1. Balfer-Interface löst Softswitch-Probleme

Wer auf dem Apple IIe in Maschinensprache programmiert, dem fallen neben der „konventionellen" Anwendung ganz andere Einsatzmöglichkeiten für die dann nur noch als RAM-Speicher fungierende Karte ein. Der Apple IIe verfügt über relativ kompliziert zu handhabende Softswitches für die Speicherverwaltung (Memory Management). Dies rührt daher, daß die 64K-Karte des Apple IIe – im Gegensatz zu RAM-Karten anderer Produzenten (IBS usw.) – in denselben Adreßbereich wie die „unteren" 64K „gemappt" ist. Es gilt jedoch der alte Grundsatz der Speicherverwaltung, daß eine Memory-Management-Utility (MMU) normalerweise nicht in *demselben* Speicherbereich liegen sollte (oder darf), der durch die Speicherverwaltung tangiert wird. Ein einfaches Beispiel zur Erläuterung:

Nehmen wir an, wir wollten das Byte der Speicherstelle $2100 der „oberen" 64K (64K-Karte) in die Speicherstelle $2100 der „unteren" 64K (64K-Platine) übertragen. Darf sich dann das Übertragungsprogramm im Speicherbereich ab $2000 in den „unteren" 64K befinden? Beim Apple IIe (und im übrigen auch beim Apple IIc) gibt es u.a. 4 Softswitches, die für unser Beispiel von Interesse sind:

a) STA $C002 macht den „unteren" Speicherbereich $0200-$BFFF lesefähig (read-enable, read-select) und damit gleichzeitig den „oberen" Speicherbereich $0200-$BFFF leseunfähig.
b) STA $C003 macht den „oberen" Speicherbereich $0200-$BFFF lesefähig und damit gleichzeitig den „unteren" Speicherbereich $0200-$BFFF leseunfähig.
c) STA $C004 macht den „unteren" Speicherbereich $0200-$BFFF schreibfähig (write-enable, write-select) und damit gleichzeitig den „oberen" Speicherbereich $0200-$BFFF schreibunfähig.
d) STA $C005 macht den „oberen" Speicherbereich $0200-$BFFF schreibfähig und damit gleichzeitig den „unteren" Speicherbereich $0200-$BFFF schreibunfähig.

Betrachten wir hierzu folgende Fälle, bei denen angenommen sei, daß vor Programmbeginn der Speicherbereich $0200-$BFFF bereits lese- *und* schreibfähig gemacht worden sei:

Fall 1

```
ORG $2000
LDA $2100
STA $2100
RTS
```

Im Fall 1 würde das Byte von Speicherstelle $2100 in *dieselbe* Speicherstelle $2100 übertragen werden. Das Programm würde funktionieren, doch erfüllt es nicht die gestellte Aufgabe und wäre im übrigen auch sonst nutzlos.

Fall 2

```
ORG $2000
STA $C005
LDA $2100
STA $2100
STA $C004
RTS
```

Im Fall 2 würde das Byte der Speicherstelle $2100 des „unteren" Speicherbereichs $0200-$BFFF in die Speicherstelle $2100 des entsprechenden „oberen" Bereichs übertragen. Das Programm würde funktionieren, erfüllt aber nicht die gestellte Aufgabe.

Fall 3

```
ORG $2000
STA $C003
LDA $2100
STA $2100
STA $C002
RTS
```

Im Fall 3 wäre zwar theoretisch die gestellte Aufgabe erfüllt, aber das Programm würde nach STA $C003 jämmerlich „absaufen", weil der nachfolgende Befehl LDA $2100 nicht mehr vom Prozessor gelesen werden könnte, weil er nunmehr auf die „oberen" 64K „blickt", während sich das Programm in Wirklichkeit „unten" befindet.

Fall 4

```
ORG $0100
STA $C003
LDA $2100
STA $2100
STA $C002
RTS
```

Im Fall 4 würde endlich alles ordnungsgemäß funktionieren, weil sich nunmehr die MMU außerhalb des Softswitchbereiches $0200-$BFFF befindet. Die Verwendung des Stacks $0100-$01FF oder gar der Zero-Page $0000-$00FF als Programmbereich kann jedoch nicht gerade also optimal bezeichnet werden, da man sich normalerweise niemals sicher ist, ob dieser Bereich wirklich immer frei ist. Beispielsweise benutzt der Applewriter IIe den Stackbereich, um den Text von „unten" nach „oben" zu übertragen.

Fall 5

```
ORG $C400
STA $C003
LDA $2100
STA $2100
STA $C002
RTS
```

Im Fall 5 liegt die MMU im Slot-RAM-Bereich der Balfer-Interface-Karte, wobei hier willkürlich Slot 4 angenommen wurde, obgleich sich diese Karte in jedem beliebigen Slot befinden kann. Da der Slot-RAM-Bereich $C400-$C4FF oder allgemein $CX00-$CXFF praktisch immer lese- und schreibfähig ist, entstehen nicht mehr die skizzierten Softswitch-Probleme. Deshalb läßt sich der RAM-Bereich $CX00-$CXFF sowie der nach LDA $CFFF zusätzlich aktivierbare Bereich $C800-$CEFF ideal für MMU-Utilities unterschiedlicher Art einsetzen.

2. SLOTRAMDISK-Driver

Als konkrete Anwendung wird nachfolgend das Assembler-Programm SLOTRAMDISK (als Hexdump) in Verbindung mit einem kurzen Applesoft-Programm gelistet, das die 64K-Karte als RAM-Disk einsetzt, die mit einem beliebigen Slot (hier Slot 4) angesprochen werden kann, wobei die Drive-Nummer ignoriert wird, also

```
CATALOG, S4, D1
CATALOG, S4, D2
```

Aus Platzgründen wird der Quellcode des RAM-Disk-Drivers nur auszugsweise abgedruckt, da ein analoger RAM-Disk-Driver, der ohne das Balfer-Interface auskommt, aber softswitchmäßig erheblich umständlicher und damit auch etwas langsamer ist, in einem der nächsten Peeker-Hefte komplett gelistet wird.

Als weitere Programme für das Balfer-Interface sind u.a. geplant:

a) Double-Hires-Printer-Driver
b) Softbreaker-Utility
c) Modem-Programm
d) 80-Z/Z-Karte-Driver

SLOTRAMDISK

für Apple IIe mit 64K-Karte in S3 und Balfer-Karte in (z.B.) S4
BSAVE SLOTRAMDISK, A24736, L825
Hex-Dump:

```
$60A0: 4C A4 60 04 AD A3 60 18
$60A8: 69 C0 8D 26 61 8D 31 61
$60B0: AD A3 60 0A 0A 0A 0A 8D
$60B8: 20 63 A2 00 BD EE 60 F0
$60C0: 06 20 ED FD E8 D0 F5 20
$60C8: 0C FD C9 CE F0 11 C9 EE
$60D0: F0 0D C9 CA F0 06 C9 EA
$60D8: F0 02 D0 EB 20 56 61 20
$60E0: 07 61 EA EA EA A9 A9 8D
$60E8: 92 B2 EA 4C D0 03 8D D2
$60F0: C1 CD C4 C9 D3 CB B6 B4
$60F8: A0 ED E9 F4 A0 C9 EE E9
$6100: F4 A0 CA AF CE A0 00 A2
$6108: 03 BD 00 BD DD 2B 61 D0
$6110: 22 CA 10 F5 A2 03 BD 2F
$6118: 61 9D 00 BD CA 10 F7 A2
$6120: 00 BD 00 63 9D 00 C4 E8
$6128: D0 F7 60 84 48 85 49 4C
$6130: 06 C4 00 A2 00 BD 43 61
$6138: F0 06 20 ED FD E8 D0 F5
$6140: 4C EA 60 8D C4 CF D3 A0
$6148: ED EF E4 E9 E6 E9 FA E9
$6150: E5 F2 F4 87 8D 00 A9 00
$6158: 85 CE 8C 00 C0 8C 02 C0
$6160: 8C 04 C0 8C 08 C0 8C 09
$6168: C0 A8 99 00 00 C8 D0 FA
$6170: 99 00 01 C8 D0 FA 8C 08
$6178: C0 A0 02 84 CF 8C 05 C0
$6180: A8 91 CE C8 D0 FB E6 CF
$6188: A6 CF E0 C0 D0 F3 8C 04
$6190: C0 8C 09 C0 AC 83 C0 AC
$6198: 83 C0 8D A4 61 A0 D0 8C
$61A0: A5 61 A8 99 00 D0 C8 D0
$61A8: FA EE A5 61 D0 F5 AC 8B
$61B0: C0 AC 8B C0 8D BE 61 A0
```

```
$61B8: D0 8C BF 61 A8 99 00 D0
$61C0: C8 D0 FA EE BF 61 AE BF
$61C8: 61 E0 E0 D0 F0 8C 08 C0
$61D0: AC 81 C0 AC 81 C0 8C 05
$61D8: C0 A2 00 BD 00 62 9D 00
$61E0: 10 E8 D0 F7 A9 11 8D 01
$61E8: 1F 8D 01 1E A9 0E 8D 02
$61F0: 1F A9 0D 8D 02 1E 8C 04
$61F8: C0 8C 01 C0 20 58 FC 60
Catalog-VTOC für RAM-Disk
$6200: 00 11 0F 03 00 00 FE 00
$6208: 00 00 00 00 00 00 00 00
$6210: 00 00 00 00 00 00 00 00
$6218: 00 00 00 00 00 00 00 00
$6220: 00 00 00 00 00 00 00 7A
$6228: 00 00 00 00 00 00 00 00
$6230: 11 01 00 00 23 10 00 01
$6238: 00 00 00 00 00 00 00 00
$6240: 00 00 00 00 00 00 00 00
$6248: 00 00 00 00 00 00 00 00
$6250: 00 00 00 00 00 00 00 00
$6258: 00 00 00 00 00 00 00 00
$6260: 00 00 00 00 00 00 00 00
$6268: 00 00 00 00 00 00 00 00
$6270: 00 00 00 00 00 00 00 00
$6278: FF 0E 00 00 1F FE 00 00
$6280: FF FF 00 00 FF FF 00 00
$6288: FF FF 00 00 FF FF 00 00
$6290: FF FF 00 00 FF FF 00 00
$6298: FF FF 00 00 FF FF 00 00
$62A0: FF FF 00 00 FF FF 00 00
$62A8: FF FF 00 00 FF FF 00 00
$62B0: FF FF 00 00 FF FF 00 00
$62B8: 00 00 00 00 00 00 00 00
$62C0: 00 00 00 00 00 00 00 00
$62C8: 00 00 00 00 00 00 00 00
$62D0: 00 00 00 00 00 00 00 00
$62D8: 00 00 00 00 00 00 00 00
$62E0: 00 00 00 00 00 00 00 00
$62E8: 00 00 00 00 00 00 00 00
$62F0: 00 00 00 00 00 00 00 00
```

```
$62F8: 00 00 00 00 00 00 00 00
RAM-Disk-Driver $C400-$C4FF
$6300: 2C 82 C0 4C 59 FF 84 48
$6308: 85 49 8E 09 C0 84 00 85
$6310: 01 AD 11 C0 85 07 AD 12
$6318: C0 85 08 A0 01 B1 00 C9
$6320: 40 F0 0F 8E 08 C0 4C 04
$6328: BD A0 0D A9 40 91 00 38
$6330: B0 50 A0 0C B1 00 C9 01
$6338: F0 04 C9 02 D0 EB 85 09
$6340: A0 08 B1 00 85 04 C8 B1
$6348: 00 85 05 C9 C0 B0 DA C9
$6350: 02 90 D6 A0 04 B1 00 C9
$6358: 20 B0 CE C9 10 90 CA 0A
$6360: 0A 0A 0A C8 18 71 00 85
$6368: 03 C9 01 90 BC C9 C0 90
$6370: 19 C9 D0 B0 0F 09 10 85
$6378: 03 AE 8B C0 AE 8B C0 18
$6380: 90 08 B0 51 AE 83 C0 AE
$6388: 83 C0 A0 00 84 02 A5 09
$6390: C9 01 D0 0F 8E 03 C0 B1
$6398: 02 91 04 C8 D0 F9 8E 02
$63A0: C0 F0 0D 8E 05 C0 B1 04
$63A8: 91 02 C8 D0 F9 8E 04 C0
$63B0: 18 A0 0D A9 00 91 00 A5
$63B8: 08 30 08 AD 81 C0 AD 81
$63C0: C0 90 12 A5 07 10 08 AD
$63C8: 83 C0 AD 83 C0 90 06 AD
$63D0: 8B C0 AD 8B C0 8E 08 C0
$63D8: 60 00 00 00 00 00 00 00
```

SLOTRAMDISK.HELLO

```
10  PRINT CHR$ (4)"BLOAD SLOTRAMDISK"
11  S = 4: REM Slot 4; Werte 1-7
12  POKE 24739,S: REM Slot S
13  POKE 24810,96: REM RTS
14  CALL 24736: REM INIT J/N
15  PRINT "ERLEDIGT"
16  PRINT  CHR$ (4)"CATALOG,S"S
```

SLOTRAMDISK.

(Quellcode nur vom Driver)

```
                    1             ORG  $6ØAØ
6ØAØ: 4C A4 6Ø      3             JMP  SLOTPOKE
6ØA3: Ø4            4    SLOT     HEX  Ø4
                    5    *
                    6    * SLOTRAMDISK
                    7    * ============
                    8    *
                    9    * RAM-Diskdriver für 64K-Karte
                    1Ø   * für Apple IIe (DOS 3.3) mit
                    11   * Balfer-Interface. U. Stiehl '85
                    12   * Init-Programm aus Platzgründen
                    13   * nicht abgedruckt. Kompletter
                    14   * Quellcode auf Peeker-Sammeldisk
                    413  *=============================
                    414  *
                    415  * Eigentlicher Driver: relokativ!
                    416  *
                    417           ORG  $C4ØØ      ;$63ØØ!
                    418  *
                    419  IOBL     EQU  ZERO+$ØØ
                    42Ø  IOBH     EQU  ZERO+$Ø1
                    421  AUXL     EQU  ZERO+$Ø2
                    422  AUXH     EQU  ZERO+$Ø3
                    423  BUFL     EQU  ZERO+$Ø4
                    424  BUFH     EQU  ZERO+$Ø5
                    425  BANKFLAG EQU  ZERO+$Ø7
                    426  ROMFLAG  EQU  ZERO+$Ø8
                    427  RDWRFLAG EQU  ZERO+$Ø9
                    428  *
                    429  * Sprung zur Reset-Adresse im
                    43Ø  * ROM, falls aus Versehen PR#S
                    431  * eingegeben wurde.
                    432  *
C4ØØ: 2C 82 CØ      433  RESET    BIT  $CØ82
C4Ø3: 4C 59 FF      434           JMP  $FF59      ;Reset
                    435  *
                    436  * IOB-Pointer in Mainzp + Auxzp
                    437  *
C4Ø6: 84 48         438  ENTRY    STY  $48
C4Ø8: 85 49         439           STA  $49        ;Mainzp!
C4ØA: 8E Ø9 CØ      44Ø           STX  $CØØ9      ;AUXZP
C4ØD: 84 ØØ         441           STY  IOBL
C4ØF: 85 Ø1         442           STA  IOBH       ;Auxzp!
                    443  *
                    444  * Rette LC-Status
                    445  *
C411: AD 11 CØ      446           LDA  $CØ11      ;BK1/2
C414: 85 Ø7         447           STA  BANKFLAG
C416: AD 12 CØ      448           LDA  $CØ12      ;ROM/RAM
C419: 85 Ø8         449           STA  ROMFLAG
                    45Ø  *
                    451  * Pseudo-Disk-Slot?
                    452  *
C41B: AØ Ø1         453  SLOT?    LDY  #$Ø1
C41D: B1 ØØ         454           LDA  (IOBL),Y   ;Slot
                    455  *
                    456  * Per absolute Adresse gepokt!!!
                    457  *
C41F: C9 4Ø         458  SLOTPOK3 CMP  #$4Ø       ;Slot 4
C421: FØ ØF         459           BEQ  RDWR?
                    46Ø  *
                    461  * Zurück zur normalen Rwts
                    462  *
C423: 8E Ø8 CØ      463  RWTS     STX  $CØØ8      ;MAINZP
C426: 4C Ø4 BD      464           JMP  $BDØ4      ;Rwts
                    465  *
                    466  * Drive-Error
                    467  *
C429: AØ ØD         468  IOERROR  LDY  #$ØD
C42B: A9 4Ø         469           LDA  #$4Ø       ;IO-Err.
C42D: 91 ØØ         47Ø           STA  (IOBL),Y
C42F: 38            471           SEC             ;Carry!
C43Ø: BØ 5Ø         472           BCS  EXBRANCH   ;IO-Error
                    473  *
                    474  * Befehl: 1=Read oder 2=Write?
                    475  *
C432: AØ ØC         476  RDWR?    LDY  #$ØC
C434: B1 ØØ         477           LDA  (IOBL),Y   ;Befehl
C436: C9 Ø1         478           CMP  #$Ø1       ;Read
C438: FØ Ø4         479           BEQ  PUFFER
C43A: C9 Ø2         48Ø           CMP  #$Ø2       ;Write
C43C: DØ EB         481           BNE  IOERROR
                    482  *
```

```
                    483  * Pufferadresse: $Ø2ØØ-$BFØØ
                    484  *
C43E: 85 Ø9         485  PUFFER   STA  RDWRFLAG   ;RD/WR
C44Ø: AØ Ø8         486           LDY  #$Ø8
C442: B1 ØØ         487           LDA  (IOBL),Y   ;Puff-L
C444: 85 Ø4         488           STA  BUFL
C446: C8            489           INY
C447: B1 ØØ         49Ø           LDA  (IOBL),Y   ;Puff-H
C449: 85 Ø5         491           STA  BUFH
C44B: C9 CØ         492           CMP  #$CØ       ;>$CØØØ?
C44D: BØ DA         493           BCS  IOERROR    ;No LC!
C44F: C9 Ø2         494           CMP  #$Ø2       ;<$Ø2ØØ?
C451: 9Ø D6         495           BCC  IOERROR    ;Zero-P.
                    496  *
                    497  * Track und Sektor in Bereich
                    498  * der 64K-Karte umwandeln
                    499  *
                    5ØØ  * T. $1Ø -> $ØØØØ (nur teils)
                    5Ø1  * T. $11 -> $1ØØØ
                    5Ø2  * T. $12 -> $2ØØØ
                    5Ø3  * T. $13 -> $3ØØØ
                    5Ø4  * T. $14 -> $4ØØØ
                    5Ø5  * T. $15 -> $5ØØØ
                    5Ø6  * T. $16 -> $6ØØØ
                    5Ø7  * T. $17 -> $7ØØØ
                    5Ø8  * T. $18 -> $8ØØØ
                    5Ø9  * T. $19 -> $9ØØØ
                    51Ø  * T. $1A -> $AØØØ
                    511  * T. $1B -> $BØØØ
                    512  * T. $1C -> $CØØØ wird Bk1 $DØØØ
                    513  * T. $1D -> $DØØØ Bk2
                    514  * T. $1E -> $EØØØ Bk2
                    515  * T. $1F -> $FØØØ Bk2
                    516  *
C453: AØ Ø4         517           LDY  #$Ø4
C455: B1 ØØ         518           LDA  (IOBL),Y   ;Track
C457: C9 2Ø         519           CMP  #$2Ø
C459: BØ CE         52Ø           BCS  IOERROR    ;>$1F
C45B: C9 1Ø         521           CMP  #$1Ø
C45D: 9Ø CA         522           BCC  IOERROR    ;<$1Ø
                    523  *
C45F: ØA            524           ASL             ;shiften
C46Ø: ØA            525           ASL
C461: ØA            526           ASL
C462: ØA            527           ASL
                    528  *
C463: C8            529           INY             ;Y=$Ø5
C464: 18            53Ø           CLC
C465: 71 ØØ         531           ADC  (IOBL),Y   ;Sektor
C467: 85 Ø3         532           STA  AUXH
C469: C9 Ø1         533           CMP  #$Ø1       ;>=$Ø1ØØ?
C46B: 9Ø BC         534           BCC  IOERROR
                    535  *
                    536  * Bank 1? $CØØØ -> $DØØØ
                    537  *
C46D: C9 CØ         538  BANK1?   CMP  #$CØ       ;$CØØØ
C46F: 9Ø 19         539           BCC  READ?      ;<$CØØØ
C471: C9 DØ         54Ø           CMP  #$DØ       ;$DØØØ
C473: BØ ØF         541           BCS  BANK2      ;>=$DØØØ
C475: Ø9 1Ø         542           ORA  #%ØØØ1ØØØØ ;Cx->Dx
C477: 85 Ø3         543           STA  AUXH
C479: AE 8B CØ      544           LDX  $CØ8B
C47C: AE 8B CØ      545           LDX  $CØ8B
C47F: 18            546           CLC
C48Ø: 9Ø Ø8         547           BCC  READ?      ;stets
                    548  *
C482: BØ 51         549  EXBRANCH BCS  RAMEXIT    ;IO-Error
                    55Ø  *
C484: AE 83 CØ      551  BANK2    LDX  $CØ83
C487: AE 83 CØ      552           LDX  $CØ83
                    553  *
C48A: AØ ØØ         554  READ?    LDY  #Ø         ;Y=Ø
C48C: 84 Ø2         555           STY  AUXL
C48E: A5 Ø9         556           LDA  RDWRFLAG
C49Ø: C9 Ø1         557           CMP  #1         ;1=Read
C492: DØ ØF         558           BNE  WRITE
                    559  *
                    56Ø  * Read = Aux. nach Main
                    561  *
C494: 8E Ø3 CØ      562  READ     STX  $CØØ3      ;AUXRD
C497: B1 Ø2         563  READER   LDA  (AUXL),Y
C499: 91 Ø4         564           STA  (BUFL),Y
C49B: C8            565           INY
C49C: DØ F9         566           BNE  READER
C49E: 8E Ø2 CØ      567           STX  $CØØ2      ;MAINRD
C4A1: FØ ØD         568           BEQ  ZURUECK1
                    569  *
```

```
                570  * Write = Main nach Aux.
                571  *
C4A3: 8E Ø5 CØ  572  WRITE    STX  $CØØ5      ;AUXWR
C4A6: B1 Ø4     573  WRITER   LDA  (BUFL),Y
C4A8: 91 Ø2     574           STA  (AUXL),Y
C4AA: C8        575           INY
C4AB: DØ F9     576           BNE  WRITER
C4AD: 8E Ø4 CØ  577           STX  $CØØ4      ;MAINWR
                578  *
C4BØ: 18        579  ZURUECK1 CLC              ;okay!
C4B1: AØ ØD     58Ø           LDY  #$ØD
C4B3: A9 ØØ     581           LDA  #$ØØ
C4B5: 91 ØØ     582           STA  (IOBL),Y   ;o.Fehler
                583  *
C4B7: A5 Ø8     584           LDA  ROMFLAG
C4B9: 3Ø Ø8     585           BMI  ZURUECK2
C4BB: AD 81 CØ  586           LDA  $CØ81      ;RDROM
C4BE: AD 81 CØ  587           LDA  $CØ81      ;WRBK2
C4C1: 9Ø 12     588           BCC  RAMEXIT
C4C3: A5 Ø7     589  ZURUECK2 LDA  BANKFLAG
C4C5: 1Ø Ø8     59Ø           BPL  ZURUECK3
C4C7: AD 83 CØ  591           LDA  $CØ83      ;BK2
C4CA: AD 83 CØ  592           LDA  $CØ83      ;RDWR
C4CD: 9Ø Ø6     593           BCC  RAMEXIT
C4CF: AD 8B CØ  594  ZURUECK3 LDA  $CØ8B      ;BK1
C4D2: AD 8B CØ  595           LDA  $CØ8B      ;RDWR
                596  *
C4D5: 8E Ø8 CØ  597  RAMEXIT  STX  $CØØ8      ;MAINZP
C4D8: 6Ø        598           RTS
```

APPLE II PASCAL

Betriebssystem und Sprache

Erstes deutsches Referenzwerk sämtlicher Befehle und Systemroutinen von Apple II Pascal – mit Addendum einschließlich Version Pascal 1.2!

Gültig für Apple II, II Plus, IIe einschließlich der 128K/80 Zeichen-Konfiguration.

Betriebssystem kommentiert ausführlich und in Deutsch Funktion und Benutzung der fast 60 Systemroutinen des Apple II Pascal Betriebssystems.

Sprache ist das vollständige, deutsche Referenzwerk der „Apple Pascal"-Programmiersprache mit u. a. Informationen über professionelle Pascal-Programmierung, Turtlegraphics, Programmbibliothek etc.

In Vorbereitung: Addendum Pascal 1.2, ein Zusatz zum Buch „Betriebssystem" für 1.2-Benutzer in Deutsch.

„Nach Unterlagen von Apple Deutschland hergestellt"

Apple II Betriebssystem,
272 Seiten, DM 49,–

Apple II Sprache,
216 Seiten, DM 39,–

Pascal 1.2 Addendum, etwa 100 Seiten,
DM 36,–

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,–

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,–

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

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

tm 4153

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

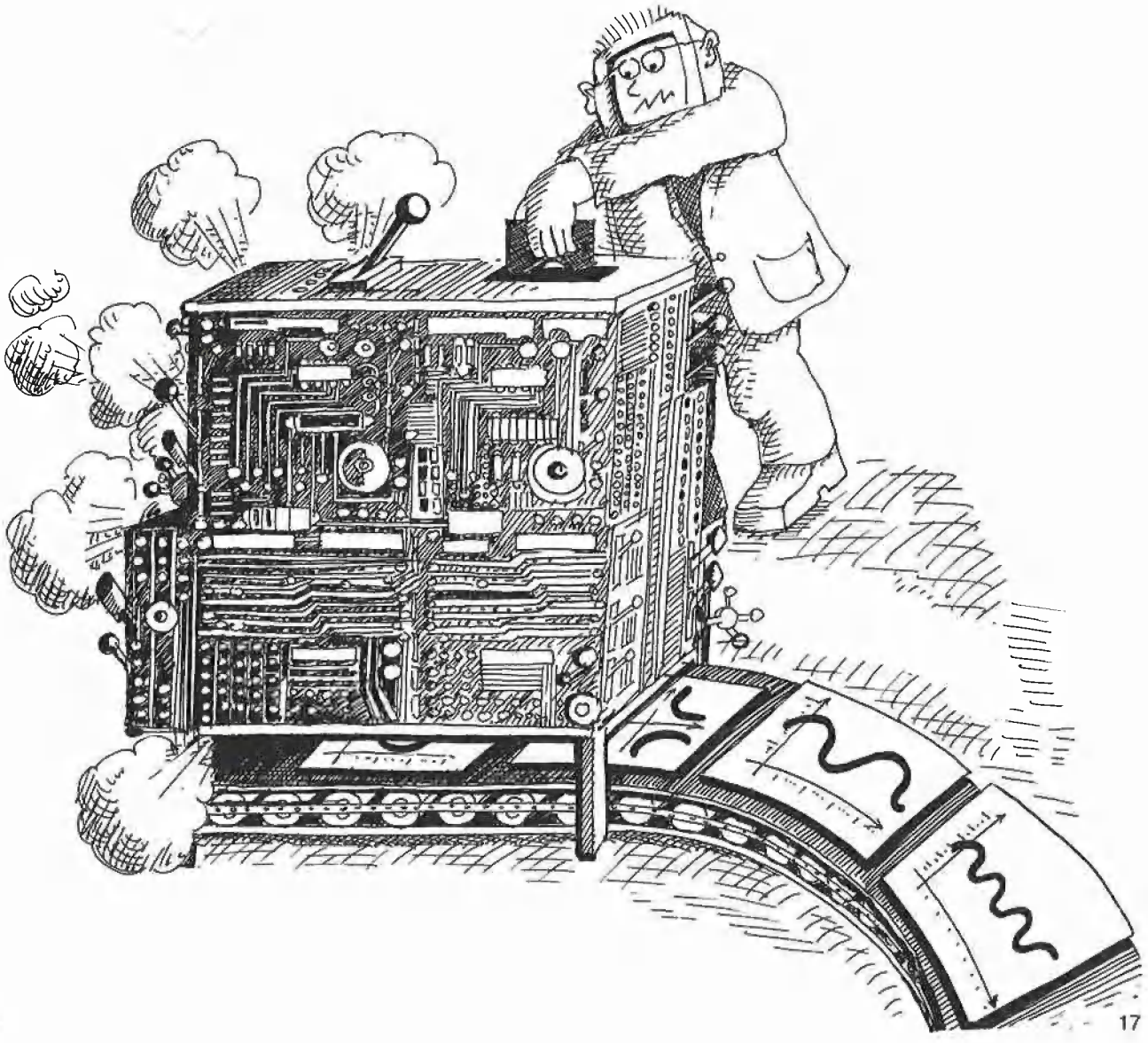
In Vorbereitung:
Macintosh Programmier-Handbuch DM 59,–

PLOT 2.0

Ein komfortabler Funktionsplotter

von Hans-Martin Eng

Kennen Sie auch das folgende Problem? Vor Ihnen liegt eine mathematische Funktion und Sie können nichts damit anfangen. Hilfreich wäre es jetzt, wenn man wüßte, wie deren grafische Darstellung aussieht. Wer jedoch schüttelt sich eine fertige Kurvendiskussion aus dem Handgelenk und hat dann auch die Gewißheit, daß alles stimmt? Und wer wollte schon die Funktion Punkt für Punkt mit der Hand malen? Aber da Sie einen Apple besitzen, kann ich Ihnen Abhilfe versprechen. Das nachstehende Plotprogramm nimmt Ihnen das Zeichnen ab.



Was kann PLOT 2.0?

Bei der Erstellung wurden folgende Anforderungen berücksichtigt: Das Programm soll
– die Eingabe der Funktion während des laufenden Programms gestatten,
– weitgehend menügesteuert ablaufen,
– sogenannte Scharparameter (r, s und t) unterstützen und die Funktion in Abhängigkeit von diesen Parametern mehrmals zeichnen,
– die Funktion in einem üblichen Achsenkreuz darstellen, das mit einer Skaleneinteilung versehen ist,
– „Pole" und andere Stellen, an denen die Funktion nicht definiert ist, intern abfangen,
– während die Funktion gezeichnet wird, eine Zusammenfassung wichtiger Daten auf der Textseite bereithalten und auf Tastendruck wiedergeben,
– auf Druckern mit Grafikinterface die Grafik nebst sämtlichen verwendeten Darstellungsparametern ausdrucken.
Unter Berücksichtigung dieser Vorgaben entstand das Applesoft-Basicprogramm **PLOT.2.0** mit dem Assemblerprogramm **PLOT.B**. PLOT.2.0 lädt sich selbst in den Speicherbereich oberhalb der ersten Grafikseite ($4001), dann PLOT.B in den Bereich zwischen $0D00 und $0E0E. Das Programm belegt ca. 13400 Bytes. Dabei ist der Speicherbedarf der Variablen und des Maschinenprogramms nicht berücksichtigt.

Die Theorie des Plottens

Hier einige Erläuterungen zum Programm und den darin auftauchenden Formeln und Algorithmen:
Der Zusammenhang zwischen Bildschirm- und normalen Koordinaten ist linear. So können die Bildschirmkoordinaten als einfache Funktionen xs (x), ys (y) wiedergegeben werden. Allgemein kann der Zusammenhang zwischen Bildschirm- und Normalkoordinaten wie folgt definiert werden:

$$\frac{s-s_L}{n-n_L}=\frac{s_H-s_L}{n_H-n_L}$$

s: Schirmkoordinate
n: Normalkoordinate

Indizes: H–größter Wert, L–niedrigster Wert

Eine einfache Umformung liefert:

$$s=\frac{s_H-s_L}{n_H-n_L}n+\frac{n_Hs_L-s_Hn_L}{n_H-n_L}$$

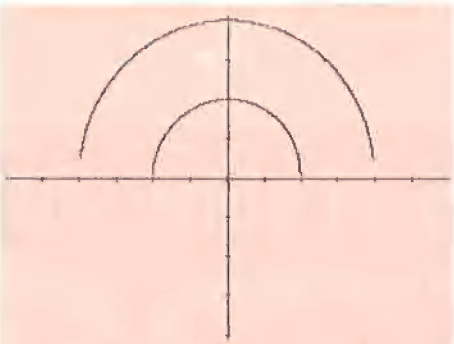
Im besonderen gilt im Falle des Hires-Bildschirms für die x-Koordinaten:

$s_H = 271$; $s_L = 0$; n_H, $n_L \rightarrow$ xH, xL

und für die y-Koordinaten:

$s_H = 0$; $s_L = 191$; n_H, $n_L \rightarrow$ yH, yL

Nun erfolgt noch die Umbenennung der Terme. Der Faktor vor dem n wird durch MX oder MY ersetzt, der Summand durch BX oder BY (analog zur Normaldarstellung der Geradengleichung: $y = m \cdot x + b$). Somit erhalten wir die Darstellung der Funktionen XS (X) und YS (X) in Zeile 3520 des Programms.



Die Halbkreisfunktion F(X) = SQR (R ∗ R – X ∗ X) im Intervall [-6,6]. Der Parameter R nimmt die Werte 2 und 4 ein.

Vielfältige Darstellungsmöglichkeiten

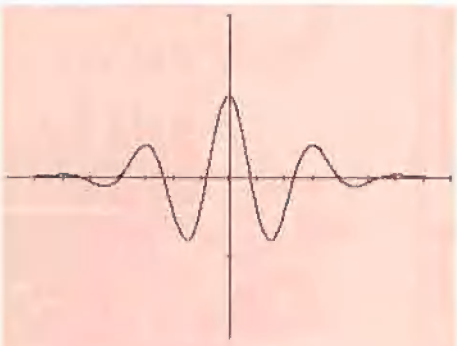
Der Darstellungsmaßstab wird durch Eingabe der Darstellungsgrenzen festgelegt. Da man jedoch beispielsweise bei der Halbkreisfunktion gerne einen Halbkreis sehen möchte und nicht irgendeine (wenn auch hübsche) Ellipse, wurde die Möglichkeit geschaffen, den Maßstab der x-Achse dem der y-Achse anzupassen. Diesen Wunsch signalisiert man durch die Eingabe „0,0" oder „," für eine Achse. Das Ergebnis ist jedoch oft noch nicht befriedigend. Daher können noch nachträglich kleine Veränderungen am Maßstab vorgenommen werden. Man kann die Achsen strecken oder stauchen und außerdem das Verhältnis der positiven zur negativen Teilachse beeinflussen. Das geschieht durch die Eingabe zweier Zahlen. Bei der Streckung gibt man an, in welchem Verhältnis die alten zu den neuen Grenzen stehen sollen (2,1 heißt, daß die neuen Grenzen halb so groß wie die alten sind). Ändert man das Verhältnis der Teilachsen zueinander, bleibt der Maßstab unberührt. Nachdem die Anpassung durchgeführt ist, verhält sich der positive Teillast zum negativen wie die zuerst eingegebene Zahl zur zweiten (z.B. vorher: XL = -3, XH = 3, Eingabe: 2,1; danach: XL = -2, XH = 4).
Die Funktion selbst wird wie eine Basic-Zeile eingegeben, wobei das „DEF FN Y(X) =" schon vorgegeben ist. Hier ein Beispiel:

Y = SIN (X) ∗ T

Nach der Eingabe werden Sie gefragt, ob alles stimmt. Dies ist der kritischste Punkt. Ein Syntaxfehler führt zum Programmabbruch, wenn die Funktion zum ersten Mal verwendet wird.

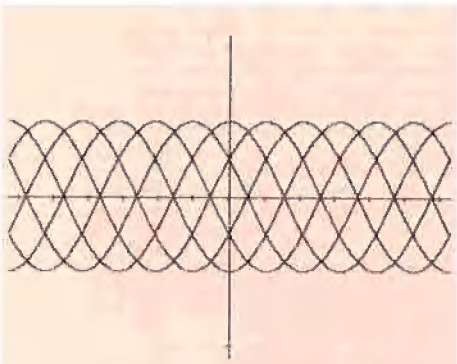
Scharparameter

Im Programm sind die Variablen R, S und T (r, s und t) als Scharparameter vorgesehen. Diese machen meines Erachtens manche Funktionen erst interessant. Bestes Beispiel: f(x) = exp (t·x). Für jedes t hat diese Funktionenschar einen gemeinsamen Punkt in (0,1). Oder f(x) = sin (x-t): Ist t ein gerades Vielfaches von π, sind beispielsweise die Schaubilder deckungsgleich. Jeweils nach der Eingabe einer Funktion wird gefragt, ob, und wenn ja, welche Parameter in der Funktion verwendet werden. Wird speziell T verwendet, bietet das Programm die Option an, T in einer FOR-NEXT-Schleife zu durchlaufen. So kann man sich lästiges Neueintippen sparen.
Das Programm legt die Parameterwerte in einer Art Stack ab. Dazu wird der Pointer I



Die Exponentialfunktion F(X) = S ∗ EXP (– R ∗ X ∗ X) ∗ COS (T ∗ X) zwischen den Grenzen -8 und 8 (R = 0,1; S = 1; T = 2).

erhöht, wenn eine Kurve bis zum rechten Rand durchgezogen ist. Dann kommen die Parameterwerte in das Array P und die Stelle (*, I). I wird wieder auf 0 gesetzt, wenn der Darstellungsmaßstab oder die Funktion geändert oder die Grafik gelöscht wird. Beim Ausdrucken werden die Elemente des Arrays bis zum aktuellen Pointerwert ausgegeben.
Wenn Sie die Kreiszahl π oder die Eulersche Zahl e benötigen, müssen Sie diese nicht als Konstanten in die Funktion schreiben. Die Variablen E und PI werden am Anfang des Programms auf den richtigen Wert gesetzt, Sie können sie also in die Funktion einbauen. Ebenso stellt das Programm den Briggsschen (Zehner-)Logarithmus als FN LG(X) zur Verfügung.



*Die Funktion F(X) = COS (X – T * PI/3) im Bereich –6,3; 6,3. Der Parameter T durchläuft die Werte 0, 1, 2, 3, 4, 5.*

Die Achsen

Das Unterprogramm zum Achsenzeichnen (4900 ff.) zieht die y-Achse zwischen xs(0), 0 und xs(0), 191; die x-Achse zwischen 0, ys(0) und 271, ys(0). Dann wird festgestellt, ob eine der Achsen am Rand des Bildschirms zu liegen kommt. Ist dies der Fall, wird eine Korrekturvariable gesetzt, die dafür sorgt, daß der Skalenstrich um zwei Bildschirmraster vom Rand weg verschoben wird. Dann wird der betragsgrößte Wert der Darstellungsgrenzen einer Achse an ein Unterprogramm (3620 ff.) übergeben. Dieses sorgt dafür, daß der längere Teilast einer Koordinatenachse maximal zehn Skalenstriche aufweist.

Druckerausgabe

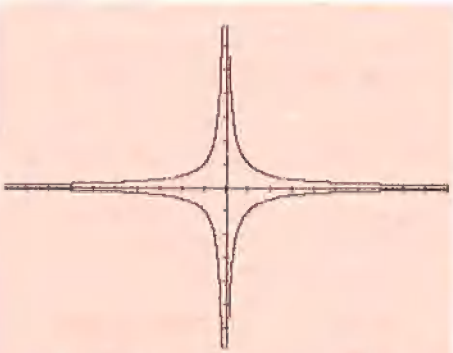
Die Printroutine (5100 ff.) ist auf einen **EPSON FX-80** mit Vollgrafikinterface zugeschnitten. Daher erkläre ich besser, was die Steuerbefehle bedeuten:
PRINT CHR$(9); CHR$(16) stellt den Command-Character des Interface auf CHR$(16) um.
ESC M stellt Schönschrift mit 96 Zeichen/Zeile ein.
ESC I setzt den linken Rand auf Spalte 6.
ESC Q setzt den rechten Rand auf Spalte 90.
ESC A setzt einen höheren Zeilenabstand.
ESC D definiert Tabulatorpositionen.
ESC N bewirkt, daß der Drucker die Perforation überspringt.
CHR$ (9) befördert den Druckkopf auf die nächste Tabulatorposition.
PRINT CHR$ (16);"G" veranlaßt das Interface, die Grafik in Normalgröße auszudrucken.

Fehlerbehandlung

Zur Fehlerbehandlungsroutine muß nicht viel angemerkt werden. Zuerst wird eine Maschinenroutine aufgerufen, die ein „Überlaufen" des Stacks verhindert. Dann wird die Art des Fehlers überprüft. Wurde Ctrl-C gedrückt, springt das Programm ins Hauptmenü ab Zeile 1800. Anwenderfehler beim Input bewirken dasselbe wie im normalen Betrieb. Tritt ein „Division by Zero"-, „Illegal Quantity"- oder „Overflow"-Error auf, wird ein neuer Anfangswert von X gesetzt und die Zeichenschleife neu gestartet. Bei allen anderen Fehlern wird auf Textdarstellung umgeschaltet und eine Fehlermeldung ausgegeben. Achtung! Tritt ein Syntaxfehler in Zeile 1680 auf, so ist meist ein Eingabefehler bei der Funktionseingabe schuld. Prüfen Sie deshalb genau nach, ob alle geöffneten Klammern auch wieder geschlossen werden.

Programmverschiebung

Die Zeilen 5600 ff. treten in Aktion, wenn das Programm nicht im Speicherbereich oberhalb der ersten Grafikseite liegt. Dann wird der Execfile **PLOT.PROTECTOR** aufgerufen. Dieser Execfile sollte die folgenden Befehle enthalten:
POKE 103,1:POKE 104,64:POKE 16384,0 (setzt den Pointer zum Anfang des Applesoftprogramms auf $4001 und die Adresse $4000 auf 0.)
LOAD **PLOT.2.0**, CALL 54514 (diese Routine setzt die übrigen Pointer richtig), RUN.



Die Funktion F(X) = T/X mit T = -1 und 1 im Intervall [-10, 10].

Alternativ dazu kann man auch ein kurzes Programm schreiben, das den Execfile direkt aufruft. Das kann erhebliche Zeitvorteile bringen, da das zweimalige Laden des Hauptprogramms doch einige Zeit in Anspruch nimmt.

Die Assemblerroutinen

Das Assemblerprogramm gliedert sich in vier Teile auf. Der erste Teil besteht aus dem Unterprogramm, das die darzustellende Funktion einliest, in den Basiccode übersetzt und in die Zeile 2 des Basicprogramms überträgt. Der zweite Teil übersetzt den Inhalt dieser Zeile zurück in ASCII-Code und schreibt alles in einen Puffer. Der dritte Teil ist eine Hilfsroutine für die Fehlerbehandlung in Zeile 5500 ff. Der vierte Teil besteht aus Tabellen. Den Eingabeteil (Zeilen 45 bis 95) übernahm ich aus einem im CHIP Special 2/80 veröffentlichten Programm. Ich erlaubte mir, das dort nur vom Miniassembler disassemblierte Programm mit einem besseren Assemblerprogramm in eine lesbarere Form zu bringen. Außerdem mußte ich einen relativ bösen Tippfehler des Autors korrigieren. Mir passierte es nämlich zu Anfang, wenn sich im Speicher noch eine alte, längere Funktion befand, daß diese alte Funktion nicht vollständig gelöscht wurde. Dies führte zu Syntaxfehlern im Basicprogramm. Des weiteren fügte ich Zeile 45 ein, da das sich anschließende Rückübersetzungsprogramm sehr regen Gebrauch von den Registern macht.
Diese Routine beginnt mit Zeile 100. Sie nimmt ein Byte aus Zeile 2 und verzweigt dann, je nachdem, ob es sich um ein Basic-Token oder ein ASCII-Zeichen handelt. Ein ASCII-Zeichen wird direkt in den Puffer geschrieben. Bei einem Token wird dessen Wert mit denen der Tabelle 2 verglichen, dann wird aus Tabelle 3 der zuge-

hörige Pointer entnommen. Dieser Pointer zeigt auf das Leerzeichen am Anfang der zugeordneten Zeichenkette. Das nächste Leerzeichen signalisiert das Ende des Strings. Ist die Funktion als Ganzes übersetzt, steht in STRPNTL die Länge des Funktionsstrings im Puffer. Zum Schluß werden die von Zeile 45 geretteten Register wieder zurückgespeichert, dann springt das Unterprogramm wieder in das Basicprogramm.

Eine Bemerkung zur **Token-Tabelle**. Ich verzichtete bewußt darauf, alle möglichen reservierten Wörter zu berücksichtigen. Ich sehe nicht ganz ein, was ein FRE(0) in einer mathematischen Funktion zu suchen hat. Wer jedoch Wert auf irgendeinen Exoten legt, sollte diesen zur Tabelle 1 hinzufügen, dann abzählen, ab dem wievielten Byte nach der Startadresse der Tabelle 1 das Space steht und diesen Wert zur Tabelle 3 hinzufügen. Anschließend ist noch in Tabelle 2 das Token einzutragen und der Wert von $0D6B um 1 zu erhöhen. Findet die Übersetzungsroutine ein unbekanntes Token, wird ein „?“ in den Puffer geschrieben.

Die Maschinenroutine in den Zeilen 164 bis 172 erwies sich als nötig, weil manche Funktionen (etwa die Halbkreisfunktion) über einen weiten Bereich keinen reellen Wert annehmen. Treten jedoch solche Fehler gehäuft auf, läuft der Stack scheinbar über und es tritt ein „Out of Memory Error“ auf. Die genannte Routine bereinigt dieses Problem auf einfache Weise. Ich habe das Programm selbst durch ein ständiges F(X) = SQR(-1) nicht aus der Fassung bringen können.

SHTABLE schließlich enthält einen kleinen Strich als Shape. Dieser Strich bildet in der Grafik die Skaleneinteilung.

So, nun bleibt Ihnen nur noch übrig, das Programm in Ihren Apple einzuprogrammieren oder die entsprechende „Peeker“-Sammeldiskette zu bestellen. Ich wünsche Ihnen viel Spaß beim Plotten Ihrer Funktionen.

Anmerkung: Beim Eintippen der Zeile 2 ist darauf zu achten, daß exakt 228 Doppelpunkte getippt werden. Das Zählen kann man sich sparen, wenn man die Zeile *ohne* Leerzeichen bis zum ersten Piepston eingibt.

PLOT.2.0

```
1     GOTO 1000
2     DEF FN Y (X) = ::::::::::::::::::::::::::::::::::::::::
      ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      ::::::::::::::::::::::::::::::::
3     RETURN
4     REM Bitte erhöhte Vorsicht beim Ändern der ersten
      Programmzeilen walten lassen!
5     REM Dies kann zu bösen Überraschungen bis hin zum
      Programmverlust führen!
1000 IF PEEK (104) < 64 THEN 5600
1010 ONERR GOTO 5500: REM diverse Konstanten
1020 PI = ATN (1) * 4:E = EXP (1):CH% = 36:CL% = -
     868:KE% = - 16384:AP% = 3072
1030 HT$ = CHR$ (9):E$ = CHR$ (27):D$ = CHR$ (4):DP$ =
     CHR$ (16): DIM P(2,29): DEF FN LG(X) = LOG (X) / LOG
     (10)
1040 AS% = 3328: IF PEEK (AS%) < > 32 OR PEEK (AS% + 20)
     < > 213 OR PEEK (AS% + 270) < > 57 THEN PRINT
     D$;"BLOAD PLOT.B,A";AS%
1100 POKE 233,13: POKE 232,162: REM SHAPE-TABLE
1110 HOME : PRINT TAB( 16);"PLOT.2.0"
1120 VTAB 10: PRINT TAB( 11);"Von Hans-Martin Eng"
1130 VTAB 12: PRINT TAB( 13);"September 1983"
1140 VTAB 14: PRINT TAB( 15);"Erweitert :"
1150 VTAB 16: PRINT TAB( 14);"Juni/Juli 1984"
1160 VTAB 18: PRINT TAB( 11);"Letzte Änderungen :"
1170 VTAB 20: PRINT TAB( 11);"Oktober/November 1984";
1180 VTAB 21: POKE - 16368,0: GET A$
1190 HOME : PRINT "Wünschen Sie eine Kurzeinführung ?":
     PRINT "Betätigen Sie bitte 'J' oder 'N'"
1200 GOSUB 4700: IF A$ = "N" THEN 1330
1210 HOME : PRINT "Dieses Programm stellt Funktionen in":
     PRINT "hochauflösender Grafik auf dem Bild-": PRINT
     "schirm dar. Etwa auftretende Polstellen": PRINT
     "werden intern abgefangen, der Darstel-"
1220 PRINT "lungsmaßstab ist durch Wahl der Darstel-":
     PRINT "lungsgrenzen (relativ) frei festlegbar."
1230 PRINT "Die Ludolfsche Zahl (Pi) und die Euler-":
     PRINT "sche Zahl (e) können ebenso, wie": PRINT
     "Funktionsparameter (r,s,t) in der Funk-": PRINT
     "tion verwendet werden. Auf diese Weise"
1240 PRINT "sind Funtionsscharen darstellbar. Die": PRINT
     "Scharparameter werden vor dem Zeichnen": PRINT
     "abgefragt."
1250 PRINT "zum Beispiel :": PRINT " Y=R * SIN (2 * PI /
     S * X - T)"
1260 PRINT "oder :": PRINT " Y = (X - T) / (X * X +1) *
     (E - EXP(X))"
1270 PRINT "Der Parameter 'T' kann sukzessive mit": PRINT
     "diskreten Werten durchlaufen werden.": PRINT "Wenn
     Sie 'T' verwenden, werden Sie ge-": PRINT "fragt, ob
     Sie von dieser Möglichkeit"
1280 PRINT "Gebrauch machen wollen.";: GET A$: PRINT
1290 PRINT "Nach jeder Eingabe werden Sie gefragt,":
     PRINT "ob Eingabefehler aufgetreten sind, Sie":
     PRINT "können dann gegebenenfalls korrigieren."
1300 PRINT "Während die Funktion gezeichnet wird,": PRINT
     "kann man durch Betätigung einer Taste": PRINT "auf
     den Textbildschirm umschalten. Dort": PRINT "steht
     eine Kurzzusammenfassung wichtiger"
1310 PRINT "Daten. Nochmaliges Betätigen einer Tas-":
     PRINT "te läßt wieder die Grafik erscheinen."
1320 PRINT "Wenn Sie <Ctrl-C> betätigen, springt": PRINT
     "das Programm ins Hauptmenü. Dort kön-": PRINT "nen
     etwa aufgetretene Eingabefehler": PRINT "korigiert
     und die Funktion neu gezeich-": PRINT "net werden.":
     GET A$
1330 GOSUB 3000: GOSUB 3700: GOSUB 3060
1340 REM Zusammenfassung
1350 HOME : PRINT "Zusammenfassung :"
1360 PRINT "Die Funktion lautet :": PRINT : PRINT F$
1370 PRINT : PRINT : PRINT "Parameter ";: ON VAL (P$)
     GOTO 1390,1400,1410,1420,1430,1440,1450
1380 PRINT : PRINT "Es werden keine Parameter
     verwendet.": GOTO 1470
1390 PRINT "R": GOTO 1470
1400 PRINT "S": GOTO 1470
1410 PRINT "T";: GOTO 1460
1420 PRINT "R und S": GOTO 1470
1430 PRINT "R und T";: GOTO 1460
1440 PRINT "S und T";: GOTO 1460
1450 PRINT "R,S und T";
1460 IF RIGHT$ (P$,1) = "K" THEN PRINT " / T in Schleife"
1470 PRINT : PRINT : PRINT "Grenzen :": PRINT TAB( 5);"X
     : ";XL;" ... ";XH
1480 PRINT TAB( 5);"Y : ";YL;" ... ";YH
1490 VTAB 23: PRINT TAB( 20);"Noch Fehler  ? ";: GOSUB
     4700: IF A$ = "N" THEN 1590
1500 HOME : PRINT "Wo traten noch Fehler auf ?"
1510 VTAB 6: HTAB 10: PRINT "1 - Funktion"
1520 VTAB 10: HTAB 10: PRINT "2 - Parameter"
1530 VTAB 14: HTAB 10: PRINT "3 - Darstellungsgrenzen"
1540 VTAB 18: HTAB 10: PRINT "0 - keine Fehler"
1550 VTAB 23: HTAB 13: PRINT "Betreffende Taste drücken";
1560 GET A$: IF VAL (A$) = 0 AND A$ < > "0" OR VAL (A$) >
     3 THEN 1560
1570 ON VAL (A$) GOSUB 3000,3700,3060
1580 GOTO 1350
1590 REM Jetzt wird gezeichnet
1600 GOSUB 4410: REM Parametereingabe
1610 HGR : POKE - 16302,0: POKE 253,2: HCOLOR= 3: SCALE=
     1: GOSUB 4900
1620 X0 = 0:ST = 1 / MX: REM Initialisierung
1630 IF RIGHT$ (P$,1) = "K" THEN FOR T = T0 TO T1 STEP
     TS: REM Starte T-Schleife
1640 GOSUB 4600:XA = XL: REM Generiere Liste in Textseite
1650 X = XA: IF X > XH THEN 1750: REM Start x-Schleife
1660 IF PEEK (KE%) > 128 THEN GOSUB 4790: REM umschalten?
1670 VTAB 23: HTAB 14: PRINT X; SPC( 10);: REM x in Liste
1680 IF X0 = 0 THEN X0 = FN XS(X):Y0 = FN YS( FN Y(X)):
     GOTO 1750: REM Ausgangspunkt
```

```
169Ø Xl = FN XS(X):Yl = FN YS( FN Y(X)): REM Endpunkt
17ØØ IF Yl < Ø THEN Yl = Ø: REM gegen 'Illegal Quantity'
171Ø IF Yl > 191 THEN Yl = 191
172Ø IF Yl = 191 AND YØ = 191 OR Yl = Ø AND YØ = Ø THEN
     XØ = Ø: GOTO 175Ø: REM => Pol
173Ø IF Yl = Ø AND YØ = 191 OR Yl = 191 AND YØ = Ø THEN
     XØ = Ø: GOTO 168Ø: REM Keine Verbindung zwischen
     Polen
174Ø HPLOT XØ,YØ TO Xl,Yl:XØ = Xl:YØ = Yl: REM Zeichnen /
     neuer Ausgangspunkt
175Ø IF X < XH THEN X = X + ST: GOTO 166Ø
176Ø I = I + l:P(Ø,I) = R:P(l,I) = S:P(2,I) = T: REM
     Parameter-Liste
177Ø IF RIGHT$ (P$,1) = "K" THEN XØ = Ø: NEXT T: REM s.
     163Ø
178Ø GET A$: REM Warteschleife
179Ø POKE 253,1: GOSUB 48ØØ: HOME : REM Text / Hauptmenü
18ØØ PRINT "Wollen Sie :": VTAB 6: PRINT "1 -
     Darstellungsgrenzen ändern"
181Ø VTAB 8: PRINT "2 - Parameter ändern"
182Ø VTAB 1Ø: PRINT "3 - Funktion ändern"
183Ø VTAB 12: PRINT "4 - Bildschirm säubern"
184Ø VTAB 14: PRINT "5 - Alte Daten auflisten"
185Ø VTAB 16: PRINT "6 - Wieder zeichnen"
186Ø VTAB 18: PRINT "7 - ausdrucken"
187Ø VTAB 23: PRINT "8 - aufhören"
188Ø GET A$: IF A$ = " " THEN GOSUB 48ØØ: GOTO 188Ø
189Ø IF VAL (A$) < 1 OR VAL (A$) > 8 THEN 188Ø
19ØØ ON VAL (A$) GOTO 192Ø,195Ø,198Ø,2Ø1Ø,2Ø4Ø,2Ø7Ø,
     21ØØ,213Ø
191Ø REM 1 Darstellungsgrenzen
192Ø GOSUB 3Ø6Ø: HGR : POKE - 163Ø2,Ø: HCOLOR= 3: GOSUB
     49ØØ: GOTO 179Ø
193Ø REM Eingabe Grenzen/zeichne Achsen und skaliere/Menü
194Ø REM 2 Parametereingabe
195Ø GOSUB 44ØØ: GOTO 179Ø
196Ø REM Eingabe Parameter/Menü
197Ø REM 3 Funktionseingabe
198Ø GOSUB 3ØØØ: GOSUB 37ØØ: GOSUB 441Ø: GOTO 179Ø
199Ø REM Eingabe Funktion/Parameter?/Parametereingabe/
     Menü
2ØØØ REM 4 Bildschirm
2Ø1Ø I = Ø: HGR : POKE - 163Ø2,Ø: GOSUB 49ØØ: GOTO 179Ø
2Ø2Ø REM Zeichne Achsen und Skala/Menü
2Ø3Ø REM 5 Datenliste zeigen
2Ø4Ø GOSUB 46ØØ: GOTO 178Ø
2Ø5Ø REM Zusammenfassung / Menü
2Ø6Ø REM 6 Zeichnen
2Ø7Ø POKE 253,2: GOSUB 48ØØ: GOTO 162Ø
2Ø8Ø REM Grafikseite/Menü
2Ø9Ø REM 7 Ausdrucken
21ØØ POKE 253,2: GOSUB 48ØØ: GOSUB 51ØØ: GOTO 179Ø
211Ø REM Grafikseite/Ausdruck/Menü
212Ø REM 8 Programmende
213Ø TEXT : HOME : VTAB 1Ø: PRINT "Hoffe, es hat Spaß
     gemacht !"
214Ø VTAB 2Ø: HTAB 36: PRINT "HME";
215Ø END : GOTO 179Ø: REM für evtl. CONT
298Ø REM Anfang Subroutinen
299Ø REM Funktionseingabe
3ØØØ I = Ø: HOME : PRINT "Bitte geben Sie die
     darzustellende": PRINT "Funktion ein !"
3Ø1Ø VTAB 1Ø: PRINT F$: VTAB 1Ø: GOSUB 54ØØ
3Ø2Ø VTAB 1Ø: CALL CL%: PRINT F$;: CALL CL%: PRINT : VTAB
     22: PRINT "Ist das richtig ? ";: CALL CL%
3Ø3Ø GOSUB 47ØØ: IF A$ = "N" THEN PRINT "Bitte
     korrigieren !": GOTO 3Ø1Ø
3Ø4Ø RETURN
3Ø5Ø REM Darstellungsgrenzen
3Ø6Ø IF NOT (XL = Ø AND XH = Ø AND YL = Ø AND YH = Ø)
     THEN HOME : GOTO 326Ø: REM alte Werte zeigen
3Ø7Ø HOME : PRINT "Bitte Grenzen eingeben! (Anweisungen?)"
3Ø8Ø GOSUB 47ØØ: IF A$ = "N" THEN VTAB 17: GOTO 315Ø
3Ø9Ø PRINT : PRINT "Soll der Maßstab der x-Achse dem
     der": PRINT "y-Achse angeglichen sein, geben Sie
     die": PRINT "gewünschten Darstellungsgrenzen für":
     PRINT "e i n e  Achse ein, für die andere": PRINT
     "einfach  Ø,Ø ."
31ØØ PRINT "Werden kleinster und größter Wert in": PRINT
     "falscher Reihenfolge eingegeben, werden": PRINT
     "sie vom Programm vertauscht."
311Ø PRINT "X-Min, Y-Min sollten immer kleiner oder":
     PRINT "gleich Ø , X-Max , Y-Max immer größer": PRINT
     "oder gleich Ø gewählt werden (sonst": PRINT "können
     keine Achsen gezeichnet werden)"
312Ø PRINT "Das Programm setzt daher falsch gesetzte":
     PRINT "Werte auf Ø !"
313Ø PRINT "Sollten Sie nicht ganz mit dem Darstel-":
     PRINT "lungsmaßstab einverstanden sein, können":
     PRINT "Sie das Verhältnis von negativem zu po-":
     PRINT "sitivem Achsenabschnitt verändern oder":
     PRINT "die Achsen strecken oder stauchen."
314Ø GET A$: HOME : PRINT "Bitte Grenzen eingeben !"
315Ø VTAB 7: PRINT : INPUT "X-MIN , X-MAX : ";XL,XH
316Ø VTAB 1Ø: INPUT "Y-MIN , Y-MAX : ";YL,YH
317Ø IF XL > XH THEN X = XL:XL = XH:XH = X
318Ø IF YL > YH THEN X = YL:YL = YH:YH = X
319Ø IF YL > Ø THEN YL = Ø
32ØØ IF YH < Ø THEN YH = Ø
321Ø IF XL > Ø THEN XL = Ø
322Ø IF XH < Ø THEN XH = Ø
323Ø IF YL = YH AND XL = XH THEN PRINT CHR$ (7);
     "Eingabefehler ! Bitte wiederholen !": GOTO 315Ø
324Ø IF YL = YH THEN GOSUB 357Ø
325Ø IF XL = XH THEN GOSUB 355Ø
326Ø VTAB 13: PRINT "Die Darstellungsgrenzen wurden wie
     folgt festgelegt"
327Ø PRINT : PRINT "Kleinstes X  : ";XL;: CALL CL%: PRINT
328Ø PRINT "Größtes X    : ";XH;: CALL CL%: PRINT
329Ø PRINT : PRINT "Kleinstes Y  : ";YL;: CALL CL%: PRINT
33ØØ PRINT "Größtes Y    : ";YH;: CALL CL%: PRINT
331Ø PRINT : VTAB 22: PRINT "Ist das richtig ? ";: CALL
     CL%
332Ø GOSUB 47ØØ: IF A$ = "J" THEN PRINT : GOTO 35ØØ
333Ø I = I:XØ = I:YØ = I:Xl = I:Yl = I:MX = I:MY = I:BX =
     I:BY = I: REM Clear Hilfsvariablen
334Ø PRINT "Schwere Fehler ? ";: GOSUB 47ØØ: IF A$ = "J"
     THEN PRINT : VTAB 1: PRINT "Dann bitte wiederholen
     !": GOTO 315Ø
335Ø HOME : PRINT "Kleine Änderungen :"
336Ø PRINT : PRINT " x-Achse strecken ? ";: GOSUB 47ØØ:
     IF A$ = "J" THEN INPUT "Verhältnis ";XØ,Xl
337Ø PRINT : VTAB 5: PRINT " y-Achse strecken ? ";: GOSUB
     47ØØ: IF A$ = "J" THEN INPUT "Verhältnis ";YØ,Yl
338Ø PRINT : VTAB 8: PRINT "Nun bitte das Verhältnis + /
     - Achse!": PRINT : PRINT " Für x-Achse ? ";: GOSUB
     47ØØ: IF A$ = "J" THEN INPUT " + , - ";MX,BX
339Ø PRINT : VTAB 12: PRINT " Für y-Achse ? ";: GOSUB
     47ØØ: IF A$ = "J" THEN INPUT " + / - ";MY,BY
34ØØ IF (XØ = Ø AND Xl < > Ø) OR (Xl = Ø AND XØ < > Ø) OR
     (YØ = Ø AND Yl < > Ø) OR (Yl = Ø AND YØ < > Ø) THEN
     PRINT : PRINT "Nie mit Ø strecken oder stauchen !":
     GOTO 336Ø
341Ø VTAB 2Ø: PRINT "Alles o.k.? ";: GOSUB 47ØØ: IF A$ =
     "N" THEN PRINT : VTAB 2: PRINT "Bitte wiederholen
     !";: CALL CL%: GOTO 336Ø
342Ø IF XØ < > Ø AND Xl < > Ø THEN XL = XL / XØ * Xl:XH =
     XH / XØ * Xl
343Ø IF YØ < > Ø AND Yl < > Ø THEN YL = YL / YØ * Yl:YH =
     YH / YØ * Yl
344Ø IF MX < > Ø OR BX < > Ø THEN XØ = XH - XL:Xl = MX +
     BX:XH = MX / Xl * XØ:XL = - BX / Xl * XØ
345Ø IF MY < > Ø OR BY < > Ø THEN YØ = YH - YL:Yl = MY +
     BY:YH = MY / Yl * YØ:YL = - BY / Yl * YØ
346Ø HOME : GOTO 326Ø
347Ø :
349Ø REM Umrechnung Bildschirm
35ØØ MX = 279 / (XH - XL):BX = - MX * XL
351Ø MY = - 191 / (YH - YL):BY = - MY * YH
352Ø DEF FN XS(X) = INT (MX * X + BX + .5): DEF FN YS(X)
     = INT (MY * X + BY + .5)
353Ø RETURN
354Ø REM x-Achse an y-Achse anpassen
355Ø FA = 279 / 191:XL = FA * (YL - YH) / 2:XH = FA * (YH
     - YL) / 2: RETURN
356Ø REM y-Achse an x-Achse anpassen
357Ø FA = 279 / 191:YL = (XL - XH) / (2 * FA):YH = (XH -
     XL) / (2 * FA): RETURN
3590 REM für Skalierung
36ØØ XØ = INT ( FN LG(X)): IF XØ < FN LG(X) THEN XØ = XØ
     + 1
361Ø ST = XØ - 1:XØ = EXP ( LOG (1Ø) * XØ):ST = EXP ( LOG
     (1Ø) * ST)
362Ø RETURN
369Ø REM Parameter ?
37ØØ HOME : PRINT "Wurden Parameter verwendet ? ";: GOSUB
     47ØØ: IF A$ = "N" THEN P$ = "Ø": RETURN
371Ø PRINT : PRINT "Bitte die betreffende Taste für die
     ge-": PRINT "wünschte Kombination von Parametern":
     PRINT "betätigen !"
372Ø VTAB 7: HTAB 1Ø: PRINT "1 - nur R"
373Ø VTAB 9: HTAB 1Ø: PRINT "2 - nur S"
374Ø VTAB 11: HTAB 1Ø: PRINT "3 - nur T"
375Ø VTAB 13: HTAB 1Ø: PRINT "4 - R und S"
```

```
376Ø VTAB 15: HTAB 1Ø: PRINT "5 - R und T"
377Ø VTAB 17: HTAB 1Ø: PRINT "6 - S und T"
378Ø VTAB 19: HTAB 1Ø: PRINT "7 - R,S und T"
379Ø VTAB 22: HTAB 1Ø: PRINT "Ø - keine Parameter"
38ØØ GET P$: IF ( VAL (P$) = Ø AND P$ < > "Ø") OR VAL
     (P$) > 7 THEN 38ØØ
381Ø HOME : IF VAL (P$) < 3 OR VAL (P$) = 4 THEN 384Ø
382Ø PRINT "Es wurde der Parameter T verwendet. ": PRINT
     "Soll T mit einer festen Schrittweite": PRINT
     "durchlaufen werden ?";
383Ø GOSUB 47ØØ: IF A$ = "J" THEN P$ = P$ + "K"
384Ø RETURN
385Ø :
388Ø REM Eingabe Parameter
389Ø REM Eingabe von T
39ØØ IF RIGHT$ (P$,1) = "K" THEN 397Ø
391Ø PRINT : VTAB 4: INPUT "Bitte T eingeben : ";T
392Ø VTAB 4: PRINT SPC( 14);"T = ";T: CALL CL%
393Ø PRINT : PRINT TAB( 2Ø);"Richtig ?";
394Ø GOSUB 47ØØ: IF A$ = "N" THEN
395Ø RETURN
396Ø REM T in Schleife
397Ø PRINT : VTAB 2: INPUT "Bitte Startwert und Endwert
     von T eingeben : ";TØ,T1
398Ø VTAB 5: INPUT "Nun die Schrittweite :";TS
399Ø IF TS = Ø THEN VTAB 7: PRINT CHR$ (7);"Nie
     Schrittweite Ø !": GOTO 398Ø
4ØØØ IF TS < Ø THEN 4Ø3Ø
4Ø1Ø IF T1 < TØ THEN T = T1:T1 = TØ:TØ = T
4Ø2Ø GOTO 4Ø4Ø
4Ø3Ø IF TØ < T1 THEN T = TØ:TØ = T1:T1 = T
4Ø4Ø L1 = 1:L2 = 7: GOSUB 475Ø: VTAB 2
4Ø5Ø PRINT "T          läuft von : ";TØ
4Ø6Ø PRINT "                bis : ";T1
4Ø7Ø PRINT "   mit Schrittweite : ";TS
4Ø8Ø PRINT : PRINT TAB( 1Ø);"Ist das richtig ? ";: GOSUB
     47ØØ: IF A$ = "N" THEN L1 = 2:L2 = 6: GOSUB 475Ø:
     GOTO 397Ø
4Ø9Ø RETURN
41ØØ :
411Ø REM Eingabe von S
412Ø PRINT : VTAB 9: INPUT "Bitte S eingeben : ";S
413Ø VTAB 9: PRINT SPC( 15);"S = ";S: CALL CL%
414Ø VTAB 11: PRINT TAB( 2Ø);"Richtig ?";: GOSUB 47ØØ: IF
     A$ = "N" THEN 412Ø
415Ø RETURN
416Ø :
417Ø REM Eingabe von R
418Ø PRINT : VTAB 15: INPUT "Bitte R eingeben : ";R
419Ø VTAB 15: PRINT SPC( 15);"R = ";R: CALL CL%
42ØØ VTAB 17: PRINT TAB( 2Ø);"Richtig ?";: GOSUB 47ØØ: IF
     A$ = "N" THEN 418Ø
421Ø RETURN
422Ø :
424Ø REM für Parametertabelle
425Ø N$ = "nicht verwendet"
426Ø ON VAL (P$) GOTO 428Ø,429Ø,43ØØ,431Ø,432Ø,433Ø,434Ø
427Ø S$ = N$:T$ = N$:R$ = N$: RETURN
428Ø R$ = STR$ (R):S$ = N$:T$ = N$: RETURN
429Ø S$ = STR$ (S):R$ = N$:T$ = N$: RETURN
43ØØ T$ = STR$ (T):R$ = N$:S$ = N$: RETURN
431Ø R$ = STR$ (R):S$ = STR$ (S):T$ = N$: RETURN
432Ø R$ = STR$ (R):T$ = STR$ (T):S$ = N$: RETURN
433Ø S$ = STR$ (S):T$ = STR$ (T):R$ = N$: RETURN
434Ø R$ = STR$ (R):S$ = STR$ (S):T$ = STR$ (T): RETURN
439Ø REM alle Parameter eingeben
44ØØ IF P$ = "Ø" THEN 443Ø
441Ø HOME : PRINT TAB( 1Ø);"Parametereingabe"
442Ø ON VAL (P$) GOSUB 444Ø,445Ø,446Ø,447Ø,448Ø,449Ø,45ØØ
443Ø HOME : RETURN
444Ø GOSUB 418Ø: RETURN
445Ø GOSUB 412Ø: RETURN
446Ø GOSUB 39ØØ: RETURN
447Ø GOSUB 412Ø: GOSUB 418Ø: RETURN
448Ø GOSUB 39ØØ: GOSUB 418Ø: RETURN
449Ø GOSUB 39ØØ: GOSUB 412Ø: RETURN
45ØØ GOSUB 39ØØ: GOSUB 412Ø: GOSUB 418Ø: RETURN
459Ø REM Kurzzusammenfassung
46ØØ HOME : PRINT "Die Funktion lautet :": PRINT : PRINT
     F$
461Ø VTAB 13: PRINT "Darstellungsgrenzen :": PRINT TAB(
     1Ø);"Kleinstes X : ";XL: PRINT TAB( 1Ø);"Größtes X
     : ";XH
462Ø PRINT TAB( 1Ø);"Kleinstes Y : ";YL: PRINT TAB(
     1Ø);"Größtes Y   : ";YH
463Ø VTAB 19: PRINT "Parameter :": GOSUB 425Ø
464Ø PRINT TAB( 1Ø);"R = ";R$: PRINT TAB( 1Ø);"T = ";T$:
     PRINT TAB( 1Ø);"S = ";S$
465Ø PRINT "Ordinate X = ";X
466Ø RETURN
467Ø :
469Ø REM Teste ob ja oder nein
47ØØ GET A$: IF A$ < > "J" AND A$ < > "N" THEN 47ØØ
471Ø RETURN
472Ø :
474Ø REM lösche zwischen L1 und L2 Bildschirm
475Ø FOR L = L1 TO L2: VTAB L: POKE CH%,Ø: CALL CL%: NEXT
     : RETURN
476Ø :
479Ø REM Grafik/Text umschalten
48ØØ POKE - 16368,Ø: REM Keyboardstrobe löschen
481Ø IF PEEK (253) = 1 THEN POKE - 163Ø3,Ø: POKE 253,2:
     RETURN : REM Textseite
482Ø POKE - 163Ø4,Ø: POKE - 16297,Ø: POKE 253,1: RETURN :
     REM Grafikseite
485Ø :
488Ø REM Zeichne Achsen
489Ø REM und skaliere
49ØØ HPLOT FN XS(Ø),Ø TO FN XS(Ø),191: HPLOT Ø, FN YS(Ø)
     TO 279, FN YS(Ø): ROT= Ø:K = Ø: IF YL = Ø THEN
     K = - 2
491Ø IF YH = Ø THEN K = 2
492Ø X = XH: IF ABS (XL) > XH THEN X = ABS (XL)
493Ø GOSUB 36ØØ: FOR X = - XØ TO XØ STEP ST
494Ø IF X > XL AND X < XH THEN DRAW 1 AT FN XS(X), FN
     YS(Ø) + K
495Ø NEXT :K = Ø: IF XL = Ø THEN K = 2
496Ø IF XH = Ø THEN K = - 2
497Ø X = YH: IF ABS (YL) > X THEN X = ABS (YL): GOSUB
     36ØØ
498Ø GOSUB 36ØØ: ROT= 16: FOR X = - XØ TO XØ STEP ST
499Ø IF X > YL AND X < YH THEN DRAW 1 AT FN XS(Ø) + K, FN
     YS(X)
5ØØØ NEXT : RETURN
5Ø1Ø :
5Ø9Ø REM Ausdruck
51ØØ PRINT : PRINT D$;"PR#1": PRINT HT$;DP$: PRINT
     E$;"M";E$;"1"; CHR$ (6);
511Ø REM EPSON FX-8Ø Steuercodes
512Ø PRINT E$;"Q"; CHR$ (9Ø);E$;"A"; CHR$ (15); TAB(
     43);E$;"-1";" PLOT.2.Ø ";E$;"-Ø"
513Ø PRINT E$;"N"; CHR$ (6); CHR$ (13); CHR$ (1Ø); CHR$
     (1Ø);"Die Funktion lautet :"
514Ø REM rechter, linker Rand, Schönschrift,
     Formularlänge
515Ø PRINT " F(X) = "; MID$ (F$,3): PRINT
516Ø PRINT E$;"D"; CHR$ (8); CHR$ (24); CHR$ (4Ø); CHR$
     (56); CHR$ (72); CHR$ (Ø)
517Ø PRINT CHR$ (1Ø); CHR$ (13);"Darstellungsgrenzen :
     ";HT$;"x läuft von :";HT$;XL;HT$;"bis : ";XH
518Ø PRINT HT$;HT$;"y läuft von :";HT$;YL;HT$;"bis : ";YH
519Ø PRINT CHR$ (13);"Parameterliste:";: IF R$ = N$ AND
     T$ = N$ AND S$ = N$ THEN PRINT " Es wurden keine
     Scharparameter verwendet !": PRINT : GOTO 527Ø
52ØØ PRINT " Folgende Parameter wurden mit den auf-":
     PRINT "geführten Werten durchlaufen."; CHR$ (1Ø)
521Ø IF R$ = N$ THEN 523Ø
522Ø PRINT "  R  :";:K = Ø: GOSUB 531Ø: PRINT
523Ø IF S$ = N$ THEN 525Ø
524Ø PRINT "  S  :";:K = 1: GOSUB 531Ø: PRINT
525Ø IF T$ = N$ THEN 527Ø
526Ø PRINT "  T  :";:K = 2: GOSUB 531Ø: PRINT
527Ø PRINT CHR$ (13); CHR$ (1Ø);"Die Grafik :": PRINT
528Ø  PRINT DP$;"G": PRINT CHR$ (1Ø); CHR$ (1Ø); TAB(
     82);"HME"; CHR$ (13); CHR$ (12): PRINT D$;"PR# Ø":
     RETURN
529Ø REM
53ØØ REM Parametertabelle
531Ø FOR J = 1 TO I: PRINT HT$;P(K,J);: IF J / 5 = INT (J
     / 5) THEN PRINT
532Ø NEXT J: PRINT : RETURN
538Ø REM
539Ø REM Rufe Maschinenprogramm für Funktionseingabe auf
54ØØ CALL AS%: GOSUB 1Ø1Ø:F$ = "Y= "
541Ø FOR I = AP% TO AP% + PEEK (2Ø6):F$ = F$ + CHR$
     (PEEK (I)): NEXT I: REM Hole Funktion aus
     Eingabepuffer
542Ø I = Ø: RETURN
543Ø REM Ende Eingabe
544Ø REM
549Ø REM Fehlerbehandlung
55ØØ LN = PEEK (218) + 256 * PEEK (219):ER = PEEK (222)
551Ø CALL 348Ø
```

```
5520 IF ER = 255 THEN TEXT : HOME : GOTO 1800: REM
     Hauptmenü
5530 IF ER = 254 THEN PRINT "?REENTER": RESUME : REM
     Anwenderfehler bei INPUT
5540 IF ER = 133 OR ER = 69 OR ER = 53 THEN XA = ST +
     X:X0 = 0: GOTO 1650
5545 REM Illegal Quantity, Overflow, Division by Zero
5550 TEXT : VTAB 20: PRINT CHR$ (13); CHR$ (7);" Fehler
     Code # ";ER;" in Zeile ";LN: END
5560 :
5580 REM Pointer in ($67) falsch gesetzt, Programm liegt
     in Grafikseite
5590 REM EXEC-File korrigiert das, lädt und startet das
     Programm von neuem
5600 HOME : VTAB 10: HTAB 5: PRINT "Programm PLOT.2.0
     wird geladen": PRINT : HTAB 14: PRINT "Bitte warten"
5610 PRINT CHR$ (4);"EXEC PLOT.PROTECTOR"
```

PLOT. PROTECTOR

```
POKE 50,128
POKE 103,1
POKE 104,64
POKE 16384,0
LOAD PLOT.2.0
CALL 54514
POKE 50,225
RUN
```

PLOT.B

```
                        1   ************************
                        2   *  Hilfsroutinen für  *
                        3   *       PLOT 2.0       *
                        4   *   Hans-Martin Eng    *
                        5   *    November 1984     *
                        6   ************************
                        7               ORG  $D000
                        8   *
                        10  PNTL      EQU  $50
                        11  PNTH      EQU  $51
                        12  STRPNTL   EQU  $CE
                        13  STRPNTH   EQU  $CF
                        14  LCHARL    EQU  $B8
                        15  LCHARH    EQU  $B9
                        16  INPNTL    EQU  $7F
                        17  INPNTH    EQU  $80
                        18  PRGANFL   EQU  $67
                        19  PRGANFH   EQU  $68
                        20  LPNT      EQU  $D6
                        21  YSAV      EQU  $D7
                        22  INBUF     EQU  $0200
                        23  INTXT     EQU  $DB5C
                        24  INPUT     EQU  $D52E
                        25  PCODE     EQU  $D56C
                        26  IOSAVE    EQU  $FF4A
                        27  IOREST    EQU  $FF3F
                        28  *
                        29  ************************
                        30  *  Funktionseingabe   *
                        31  *   Zeilen 46 bis 95   *
                        32  *  von Bernd Heyer    *
                        33  *       (1980)        *
                        34  *  in CHIP SPECIAL #2 *
                        36  *   disassembliert,   *
                        37  *   kommentiert und   *
                        38  * von Fehlern befreit *
                        39  * von Hans-Martin Eng *
                        40  *    November 1984    *
                        41  ************************
                        42  *
                        43  * Register retten und Pointer
                        44  * suchen
0D00: 20 4A FF          45  LOS       JSR  IOSAVE
0D03: A6 B8             46            LDX  LCHARL
0D05: A4 B9             47            LDY  LCHARH
0D07: 86 7F             48            STX  INPNTL
0D09: 84 80             49            STY  INPNTH
                        50  * "Y" als "Prompt-Character" an
                        51  * Input-Routine übergeben
0D0B: A9 59             52            LDA  #$59
0D0D: 20 5C DB          53            JSR  INTXT
0D10: A2 BD             54            LDX  #$BD
                        55  * Funktionsvorschrift einlesen
0D12: 20 2E D5          56            JSR  INPUT
                        57  * umwandeln in Tokens
0D15: 20 6C D5          58            JSR  PCODE
0D18: A6 7F             59            LDX  INPNTL
```

```
ØD1A: A4 8Ø      6Ø            LDY  INPNTH
ØD1C: 86 B8      61            STX  LCHARL
ØD1E: 84 B9      62            STY  LCHARH
                 63   * suche Startadresse des
                 64   * Applesoftprogramms.
                 65   *  20 Bytes später steht
                 66   *  das "=" in Zeile 2
ØD2Ø: A5 68      67            LDA  PRGANFH
ØD22: 85 51      68            STA  PNTH
ØD24: 18         69            CLC
ØD25: A5 67      7Ø            LDA  PRGANFL
ØD27: 69 15      71            ADC  #$15
ØD29: 85 5Ø      72            STA  PNTL
ØD2B: 9Ø Ø2      73            BCC  LASSEN
ØD2D: E6 51      74            INC  PNTH
                 75   * Zeile 2 wird mit 228
                 76   * Doppelpunkten aufgefüllt
ØD2F: AØ E3      77   LASSEN   LDY  #$E3
ØD31: A9 3A      78            LDA  #$3A       ;':'
ØD33: 91 5Ø      79   SCHLEIFE STA  (PNTL),Y
ØD35: 88         8Ø            DEY
ØD36: DØ FB      81            BNE  SCHLEIFE
                 82   * Nun wird der Inhalt des
                 83   * Eingabepuffers hinter das
                 84   * "=" in Zeile 2 geschrieben
ØD38: B9 FD Ø1   85   TRANSF   LDA  INBUF-3,Y
ØD3B: 91 5Ø      86            STA  (PNTL),Y
ØD3D: C8         87            INY
                 88   * TAX setzt die Prozessorflags.
                 89   * Abbruch bei Zeilenende
ØD3E: AA         9Ø            TAX
                 91   * $ØØ durch ":" ersetzen
ØD3F: DØ F7      92            BNE  TRANSF
ØD41: A9 3A      93            LDA  #$3A
ØD43: 88         94            DEY
ØD44: 91 5Ø      95            STA  (PNTL),Y
                 96   * In einen Puffer ab $ØC ØØ wird
                 97   * ein String geschrieben, der
                 98   * die Funktionsvorschrift enthält.

ØD46: A9 FF      1ØØ  STRING   LDA  #$FF
ØD48: A2 ØC      1Ø1           LDX  #$ØC
ØD4A: 85 CE      1Ø2           STA  STRPNTL
ØD4C: 86 CF      1Ø3           STX  STRPNTH
ØD4E: 84 D6      1Ø4           STY  LPNT
ØD5Ø: AØ ØØ      1Ø5           LDY  #$ØØ
                 1Ø6   * Hole Zeichen/Token aus der
                 1Ø7   * Programmzeile
ØD52: B1 5Ø      1Ø8   NEXT     LDA  (PNTL),Y
ØD54: 38         1Ø9           SEC
                 11Ø   * Ist es ein Token ?
ØD55: C9 8Ø      111           CMP  #$8Ø
                 112   * wenn ja, suche zugehörige
                 113   * Zeichenkette in Tabelle
ØD57: BØ ØF      114           BCS  TOKEN
                 115   * nein, schreibe Zeichen in Puffer
ØD59: E6 CE      116           INC  STRPNTL
ØD5B: A2 ØØ      117           LDX  #$ØØ
ØD5D: 81 CE      118           STA  (STRPNTL,X)
ØD5F: C8         119  ENDE?    INY
ØD6Ø: C4 D6      12Ø           CPY  LPNT
ØD62: DØ EE      121           BNE  NEXT
ØD64: 2Ø 3F FF   122           JSR  IOREST
ØD67: 6Ø         123           RTS
                 124   * Alle Register werden gebraucht
ØD68: 84 D7      125  TOKEN     STY  YSAV
ØD6A: AØ 13      126           LDY  #$13
ØD6C: 88         127  NXTT     DEY
ØD6D: 3Ø 1E      128           BMI  ERR
                 129   * Vergleich mit Token-Tabelle
ØD6F: D9 EB ØD   13Ø           CMP  TBL2,Y
ØD72: DØ F8      131           BNE  NXTT
                 132   * Pointer auf Beginn des Strings
                 133   * aus Tabelle entnehmen
ØD74: BE FD ØD   134           LDX  TBL3,Y
ØD77: AØ ØØ      135           LDY  #$Ø
ØD79: BD AD ØD   136           LDA  TBL1,X
ØD7C: E6 CE      137  NXTCHAR  INC  STRPNTL
ØD7E: 91 CE      138           STA  (STRPNTL),Y
ØD8Ø: E8         139           INX
ØD81: BD AD ØD   14Ø           LDA  TBL1,X
                 141   * Wenn jetzt ein Space im Accu
                 142   * steht, ist der String komplett
ØD84: C9 2Ø      143           CMP  #$2Ø
ØD86: DØ F4      144           BNE  NXTCHAR
ØD88: A4 D7      145           LDY  YSAV
ØD8A: B8         146           CLV
ØD8B: 5Ø D2      147           BVC  ENDE?
```

```
                 148   * Es trat ein Token auf, das nicht
                 149   * in der Token-Tabelle berück-
                 15Ø   * sichtigt ist. Wird darauf Wert
                 151   * gelegt, sollte man es hinzu-
                 152   * fügen.
ØD8D: E6 CE      153   ERR      INC  STRPNTL
ØD8F: A2 ØØ      154            LDX  #$Ø
ØD91: A9 3F      155            LDA  #$3F       ;'?'
ØD93: 81 CE      156            STA  (STRPNTL,X)
ØD95: B8         157            CLV
ØD96: 5Ø C7      158            BVC  ENDE?
                 161   * APPLESOFT ONERR SUBROUTINE
                 162   * Aus Applesoft Programming
                 163   * Reference Manual, S.82
ØD98: 68         164            PLA
ØD99: A8         165            TAY
ØD9A: 68         166            PLA
ØD9B: A6 DF      167            LDX  $DF
ØD9D: 9A         168            TXS
ØD9E: 48         169            PHA
ØD9F: 98         17Ø            TYA
ØDAØ: 48         171            PHA
ØDA1: 6Ø         172            RTS
                 174   * Ab hier Tabellen
                 176   * Ein Shape zur Skalierung
                 177   * sechs Vektoren
ØDA2: Ø1 ØØ Ø4   178   SHTABLE  HEX  Ø1ØØØ4ØØ
ØDA5: ØØ
ØDA6: Ø2 Ø2 Ø4   179            HEX  Ø2Ø2Ø4Ø4Ø4ØØ
ØDA9: Ø4 Ø4 Ø4 ØØ
                 18Ø   * 2 nach unten ohne Plot
                 181   * 4 nach oben mit Plot
                 185   * Reservierte Wörter
ØDAD: 2Ø 46 4E   186   TBL1     ASC  ' FN'
ØDBØ: 2Ø 2B      187            ASC  ' +'
ØDB2: 2Ø 2D      188            ASC  ' -'
ØDB4: 2Ø 2A      189            ASC  ' *'
ØDB6: 2Ø 2F      19Ø            ASC  ' /'
ØDB8: 2Ø 5E      191            ASC  ' ↑'
ØDBA: 2Ø 53 47   192            ASC  ' SGN'
ØDBD: 4E
ØDBE: 2Ø 49 4E   193            ASC  ' INT'
ØDC1: 54
ØDC2: 2Ø 41 42   194            ASC  ' ABS'
ØDC5: 53
ØDC6: 2Ø 55 53   195            ASC  ' USR'
ØDC9: 52
ØDCA: 2Ø 53 51   196            ASC  ' SQR'
ØDCD: 52
ØDCE: 2Ø 52 4E   197            ASC  ' RND'
ØDD1: 44
ØDD2: 2Ø 4C 4F   198            ASC  ' LOG'
ØDD5: 47
ØDD6: 2Ø 45 58   199            ASC  ' EXP'
ØDD9: 5Ø
ØDDA: 2Ø 43 4F   2ØØ            ASC  ' COS'
ØDDD: 53
ØDDE: 2Ø 53 49   2Ø1            ASC  ' SIN'
ØDE1: 4E
ØDE2: 2Ø 54 41   2Ø2            ASC  ' TAN'
ØDE5: 4E
ØDE6: 2Ø 41 54   2Ø3            ASC  ' ATN '
ØDE9: 4E 2Ø
                 2Ø5   * Token-Tabelle
ØDEB: C2 C8 C9   2Ø6   TBL2     HEX  C2C8C9CA
ØDEE: CA
ØDEF: CB CC D2   2Ø7            HEX  CBCCD2D3
ØDF2: D3
ØDF3: D4 D5 DA   2Ø8            HEX  D4D5DADB
ØDF6: DB
ØDF7: DC DD DE   2Ø9            HEX  DCDDDEDF
ØDFA: DF
ØDFB: EØ E1      21Ø            HEX  EØE1
                 212   * Pointer zum Startpunkt in der
                 213   * Schlüsselworttabelle
ØDFD: ØØ Ø3 Ø5   214   TBL3     HEX  ØØØ3Ø5Ø7Ø9
ØEØØ: Ø7 Ø9
ØE Ø2: ØB ØD 11  215            HEX  ØBØD111519
ØE Ø5: 15 19
ØE Ø7: 1D 21 25  216            HEX  1D212529
ØE ØA: 29
ØE ØB: 2D 31 35  217            HEX  2D313539
ØE ØE: 39

271 Bytes
```

CONVERT560

von Marc van Woerkom

Konvertierungsroutine für Double-Hires-Drucker-Dump

Die neuen Computer der Apple-II-Familie, d.h. der Apple IIc und der Apple IIe mit erweiterter 80-Zeichenkarte (= 64K-Karte), bieten die Möglichkeit, Graphiken mit einer Auflösung von 560 x 192 Punkten zu erzeugen (= sog. Double-Hires-Graphik.) Im Peeker 2/84 wurde schon über die Organisation der Double-Hires-Graphik berichtet. Apple hat aber leider diese neuen Graphikfähigkeiten weder im Applesoft-Interpreter noch in der Treibersoftware der erweiterten 80-Zeichenkarte implementiert, so daß Double-Hires-Graphik nur mit externer Software möglich ist. Derartige Software ist inzwischen schon vorhanden (s. Peeker 3/85: Test der Graphikpakete „Doublestuff" und „Superplot").

Double-Hires-Graphik ist sehr praktisch zum Plotten von mathematischen Funktionen, wie **Bild 1** und **Bild 2** eindrucksvoll zeigen, doch bieten die bisherigen Softwarepakete keine Möglichkeit des Ausdrucks dieser Double-Hires-Graphik, was natürlich sehr unbefriedigend ist, da man seine Ergebnisse ja gerne auch schwarz auf weiß zu Demonstrationszwecken usw. haben möchte. Dadurch sah ich mich veranlaßt, hier Abhilfe zu schaffen, und zwar mit möglichst geringem Aufwand.

Die einfachste Implementierung dürfte in der Ausnutzung der bereits existierenden Programme zum Ausdruck normaler hochauflösender Graphik in Verbindung mit der hier veröffentlichten Utility **CONVERT560** bestehen. Die Begründung dafür ist, daß quasi jeder graphikfähige Drucker anders angesteuert werden muß!
Da der Aufwand zur Erstellung von neuer „Graphik-Dump-Software" zu *jedem* Drucker-Modell enorm wäre, kann man die alte, bereits vorhandene Software zusammen mit CONVERT560 benutzen, das den Vorteil bietet, mit jedem Drucker zu arbeiten. Voraussetzung ist allerdings, daß die vorhandene Dump-Software in der Lage ist, zwei Hires-Bilder *nebeneinander* auszudrucken (2 Hires-Dumps = 1 Double-Hires-Dump).

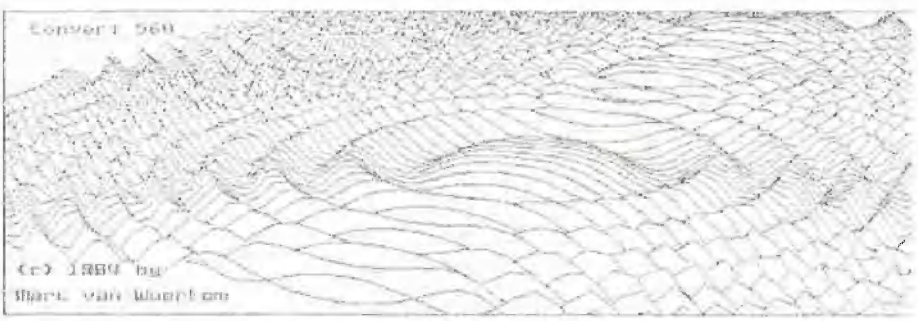


Bild 1: Normales DHGR-Bild

Wie CONVERT560 funktioniert

CONVERT560 zerlegt einfach ein Double-Hires-Bild mit einer Auflösung von 560 x 192 Punkten in zwei „normale" Bilder zu je 280 x 192 Punkten. Diese werden dann wie gewohnt ausgedruckt. (Nicht jedes Interface-Programm ist jedoch dazu in der Lage. Beispielsweise kann die zum Imagewriter mitgelieferte Toolkit-Diskette nicht zwei Bilder nebeneinander ausdrucken. Anm. d. Red.) Vor dem Aufruf von CONVERT560 befindet sich das Double-Hires-Bild in den zwei Speicherbereichen $2000–$3FFF (a) von den „unteren" 64K und (b) von den „oberen" 64K. Nach dem Aufruf von CONVERT560 liegt das Double-Hires-Bild (a) in dem „unteren" Bereich $2000–$3FFF (linke Bildhälfte) und (b) in dem „unteren" Bereich $4000–$5FFF (rechte Bildhälfte).

Um sicherzugehen, daß CONVERT560 mit jeder Art von Graphik-Dump-Software zusammenarbeitet, wurden folgende Gesichtspunkte bei der Erstellung von CONVERT560 beachtet:

a CONVERT560 sollte relokativ*, sprich überall im Speicher lauffähig sein, um an die vorhandene Graphik-Dump-Software angehängt werden zu können.
b Es sollte möglichst wenig Speicherplatz belegen, zum einen wegen (a) und zum anderen, um es eventuell in einem EPROM unterbringen zu können.
c Die Bereiche zur Abspeicherung der durch CONVERT560 erhaltenen Bildhälften sollten frei wählbar sein, denn diese

* Die Register-Save-Routine ist nicht relokativ (Anm. d. Red.)

werden im Normalfall im HGR1- und HGR2-Bereich abgelegt, wodurch das Double-Hires-Bild zerstört wird, da es auch den HGR1-Bereich belegt. Dies ist vielleicht nicht immer erwünscht.

Diese drei Gesichtspunkte wurden wie folgt realisiert:

- Sollte bei Ihnen die Startadresse $0300 nicht nicht möglich sein, so ändern Sie bitte die „ORG"-Anweisung im Source-Listing.
- CONVERT560 belegt weniger als eine Seite im Speicher, genau $00A6 (dezimal 166) Bytes. Um dies zu erreichen, wurden folgende Maßnahmen durchgeführt:

Der von CONVERT560 zusätzlich benötigte, 80 Bytes lange Zwischenspeicher wurde in das Ende des Tastaturspeichers ($0280- $02FF) gelegt.
CONVERT560 muß die Anfangsadressen der 192 Graphik-Zeilen berechnen können. Aus Platzgründen wurde jedoch keine schnelle „Lookup-Table" verwendet, sondern die 10mal kürzere und damit langsamere, modifizierte Version von HPOSN ($F411).

- Wünschen Sie andere Speicherbereiche für die entstandene linke und rechte Bildhälfte, so ändern sie im Source-Listing die Definitionen der Label „LeftPic" und „RightPic".

Die Funktionsweise von CONVERT560 ist relativ einfach: Es wird jede der 192 Zeilen der Double-Hires-Graphik einzeln erst in den Zwischenspeicher geladen, dort richtig umgestellt und dann in die angegebenen Speicherbereiche geschoben.

Neben dem Source-Listing von CONVERT560 ist noch ein kurzes Beispiel-Programm **CONVERT560.DEMO** abgedruckt, das den Gebrauch von CONVERT560 verdeutlichen soll. Mit ihm wurden auch die Bilder 1 und 2 gedruckt, wobei ein Epson FX-80 mit Epson Interface 8132W zur Verfügung stand. Dieses Interface hat den Vorteil, daß die HGR1- und HGR2-Seiten direkt nebeneinander ausgedruckt werden können.
Dabei ist das Gesamtbild so komprimiert, daß es genauso groß wie ein normales Bild ist, so daß die Proportionen mit denen der 80-Zeichenkarte übereinstimmen.

CONVERT560.DEMO

```
100 REM CONVERT560.DEMO
110 REM für Epson FX-80/8132W
120 REM Das DHGR-Bild ist
130 REM bereits geladen.
140 TEXT : HOME : NORMAL
150 D$ = CHR$ (4): PRINT :
    PRINT D$;"BLOAD CONVERT560"
160 PRINT : CALL 768: REM Konvertieren
170 I$ = CHR$ (9): PRINT :
    PRINT CHR$ (4);"PR#1"
180 PRINT : PRINT I$;"GBD"
190 PRINT : PRINT D$;"PR#0"
200 PRINT: END
```

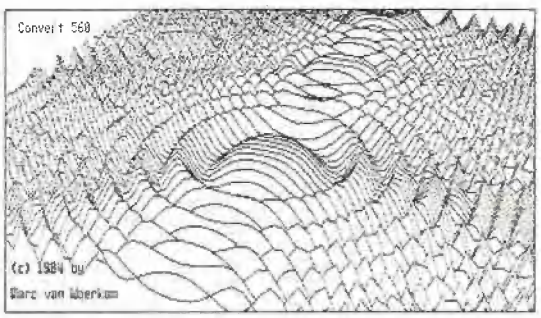


Bild 2: Gestauchtes DHGR-Bild

CONVERT560

```
BSAVE CONVERT560, A768, L166
Mit BLOAD CONVERT560 laden
Mit CALL 768 aufrufen
```

```
1   *       CONVERT560
2   *       ==========
3   *-------------------------------
4   *
5   * Eine Double-Hires-Utility
6   * von Marc van Woerkom 1985
7   *-------------------------------
8   *
9   * Kurzbeschreibung:
10  *
11  * CONVERT560 bietet eine
12  * einfache Möglichkeit,
13  * Double-Hires-Bilder aus-
14  * zudrucken, denn es
15  * wandelt ein DHGR-Bild so
16  * um, daß danach die linke
17  * Hälfte im HGR1-Bereich
18  * und die rechte Hälfte im
19  * HGR2-Bereich abgelegt
20  * ist. Diese gewöhnlichen
21  * Hires-Bilder können dann
22  * problemlos mit den schon
23  * existierenden, normalen
24  * Graphik-Treibern ausge-
25  * druckt werden.
26  *
27  * Mit CALL 768 aufrufen.
28  *-------------------------------
29
30  * Softswitches
31
32  PAGE1    EQU  $C054
33  PAGE2    EQU  $C055
34  HIRES    EQU  $C057
35  SET80COL EQU  $C00D
36  CLR80COL EQU  $C00C
37  CHK80COL EQU  $C01F
38  SET80STO EQU  $C001
39  CLR80STO EQU  $C000
40  CHK80STO EQU  $C018
41
42  * Zeropage-Adressen
43
```

Fortsetzung Seite 30

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

Hüthig-FACHBUCH-TIP

Lohnsteuerjahresausgleich? - Einkommensteuererklärung?

Mein C-64 hilft mir dabei!

von Ilona Kwiatkowski, Peter Vogt

1985, 188 S., kart., DM 38,—
ISBN 3-7785-1084-3

Das vorliegende Buch zeigt dem Leser, wie er seinen Hobbycomputer beim Erstellen des Lohnsteuerjahresausgleichs einsetzen kann. Die Autoren sind dabei von einer „Durchschnittsfamilie" ausgegangen, damit die Belange jedes Lohnempfängers gebührend berücksichtigt werden. Das Buch enthält das komplette Listing des Programms, das auf jedem gängigen Hobbycomputer ausgeführt werden kann. Es ist sowohl für den EDV-Hobbyisten als auch für den Leser geschrieben, der sich mit der EDV zwar nur am Rande beschäftigt, aber seinen Heimcomputer zur Durchführung seines Lohnsteuerjahresausgleichs einsetzen möchte.
Das Buch enthält neben grundsätzlichen Bemerkungen zum Lohnsteuerjahresausgleich eine Beschreibung des Programms und für denjenigen, der nur das Programm benutzen möchte, zusätzlich eine ausführliche Bedienungsanleitung. Es werden Hinweise gegeben, wie das Programm nach individuellen Bedürfnissen abgeändert oder erweitert werden kann.

BESTELLCOUPON

Bitte ausfüllen und an Hüthig Vertriebsservice, Postfach 102869 · 6900 Heidelberg schicken.

Buchtitel

Name

Straße

Ort

Unterschrift

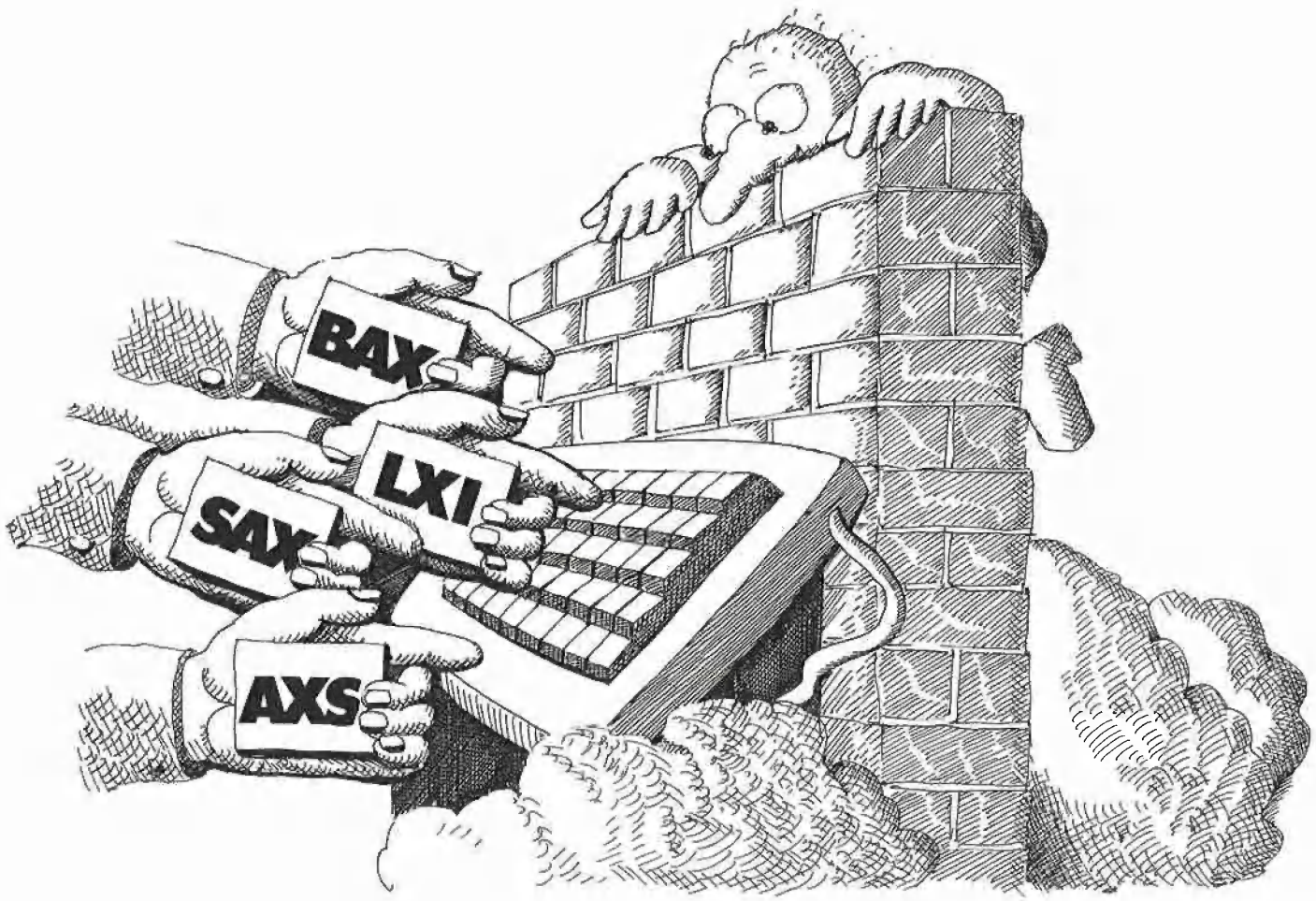
```
                        44  YLine    EQU  $F9
                        45  Hgrl     EQU  $FA
                        46  LPic     EQU  $FC
                        47  RPic     EQU  $FE
                        48
                        49  * Programm-Adressen
                        50
                        51  HGR1     EQU  $2000
                        52
                        53  Buffer   EQU  $300-80
                        54  LeftPic  EQU  $2000
                        55  RightPic EQU  $4000
                        56
                        57           ORG  $300
                        58  *
                        59  * Retten der Prozessorregister.
                        60  *
                        61  * Ferner wird der Zustand
                        62  * der Softswitches 80STO
                        63  * und 80COL gerettet;
                        64  * dann werden diese zwei und
                        65  * HIRES aktiviert, um den
                        66  * Zugriff auf den HGR1-Bereich
                        67  * des 64K-Zusatzspeichers
                        68  * (AUX) zu ermöglichen.
                        69  *
0300: 8D A1 03          70  Convert  STA  Regs
0303: 8E A2 03          71           STX  Regs+1
0306: 8C A3 03          72           STY  Regs+2
0309: AD 18 C0          73           LDA  CHK80STO
030C: 8D A4 03          74           STA  Status
030F: AD 1F C0          75           LDA  CHK80COL
0312: 8D A5 03          76           STA  Status+1
0315: 8D 01 C0          77           STA  SET80STO
0318: 8D 0D C0          78           STA  SET80COL
031B: 8D 57 C0          79           STA  HIRES
                        80  *
                        81  * Hauptschleife:
                        82  *
                        83  * Jede der 192 Zeilen wird
                        84  * einzeln konvertiert und in
                        85  * die Zielbereiche geschoben.
                        86  *
031E: A2 00             87           LDX  #0
0320: 86 F9             88  Conv2    STX  YLine
0322: 8A                89           TXA
                        90  *
                        91  * Berechnung der Startadresse
                        92  * der Zeile im Akkumulator und
                        93  * deren Speicherung im Pointer
                        94  * Hgrl/Hgrl+1.
                        95  *
                        96  * (Es handelt sich hierbei
                        97  * aus Platzgründen um eine
                        98  * modifizierte Version von
                        99  * HPOSN ($F411) und nicht um
                        100 * eine schnelle, aber sehr viel
                        101 * Speicherplatz verschlingende
                        102 * Lookup-Table!)
                        103 *
0323: 29 C0             104          AND  #%11000000
0325: 85 FA             105          STA  Hgrl
0327: 4A                106          LSR
0328: 4A                107          LSR
0329: 05 FA             108          ORA  Hgrl
032B: 85 FA             109          STA  Hgrl
032D: 8A                110          TXA
032E: 85 FB             111          STA  Hgrl+1
0330: 0A                112          ASL
0331: 0A                113          ASL
0332: 0A                114          ASL
0333: 26 FB             115          ROL  Hgrl+1
0335: 0A                116          ASL
0336: 26 FB             117          ROL  Hgrl+1
0338: 0A                118          ASL
0339: 66 FA             119          ROR  Hgrl
033B: A5 FB             120          LDA  Hgrl+1
033D: 29 1F             121          AND  #$1F
                        122 *
                        123 * Setzen der Pointer
                        124 * Hgrl/Hgrl+1, LPic/LPic+1
                        125 * und RPic/RPic+1.
                        126 *
033F: A8                127          TAY
0340: 18                128          CLC
0341: 69 20             129          ADC  #>HGR1
0343: 85 FB             130          STA  Hgrl+1
0345: 98                131          TYA
0346: 69 20             132          ADC  #>LeftPic
0348: 85 FD             133          STA  LPic+1
034A: 98                134          TYA
034B: 69 40             135          ADC  #>RightPic
034D: 85 FF             136          STA  RPic+1
034F: A5 FA             137          LDA  Hgrl
0351: 85 FC             138          STA  LPic
0353: 85 FE             139          STA  RPic
                        140 *
                        141 * Laden der jetzigen Double-
                        142 * Hires-Zeile in den 80 Bytes
                        143 * fassenden Puffer, und zwar
                        144 * so, daß die ersten 40 Bytes
                        145 * des Puffers die linke und
                        146 * die restlichen 40 Bytes die
                        147 * rechte Hälfte der Zeile
                        148 * enthalten.
                        149 *
0355: A0 27             150          LDY  #39
0357: A2 4F             151          LDX  #79
0359: 2C 54 C0          152  Conv3    BIT  PAGE1
035C: B1 FA             153          LDA  (Hgrl),Y
035E: 9D B0 02          154          STA  Buffer,X
0361: CA                155          DEX
0362: 2C 55 C0          156          BIT  PAGE2
0365: B1 FA             157          LDA  (Hgrl),Y
0367: 9D B0 02          158          STA  Buffer,X
036A: CA                159          DEX
036B: 88                160          DEY
036C: 10 EB             161          BPL  Conv3
                        162 *
                        163 * Softswitch PAGE1 aktivieren,
                        164 * damit jetzt nur noch auf den
                        165 * Hauptspeicher zugegriffen
                        166 * werden kann.
                        167 *
                        168 * Verschieben des Puffer-
                        169 * inhalts in die jeweiligen
                        170 * Speicherbereiche für die
                        171 * linke und rechte Bildhälfte.
                        172 *
036E: 8D 54 C0          173          STA  PAGE1
                        174 *
0371: A0 27             175          LDY  #39
0373: B9 B0 02          176  Conv4    LDA  Buffer,Y
0376: 91 FC             177          STA  (LPic),Y
0378: B9 D8 02          178          LDA  Buffer+40,Y
037B: 91 FE             179          STA  (RPic),Y
037D: 88                180          DEY
037E: 10 F3             181          BPL  Conv4
                        182 *
                        183 * Erniedrigung des Zeilen-
                        184 * pointers und Rücksprung,
                        185 * falls nicht alle Zeilen
                        186 * konvertiert sind.
                        187 *
0380: A6 F9             188          LDX  YLine
0382: E8                189          INX
0383: E0 C0             190          CPX  #192
0385: 90 99             191          BCC  Conv2
                        192 *
                        193 * Wiederherstellen der Register-
                        194 * inhalte und 80-Zeichen-
                        195 * Softswitches.
                        196 *
                        197 * Danach erfolgt der Rück-
                        198 * sprung zum aufrufenden
                        199 * Applesoft-Programm.
                        200 *
0387: AD A4 03          201          LDA  Status
038A: 30 03             202          BMI  Conv5
038C: 8D 00 C0          203          STA  CLR80STO
038F: AD A5 03          204  Conv5    LDA  Status+1
0392: 30 03             205          BMI  Conv6
0394: 8D 0C C0          206          STA  CLR80COL
0397: AD A1 03          207  Conv6    LDA  Regs
039A: AE A2 03          208          LDX  Regs+1
039D: AC A3 03          209          LDY  Regs+2
03A0: 60                210          RTS
                        211 *
                        212  Regs     DS   3
                        213  Status   DS   2

166 Bytes
```

Disassembler für „geheime“ 6502-Opcodes

von Christoph Bregler

Der hier vorgestellte Disassembler ist eine Erweiterung des alten Monitor-Disassemblers um 93 zusätzliche 6502-NMOS-Opcodes, die in den Datenblättern und in den meisten Assembler-Büchern nicht beschrieben werden.
Die Befehle sind auf dem Apple II, II+ und IIe verfügbar, da diese Geräte im Gegensatz zum IIc, der den 65C02-CMOS-Prozessor enthält, noch die alte NMOS-Version benutzen.



Bei dem normalen 6502-Prozessor sind von 256 möglichen Codes nur 151 als Assembler-Befehle definiert. Die CPU interpretiert aber aus den restlichen Codes auch Befehle, die zwar zum Teil unsinnig sind, aber auch manche nützliche Operationen enthalten.
Mit dem vorhandenen Monitor-Disassembler kann man ein Programm mit solchen Befehlen nicht auslisten, da dann nur „???" ausgedruckt wird. Ich habe nun einen „Extended Disassembler" (**EDA**) programmiert, der die zusätzlichen Befehle richtig ausgibt.
Nach dem Eintippen des Programms sollte es mit „BSAVE EDA, A$94C8, L$0138" auf Diskette gespeichert werden. Durch „BRUN EDA" initialisiert sich die Routine selbst und kann dann vom Monitor aus wie der „L"-Befehl aufgerufen werden, wobei das „L" durch Ctrl-Y zu ersetzen ist.
EDA ist keine gepatchte Language-Card-Version des alten Disassemblers, sondern wird in der Page 3 und direkt unter dem DOS-Puffer abgespeichert, so daß er verschwindend wenig vom freien Speicherplatz in Anspruch nimmt.
Somit läuft EDA auch auf Apple-II-Versionen mit nur 48K RAM oder unter ProDOS, das die Language-Card benutzt.

Im folgenden sind zwei Anwendungsbeispiele der zusätzlichen Op-Codes näher erläutert:

DEC/CMP adr.
INC/SBC adr.

Diese Befehle erniedrigen oder erhöhen den Inhalt einer Speicheradresse und vergleichen das Ergebnis mit dem Akkumulator.
Solche Operationen sind oft für die Ausführung von Schleifen nützlich, in denen z.B. ein Pointer von einer Anfangsadresse auf eine Endadresse heruntergezählt wird. Dies ist nun mit einem einzigen Befehl möglich.
Beispiel 1 zeigt eine solche Schleife, die einen Speicherbereich löscht.

AXM #Operand

Dieser Befehl verknüpft Akkumulator und X-Register durch die AND-Funktion, zieht davon den Operanden ab und speichert das Ergebnis in das X-Register. Das Carry-Flag muß nicht gesetzt oder gelöscht werden. Wird der Akkumulator vorher mit $FF geladen, so schafft dieser Befehl die Möglichkeit, direkt im X-Register zu addieren oder zu subtrahieren.
Beispiel 2 demonstriert diesen Befehl und den Zeitgewinn gegenüber der Benutzung der „echten" Codes.

Eine Auflistung aller 93 zusätzlichen Codes kann der **Tabelle 1** entnommen werden*.
Bei vielen Befehlen handelt es sich nur um eine Aneinanderreihung zweier normaler 6502-Befehle, wobei die Mnemonics dann aus den zwei Befehlen und einem Schrägstrich bestehen.
Wegen der fehlenden Festlegung habe ich für die restlichen Codes die Mnemonics frei definiert. Bei ihnen druckt EDA dann noch ein „§" davor, um sie von den anderen Befehlen abzusetzen.
Alle übrigen nicht aufgelisteten Codes „hängen den Prozessor auf" (z.B. $02). Man könnte sie höchstens als HALT-Befehle benutzen, bei denen der Prozessor dann wartet, bis ein Reset oder Interrupt erfolgt.

* Eine analoge Matrixtabelle findet man in „incider", 4/85, S. 121

Beispiel 1

```
                1   * X = High-Byte der Anfangsadresse-1
                2   * A = High-Byte der Endadresse
                3
A0 00           4           LDY  #0
84 06           5           STY  $6
85 07           6           STA  $7
98              7   LP1     TYA
91 06           8   LP2     STA  ($6),Y
C8              9           INY
D0 FB           10          BNE  LP2
8A              11          TXA
C7 07           12          DEC/CMP $7
90 F5           13          BCC  LP1
```

Beispiel 2

```
                1   * Subtraktion (z.B. X - 7 -> X)
                2   * 4 Mikrosekunden
                3
A9 FF           4           LDA  #$FF
CB 07           5           AXM  #7         ;C=0 bei Überlauf
                6
                7   * Subtraktion mit "echten" Opcodes
                8   * 8 Mikrosekunden
                9
8A              10          TXA
38              11          SEC
E9 07           12          SBC  #7
AA              13          TAX
                14
                15  * Addition (z.B. X + 13 -> X)
                16
A9 FF           17          LDA  #$FF
CB F3           18          AXM  #-13       ;C=1 bei Überlauf
                19  * Im Zweierkomplemet ist -13 = $F3
```

Tabelle 1: Die „geheimen" 6502-Opcodes

Code:	Mnemonic:	Adress.:	Zeit:	Funktion:
03	ASL/ORA	(ind.,X)	8	ASL Adr. & ORA Adr.
07	-"-	zeropage	5	-"-
0F	-"-	absolute	6	-"-
13	-"-	(ind.),Y	8	-"-
17	-"-	z.page,X	6	-"-
1B	-"-	absol.,Y	7	-"-
1F	-"-	absol.,X	7	-"-
23	ROL/AND	(ind.,X)	8	ROL Adr. & AND Adr.
27	-"-	zeropage	5	-"-
2F	-"-	absolute	6	-"-
33	-"-	(ind.),Y	8	-"-
37	-"-	z.page,X	6	-"-
3B	-"-	absol.,Y	7	-"-
3F	-"-	absol.,X	7	-"-
43	LSR/EOR	(ind.,X)	8	LSR Adr. & EOR Adr.
47	-"-	zeropage	5	-"-
4F	-"-	absolute	6	-"-
53	-"-	(ind.),Y	8	-"-
57	-"-	z.page,X	6	-"-
5B	-"-	absol.,Y	7	-"-
5F	-"-	absol.,X	7	-"-
63	ROR/ADC	(ind.,X)	8	ROR Adr. & ADC Adr.
67	-"-	zeropage	5	-"-
6F	-"-	absolute	6	-"-
73	-"-	(ind.),Y	8	-"-
77	-"-	z.page,X	6	-"-
7B	-"-	absol.,Y	7	-"-
7F	-"-	absol.,X	7	-"-
83	STX/STA	(ind.,X)	6	Store (A AND X) Adr.
87	-"-	zeropage	3	-"-
8F	-"-	absolute	4	-"-
97	-"-	z.page,Y	4	-"-
A3	LDX/LDA	(ind.,X)	6	LDX Adr. & LDA Adr.
A7	-"-	zeropage	3	-"-
AB	-"-	immed.	2	-"-
AF	-"-	absolute	4	-"-
B3	-"-	(ind.),Y	5	-"-

```
B7          -"-         z.page,Y   4          -"-
BF          -"-         absol.,Y   5          -"-
C3      DEC/CMP         (ind.,X)   8    DEC Adr. & CMP Adr.
C7          -"-         zeropage   5          -"-
CF          -"-         absolute   6          -"-
D3          -"-         (ind.),Y   8          -"-
D7          -"-         z.page,X   6          -"-
DB          -"-         absol.,Y   7          -"-
DF          -"-         absol.,X   7          -"-
E3      INC/SBC         (ind.,X)   8    INC Adr. & SBC Adr.
E7          -"-         zeropage   5          -"-
EF          -"-         absolute   6          -"-
F3          -"-         (ind.),Y   8          -"-
F7          -"-         z.page,X   6          -"-
FB          -"-         absol.,Y   7          -"-
FF          -"-         absol.,X   7          -"-
93      BAX             (ind.),Y   6    Speichere <B>itØ/3 von
                                        <A> AND <X> nach Adr.
4B      LSA             immed.     2    AND Adr. & LSR A
6B      ROA             immed.     2    AND Adr. & ROR A
8B      LXI             immed.     2    <L>ade A mit <X> AND
                                        <I>mmedediate
9B      SAX             impl.      5    Lade <S> mit <A> AND <X>
                                        (besteht aus 3 Bytes, von
                                        denen die letzten 2 Bytes
                                        beliebig sein können)
BB      AXS             absol.,Y   4    Lade <A>,<X>,<S> mit S
                                        AND Adr.
CB      AXM             immed.     2    Lade X mit <A> AND <X>
                                        <M>inus immediate
9C      SBY             absol.,X   5    <S>peichere <B>itØ/3 von
                                        <Y> nach Adr.
9E      SHX             absolute   5    <S>peichere <H>igh Byte+1
                                        der Adr. AND <X> nach Adr.
9F      HXA             absolute   5    Speichere <H>igh Byte+1
                                        der Adr. AND <X> AND <A>
                                        nach Adr.
ØB,2B   AND             immed.
EB      SBC             immed.
1A,3A,5A NOP            impl.
7A,DA,FA
8Ø,82,C2 NOP            impl.          besteht aus 2 Bytes, von
E2,Ø4,14                               denen das letzte Byte
34,44,54                               beliebig sein kann
64,74,D4
F4,89
ØC,1C,3C NOP            impl.          besteht aus 3 Bytes, von
5C,7C,DC                               denen die letzten 2 Bytes
FC                                     beliebig sein können
```

EDA

```
1   ***********************************
2   *                                 *
3   * E D A :                         *
4   *                                 *
5   * (E)xtended (D)is-(A)ssembler    *
6   *                                 *
7   * von Christoph Bregler           *
8   *       Oktober 1984              *
9   *                                 *
1Ø  *                                 *
11  * für die zusätzlichen            *
12  * 65Ø2-Codes                      *
13  * (keine 65CØ2-Codes!)            *
14  *                                 *
15  * Speicherbelegung: $ØØØ-$Ø3CC    *
16  *                   $95B8-$95FF   *
17  *                                 *
18  * Benutzung:                      *
19  * Wie den Monitor-List-Befehl.    *
2Ø  * Anstatt 'L' wird Ctrl-Y ein-    *
21  * gegeben.                        *
22  *                                 *
23  ***********************************
24
25
26  LMNEM    EQU  $2C
27  FORMAT   EQU  $2E
28  LENGTH   EQU  $2F
29  CSWL     EQU  $36
3Ø  CSWH     EQU  $37
31  PCL      EQU  $3A
32  PCH      EQU  $3B
33  HIMEM    EQU  $73
34  XSAVE    EQU  $EE
35  XSAV1    EQU  $EF
36  LCOUNT   EQU  $FC
37  COUNT    EQU  $FD
38  CSWL1    EQU  $FE
39  CSWH1    EQU  $FF
4Ø
41  INSDS1   EQU  $F882
42  INSDS2   EQU  $F88C
43  INSTDSP  EQU  $F8DØ
44  NXTCOL   EQU  $F8F5
45  PCADJ    EQU  $F953
46  COUT     EQU  $FDED
47  A1PC     EQU  $FE75
48
5Ø          ORG  $94C8
51
52  * Schiebt EDA in Page 3,
53  * setzt Ctrl-Y-Pointer und HIMEM
54  * für die Opcode-Tabellen
55
94C8: A2 CD     56          LDX  #FMT3-ROUT
94CA: BD EA 94  57  INITLP  LDA  ROUT-1,X
94CD: 9D FF Ø2  58          STA  $2FF,X
94DØ: CA        59          DEX
94D1: DØ F7     6Ø          BNE  INITLP
94D3: A9 4C     61          LDA  #$4C
94D5: 8D F8 Ø3  62          STA  $3F8
94D8: A9 ØØ     63          LDA  #<EXTLIST
94DA: 8D F9 Ø3  64          STA  $3F9
94DD: A9 Ø3     65          LDA  #>EXTLIST
94DF: 8D FA Ø3  66          STA  $3FA
94E2: A9 B8     67          LDA  #<FMT3
94E4: 85 73     68          STA  HIMEM
94E6: A9 95     69          LDA  #>FMT3
94E8: 85 74     7Ø          STA  HIMEM+1
94EA: 6Ø        71          RTS
72
73  ROUT
74
75  * EDA-Beginn bei $3ØØ
76
77          ORG  $3ØØ
78
79  * Ruft 2Ømal (2Ø Zeilen) INSTDSP2
8Ø  * auf und erhöht mit PCADJ den
81  * Befehlszähler
82
Ø3ØØ: A5 36     83  EXTLIST  LDA  CSWL
Ø3Ø2: 85 FE     84          STA  CSWL1
Ø3Ø4: A5 37     85          LDA  CSWH
Ø3Ø6: 85 FF     86          STA  CSWH1
Ø3Ø8: A9 14     87          LDA  #2Ø
Ø3ØA: 85 FC     88          STA  LCOUNT
Ø3ØC: 2Ø 75 FE  89          JSR  A1PC
Ø3ØF: 2Ø 47 Ø3  9Ø  EXTLIST1 JSR  INSTDSP2
Ø312: 2Ø 2A Ø3  91          JSR  STCSW2
Ø315: 2Ø 53 F9  92          JSR  PCADJ
Ø318: 85 3A     93          STA  PCL
Ø31A: 84 3B     94          STY  PCH
Ø31C: C6 FC     95          DEC  LCOUNT
Ø31E: DØ EF     96          BNE  EXTLIST1
Ø32Ø: 6Ø        97          RTS
98
99  * Biegt CSWL-Pointer zu COUT2 um
1ØØ
Ø321: A2 33     1Ø1 STCSW1  LDX  #<COUT2
Ø323: 86 36     1Ø2         STX  CSWL
Ø325: A2 Ø3     1Ø3         LDX  #>COUT2
Ø327: 86 37     1Ø4         STX  CSWH
Ø329: 6Ø        1Ø5         RTS
1Ø6
1Ø7 * Setzt CSWL zurück, so daß ein
1Ø8 * eventuelles DOS nichts "merkt"
1Ø9
Ø32A: A6 FE     11Ø STCSW2  LDX  CSWL1
Ø32C: 86 36     111         STX  CSWL
Ø32E: A6 FF     112         LDX  CSWH1
Ø33Ø: 86 37     113         STX  CSWH
Ø332: 6Ø        114         RTS
115
116 * COUT-Routine, die sowohl ohne
117 * DOS als auch mit DOS nicht
118 * "abzuhängen" ist und nach dem
119 * Xten Aufruf (X=COUNT) wieder zu
12Ø * EDA springt
121
Ø333: 86 EE     122 COUT2   STX  XSAVE
Ø335: 2Ø 2A Ø3  123         JSR  STCSW2
Ø338: 2Ø ED FD  124         JSR  COUT
Ø33B: 2Ø 21 Ø3  125         JSR  STCSW1
```

```
033E: A6 EE      126           LDX  XSAVE
0340: C6 FD      127           DEC  COUNT
0342: D0 02      128           BNE  RET
0344: 68         129           PLA
0345: 68         130           PLA
0346: 60         131 RET       RTS
                 132
                 133 * Prüft, ob ein nicht definierter
                 134 * Befehl anliegt
                 135
0347: 20 82 F8   136 INSTDSP2  JSR  INSDS1
034A: C9 10      137           CMP  #$10        ;?-CODE
034C: D0 28      138           BNE  CONT
034E: 20 21 03   139           JSR  STCSW1
                 140
                 141 * Prüft, ob ein Befehl anliegt,
                 142 * der durch die folgende De-
                 143 * codierung falsch ausgedruckt wird
                 144
0351: B1 3A      145           LDA  (PCL),Y
0353: A2 10      146           LDX  #16
0355: DD EF 95   147 EXTLP     CMP  EXTBL,X
0358: F0 50      148           BEQ  EXT
035A: CA         149           DEX
035B: 10 F8      150           BPL  EXTLP
                 151
035D: 4A         152           LSR
035E: B0 19      153           BCS  ODD
                 154
                 155 * Decodierung der NOP- und CRASH-
                 156 * Befehle
                 157
0360: 4A         158           LSR
0361: 29 03      159           AND  #3
0363: AA         160           TAX
0364: F0 0C      161           BEQ  CRASH
0366: BD B8 95   162 NOP       LDA  FMT3,X
0369: 85 2E      163           STA  FORMAT
036B: 29 03      164           AND  #3
036D: 85 2F      165           STA  LENGTH
036F: A9 3E      166           LDA  #$3E        ;NOP+?
0371: 38         167           SEC
0372: 90 F2      168 CRASH     BCC  NOP
0374: 49 10      169           EOR  #$10        ;?-CODE
0376: 4C D3 F8   170 CONT      JMP  INSTDSP+3
                 171
                 172 * Decodierung der "Doppel-Mnemo-
                 173 * nic"-Befehle (z.B. INC/SBC)
                 174
0379: 2A         175 ODD       ROL
037A: 29 F0      176           AND  #$F0
037C: 09 05      177           ORA  #5
037E: 20 8E F8   178           JSR  INSDS2+2
0381: 48         179           PHA
0382: B1 3A      180           LDA  (PCL),Y
0384: 29 F0      181           AND  #$F0
0386: 09 06      182           ORA  #6
0388: 20 8E F8   183           JSR  INSDS2+2
038B: 48         184           PHA
038C: B1 3A      185           LDA  (PCL),Y
038E: AA         186           TAX
038F: 4A         187           LSR
0390: 4A         188           LSR
0391: 4A         189           LSR
0392: 8A         190           TXA
0393: 69 FE      191           ADC  #-2
0395: 20 8E F8   192           JSR  INSDS2+2
0398: A9 0F      193           LDA  #15         ;COLM.-8
039A: 85 FD      194           STA  COUNT
039C: 68         195           PLA
039D: 20 D3 F8   196           JSR  INSTDSP+3
03A0: A9 AF      197           LDA  #"/"
03A2: 20 ED FD   198           JSR  COUT
03A5: A2 03      199           LDX  #3
03A7: 4C E9 F8   200           JMP  $F8E9
                 201
                 202 * Decodierung der "unregelmäßi-
                 203 * gen" Codes nach den Opcode-
                 204 * Tabellen in PAGE $95
                 205
03AA: BD BC 95   206 EXT       LDA  FMT4,X
03AD: 86 EF      207           STX  XSAV1
03AF: 85 2E      208           STA  FORMAT
03B1: 29 03      209           AND  #3
03B3: 85 2F      210           STA  LENGTH
03B5: A9 0D      211           LDA  #13         ;COLM.-8
03B7: 85 FD      212           STA  COUNT
03B9: A9 52      213           LDA  #$52        ;$-CODE
```

```
03BB: 20 D3 F8   214           JSR  INSTDSP+3
03BE: A6 EF      215           LDX  XSAV1
03C0: BD CD 95   216           LDA  MNEML1,X
03C3: 85 2C      217           STA  LMNEM
03C5: BD DE 95   218           LDA  MNEMR1,X
03C8: A2 03      219           LDX  #3
03CA: 4C F3 F8   220           JMP  NXTCOL-2
                 221
                 222 RTEND
                 223
                 224 * Tabellen, die in PAGE $95
                 225 * stehen:
                 226
                 227           ORG  ROUT+RTEND-EXTLIST
                 228
                 229 * Tabelle für die Länge der
                 230 * NOP-Befehle
                 231
95B8: 81 81 00   232 FMT3      HEX  81810082
95BB: 82
                 233
                 234 * Tabelle für die Adressierungs-
                 235 * arten:
                 236
95BC: 81 81 81   237 FMT4      HEX  818181      ;$XX
95BF: 4D         238           HEX  4D          ;($XX),Y
95C0: 81         239           HEX  81          ;$XX
95C1: 21 21 21   240           HEX  2121212121 ;#$XX
95C4: 21 21
95C6: 82         241           HEX  82          ;$XXXX
95C7: 86         242           HEX  86          ;$XXXX,Y
95C8: 21 21      243           HEX  2121        ;#$XX
95CA: 92         244           HEX  92          ;$XXXX,X
95CB: 82 82      245           HEX  8282        ;$XXXX
                 246
                 247 * Tabelle für die zusätzlichen
                 248 * Mnemonics:
                 249
95CD: 7C 7C 7C   250 MNEML1    HEX  7C7C7C      ;NOP
95D0: 18         251           HEX  18          ;BAX
95D1: 7C         252           HEX  7C          ;NOP
95D2: 13 13      253           HEX  1313        ;AND
95D4: 6D         254           HEX  6D          ;LSA
95D5: 9C         255           HEX  9C          ;ROA
95D6: 6E         256           HEX  6E          ;LXI
95D7: A0         257           HEX  A0          ;SAX
95D8: 16         258           HEX  16          ;AXS
95D9: 16         259           HEX  16          ;AXM
95DA: A0         260           HEX  A0          ;SBC
95DB: A0         261           HEX  A0          ;SBY
95DC: A2         262           HEX  A2          ;SHX
95DD: 4E         263           HEX  4E          ;HXA
                 264
95DE: 22 22 22   265 MNEMR1    HEX  222222      ;NOP
95E1: B2         266           HEX  B2          ;BAX
95E2: 22         267           HEX  22          ;NOP
95E3: CA CA      268           HEX  CACA        ;AND
95E5: 04         269           HEX  04          ;LSA
95E6: 04         270           HEX  04          ;ROA
95E7: 54         271           HEX  54          ;LXI
95E8: B2         272           HEX  B2          ;SAX
95E9: 68         273           HEX  68          ;AXS
95EA: 5C         274           HEX  5C          ;AXM
95EB: C8         275           HEX  C8          ;SBC
95EC: F4         276           HEX  F4          ;SBY
95ED: 72         277           HEX  72          ;SHX
95EE: 44         278           HEX  44          ;HXA
                 279
                 280 * Tabelle der Codes mit den
                 281 * Mnemonics von oben:
                 282
95EF: 82 C2 E2   283 EXTBL     HEX  82C2E2      ;NOP
95F2: 93         284           HEX  93          ;BAX
95F3: 89         285           HEX  89          ;NOP
95F4: 0B 2B      286           HEX  0B2B        ;AND
95F6: 4B         287           HEX  4B          ;LSA
95F7: 6B         288           HEX  6B          ;ROA
95F8: 8B         289           HEX  8B          ;LXI
95F9: 9B         290           HEX  9B          ;SAX
95FA: BB         291           HEX  BB          ;AXS
95FB: CB         292           HEX  CB          ;AXM
95FC: EB         293           HEX  EB          ;SBC
95FD: 9C         294           HEX  9C          ;SBY
95FE: 9E         295           HEX  9E          ;SHX
95FF: 9F         296           HEX  9F          ;HXA
                 297
                 298 DOSBUF

312 Bytes
```

Ein gewöhnlicher Mikroprozessor kann im allgemeinen keine höheren mathematischen Funktionen als Maschinenbefehl ausführen. Ausgenommen sind natürlich die Arithmetik-Prozessoren. Neuere Mikroprozessoren wie der MC 68000 von Motorola haben immerhin Divisions- und Multiplikationsbefehle für Ganzzahlen in ihrem Befehlssatz. Fließkommaarithmetik aber beherrschen diese noch immer nicht. Wie man höhere mathematische Funktionen in Pascal beschreiben kann, soll dieser Artikel zeigen.

Transzendente Funktionen in Pascal

von Dieter Geiß

Ergebnisse von Berechnungen höherer mathematischer Funktionen sind fast nie genau, da sie Fließkommaarithmetik voraussetzen. Die sogenannten reellen Zahlen können durch keine noch so große Anzahl von Bits in jedem Zahlenbereich genau dargestellt werden. Die Eulersche Zahl e z. B. hat unendlich viele Nachkommastellen. Das liegt daran, daß sich e aus einer Potenzreihe, der sog. Exponentialreihe, berechnen läßt. Diese Reihe hat unendlich viele Glieder. Die ersten lauten:
1 + 1 / 1! + 1 / 2! + 1 / 3! + 1 / 4! ...
Entsprechend ist eine Potenz von e, z.B.:
e ↑ x = 1 + x / 1! + x ↑ 2 / 2! + x ↑ 3 / 3! ...

Ähnliches gilt für den Sinus, den Cosinus und für andere Funktionen wie den Sinus Hyperbolicus (sinh) oder den Cosinus Hyperbolicus (cosh). Der Sinus ist also nicht irgendeine Relation zwischen Gegenkathete und Hypotenuse eines rechtwinkligen Dreiecks, sondern eine Potenzreihe, die für alle Werte x, für die man den Sinus berechnen will, gegen einen bestimmten Funktionswert konvergiert. Diesen Funktionswert nennt man dann den Sinus der Zahl x.

Das hier vorgestellte Programm zeigt die Implementierung von transzendenten Funktionen in UCSD-Pascal. Es ist als ‚Unit' geschrieben und läßt sich durch ein einfaches „uses" nach dem Programmkopf benutzen. Die Segmentnummer ist absichtlich als 29 gewählt, die damit der Nummer des in der SYSTEM.LIBRARY enthaltenen TRANSCEND entspricht.

Für den Arcustangens wurde allerdings die Abkürzung arctan und nicht atan, wie in TRANSCEND, gewählt. Das entspricht der Definition dieser Funktion in Standard-Pascal.

Hat man die ersten sieben Funktionen implementiert, lassen sich alle weiteren Funktionen mit Hilfe dieser sieben ausdrücken.

Der Sinus eines Wertes x wird in der ‚function sine' über die Potenzreihe des Sinus berechnet. Sie lautet:
sin (x) = x - x ↑ 3 / 3! + x ↑ 5 / 5! - x ↑ 7 / 7! ...
Bevor die ersten fünf Glieder dieses Ausdrucks mittels des Hornerschemas berechnet werden, wird der Wert x, auch das Argument genannt, in den Bereich [-Pi / 2, +Pi / 2] gebracht, denn es gibt ein z und ein n, für die gelten:
x = z + n · Pi.
Das liegt an der Periodizität des Sinus. Ist der Wert des Sinus sehr klein (< 0.00025), dann gilt näherungsweise: x = sin (x). Außerdem wird in der Berechnung die Beziehung sin (-x) = - sin (x) benutzt.

Der Cosinus ist nichts anderes als ein um Pi / 2 verschobener Sinus, und so wird er auch berechnet.

Bei der Exponentialfunktion ‚exp' wird ein z und ein n gefunden, so daß gilt:
e ↑ x = e ↑ z · e ↑ (n · ln (2)) = e ↑ z · (e ↑ ln (2)) ↑ n = e ↑ z · 2 ↑ n.
Diese Darstellung ist deshalb günstig, weil der Exponent einer Realzahl intern zur Basis 2 dargestellt wird, und nicht zur Basis 10. Bevor der Wert e ↑ x berechnet wird, muß x in den Bereich [-ln (2) / 2, +ln (2) / 2] gebracht worden sein.

Ähnlich wird beim ‚ln' verfahren. Man kann nämlich ein z und ein n finden, so daß gilt:
ln (x) = ln (z) + n · ln (2).

Dazu muß ein z aus dem Bereich [0.5, 1] und n ein natürliche Zahl sein.

Die Quadratwurzel ‚sqrt' wird ebenfalls in ähnlicher Form berechnet. Es werden ein n und ein z gesucht, so daß x = z · x ↑ n ist. Für n und z gilt derselbe Bereich wie beim ‚ln'. Es ist dann
sqrt (x) = sqrt (z) · 2 ↑ (n div 2),
wenn n gerade ist, und
sqrt (x) = sqrt (z) · (2 ↑ ((n + 1) div 2)) / sqrt (2).
Die Wurzel selbst wird mit Hilfe des Newtonschen Näherungsverfahrens in drei Schritten berechnet.

Bei allen Funktionen wird das Wissen vorausgesetzt, wie die Realzahlen intern codiert sind. Dies macht sich in dem Typ Realrec bemerkbar, der zwei Varianten überlagert, nämlich die Realzahl selbst und die einzelnen Komponenten der Realzahl. Diese Komponenten sind Mantisse, Exponent und Vorzeichen.

Wer flüssig in Pascal programmieren kann und trotzdem die Berechnungen der transzendenten Funktionen nicht ganz versteht, mag sich damit trösten, daß es wohl Mathematiker gewesen sein müssen, die sich derartige Tricks haben einfallen lassen. Dazu gehören auch die vielen Kniffe, welche Rundungsfehler etwas einschränken sollen. Ich selbst habe auch nicht alles auf Anhieb verstanden.

TRANSCEND.PASCAL.SOURCE (auf Peeker-Sammeldisk # 5 als DOS-Textfile)

```
{$S+}
unit transcend; intrinsic code 29;
interface

function sin    (x : real) : real;
function cos    (x : real) : real;
function exp    (x : real) : real;
function arctan (x : real) : real;
function ln     (x : real) : real;
function log    (x : real) : real;
function sqrt   (x : real) : real;
function ld     (x : real) : real;
function tan    (x : real) : real;
function cot    (x : real) : real;
function arcsin (x : real) : real;
function arccos (x : real) : real;
function arccot (x : real) : real;
function sinh   (x : real) : real;
function cosh   (x : real) : real;
function tanh   (x : real) : real;
function coth   (x : real) : real;
function arsinh (x : real) : real;
function arcosh (x : real) : real;
function artanh (x : real) : real;
function arcoth (x : real) : real;
function xpwry  (x, y : real) : real;

implementation
{-------------------------------------------}
const PIdiv2     = 1.57Ø7963;
      Sqrt2div2  = Ø.7Ø71Ø678;

type  Realrec    = packed record
                     case integer of
                       Ø : (Realval : real);
                       1 : (Mant1   : integer;
                            Mant2   : Ø..127;
                            Exp     : Ø..255;
                            Sign    : Ø..1)
                   end; {Realrec}

{-------------------------------------------}
function sine (X : real; Sign : integer) : real;

const RezPI  =  Ø.3183Ø99;        {1/PI}
      PI1    =  3.14Ø625;         {PI1 + PI2 = PI}
      PI2    =  Ø.ØØØ96765364;
      Fak9   =  Ø.ØØØØØ26Ø1903;   {1/9!}
      Fak7   = -Ø.ØØØ1980742Ø;    {1/7!}
      Fak5   =  Ø.ØØ8333Ø25;      {1/5!}
      Fak3   = -Ø.16666657;       {1/3!}

var   Xsqr   : real;
      Npi    : integer;

begin {sine}
  if X > 1ØØØØØØ.Ø then halt;
  Npi := round (X * RezPI);
  if odd (Npi) then Sign := - Sign;
  X := X - Npi * PI1 - Npi * PI2;
  if abs (X) >= Ø.ØØØ25 then
  begin
    Xsqr := X * X;
    X    := X + X * ((((Fak9 * Xsqr + Fak7) *
            Xsqr + Fak5) * Xsqr + Fak3) * Xsqr)
  end; {if}
  sine := Sign * X
end; {sine}

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

function sin {(x : real) : real};

begin {sin}
  if x < Ø.Ø
  then sin := sine (-X, -1)
  else sin := sine ( X,  1)
end; {sin}
{-------------------------------------------}

function cos {(x : real) : real};

begin {cos}
  cos := sine (abs (x) + PIdiv2, 1)
end; {cos}
```

```
function exp {(x : real) : real};

const RezLN2 =  1.442695;       {1/LN (2)}
      LN2_1  =  Ø.6933593;      {LN2_1 + LN2_2 = LN (2)}
      LN2_2  = -Ø.ØØØ21219444;
      RC1    =  Ø.ØØ41ØØ2895;
      RC2    =  Ø.Ø4998718;

var   N      : integer;
      Xsqr   : real;
      R      : real;
      Tricky : Realrec;

begin {exp}
  if x = Ø.Ø
  then exp := 1.Ø
  else
    begin
      N              := round (x * RezLN2);
      x              := x - N * LN2_1 - N * LN2_2;
      Xsqr           := x * x;
      R              := x * (RC1 * Xsqr + Ø.25);
      Tricky.Realval := Ø.5 + R / (RC2 * Xsqr + Ø.5 - R);
      Tricky.Exp     := Tricky.Exp + (N + 1);
      exp            := Tricky.Realval
    end {else}
end; {exp}

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

function arctan {(x : real) : real};

const Maxloop = 8;

var Xsqr : real;
    R    : real;
    Neg  : boolean;
    I    : integer;
begin
  Xsqr := x * x;
  if abs (x) > 1.Ø then Xsqr := 1.Ø / Xsqr;
  R := 2 * Maxloop + 1;
  for I := Maxloop downto 1 do R := 2.Ø * I - 1.Ø +
           I * I * Xsqr / R;
  if abs (x) > 1.Ø
  then
    begin
      Neg  := X < Ø;
      Xsqr := PIdiv2 - 1.Ø / (abs (x) * R);
      if Neg
      then arctan := -Xsqr
      else arctan :=  Xsqr
    end {if}
  else arctan := x / R
end; {arctan}

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

function ln {(x : real) : real};

const LN2_1  =  Ø.6933593;
      RC1    =  Ø.5527Ø747;
      RC2    =  6.632718;
      LN2_2  = -Ø.ØØØ21219444;

var   Tricky : Realrec;
      R1     : real;
      R2     : real;
      R3     : real;
      R4     : real;
      E      : integer;

begin {ln}
  if X <= Ø.Ø then halt;
  Tricky.Realval := x;
  E              := Tricky.Exp - 126;
  Tricky.Exp     := 126;
  x              := Tricky.Realval;
  if x > Sqrt2div2
  then R2 := (x - Ø.5 - Ø.5) / (x * Ø.5 + Ø.5)
  else
    begin
      E  := E - 1;
      R3 := X - Ø.5;
      R2 := R3 / (R3 * Ø.5 + Ø.5)
    end; {else}
  R1 := R2 * R2;
  ln := E * LN2_1 + (R2 + R2 * (R1 * RC1 /
        (RC2 - R1))) + E * LN2_2
end; {ln}
```

CP/D EDV–Anlagen GmbH *Vulkanstr. 13, 4000 Düsseldorf*

Tel.: 0211/776270 Tx.:858 8060

CP/D hat's

So reagierten Macintosh Besitzer...

Bereits auf 512K umgerüstet	512K Mac gekauft	Will jetzt aufrüsten
Ach du dickes Ei	Eijeijei	Ei wie fein

Die Aufrüstung Ihres 128K Mac auf 512K kostet bei uns:

DM **1690,—** incl. MWST.

Schicken Sie uns Ihren Macintosh. Rücksendung innerhalb von 24 Stunden. Auf Wunsch geben wir Ihnen auch Händleradressen in Ihrem Raum bekannt.

```
function log {(x : real) : real};

begin {log}
  log := ln (x) * Ø.43429448 {1 / ln (1Ø) = Ø.43429448}
end; {log}

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

function sqrt {(x : real) : real};

const RC1    = Ø.41731;
      RC2    = Ø.59Ø16;

var   R      : real;
      Tricky : Realrec;
      E      : integer;
      I      : integer;

begin {sqrt}
  if X < Ø then halt;
  if X = Ø.Ø
  then sqrt := Ø.Ø
  else
    begin
      Tricky.Realval := x;
      E              := Tricky.Exp - 126;
      Tricky.Exp     := 126;
      x              := Tricky.Realval;
      R              := RC1 + RC2 * x;
      for I := Ø to 2 do R := Ø.5 * (R + x / R);
      if odd (E) then
      begin
        R := R * Sqrt2div2;
        E := E + 1
      end; {if}
      Tricky.Realval := R;
      Tricky.Exp     := E div 2 + Tricky.Exp;
      sqrt           := Tricky.Realval
    end {else}
end; {sqrt}

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

function ld {(x : real) : real};

begin {log}
  ld := ln (x) * 1.44269504 {1 / ln (2) = 1.44269504}
end; {log}

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

function tan {(x : real) : real};

begin {tan}
  tan := sin (x) / cos (x)
end; {tan}

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

function cot {(x : real) : real};

begin {cot}
  cot := 1 / tan (x)
end; {cot}

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

function arcsin {(x : real) : real};

begin {arcsin}
  if abs (x) > 1 then halt;
  if abs (x) = 1.Ø
  then arcsin := x * PIdiv2
  else arcsin := arctan (x / sqrt (1 - sqr (x)))
end; {arcsin}

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

function arccos {(x : real) : real};

begin {arccos}
  if abs (x) > 1 then halt;
  arccos := PIdiv2 - arcsin (x)
end; {arccos}

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

function arccot {(x : real) : real};

begin {arccot}
  arccot := PIdiv2 - arctan (x)
end; {arccot}
```

```
function sinh {(x : real) : real};

begin {sinh}
  sinh := Ø.5 * (exp (x) - exp (-x))
end; {sinh}

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

function cosh {(x : real) : real};

begin {cosh}
  cosh := Ø.5 * (exp (x) + exp (-x))
end; {cosh}

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

function tanh {(x : real) : real};

begin {tanh}
  tanh := sinh (x) / cosh (x)
end; {tanh}

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

function coth {(x : real) : real};

begin {coth}
  if x = Ø then halt;
  coth := 1 / tanh (x)
end; {coth}

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

function arsinh {(x : real) : real};

begin {arsinh}
  arsinh := ln (x + sqrt (sqr (x) + 1))
end; {arsinh}

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

function arcosh {(x : real) : real};

begin {arcosh}
  if x < 1.Ø then halt;
  arcosh := ln (x + sqrt (sqr (x) - 1))
end; {arcosh}

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

function artanh {(x : real) : real};

begin {artanh}
  if abs (x) >= 1 then halt;
  artanh := Ø.5 * ln ((1 + x) / (1 - x))
end; {artanh}

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

function arcoth {(x : real) : real};

begin {arcoth}
  if abs (x) <= 1 then halt;
  arcoth := Ø.5 * ln ((x + 1) / (x - 1))
end; {arcoth}

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

function xpwry {(x, y : real) : real};

begin {xpwry}
  if x < Ø then halt;
  if y = Ø
  then xpwry := 1.Ø
  else
    if x = Ø
    then
      if Y < Ø
      then halt
      else xpwry := Ø.Ø
    else xpwry := exp (y * ln (x))
end; {xpwry}

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

begin {transcend}
end {transcend}
```

Das CP/M-Directory näher beleuchtet

Der Aufbau von 16-Sektor-CP/M-Disketten

von Dr. Jürgen B. Kehrel

Sie wollen von DOS aus auf CP/M-Textfiles zugreifen? Sie wollen gelöschte Files wiederbeleben? Sie wollen wissen, wie Ihre CP/M-Disketten belegt sind? Immer dann und in vielen anderen Fällen müssen Sie den genauen Aufbau des Directory kennen.

Die Z80-Karte von Microsoft ist eine der häufigsten Ergänzungen zum Apple II, denn sie eröffnet den Zugang zu vielen interessanten kommerziellen Programmen. Trotzdem steht in Büchern und Zeitschriften nur wenig über Interna von Apple-CP/M. Das mitgelieferte Softcard-Handbuch schweigt sich z.B. über den Aufbau einer CP/M-Diskette völlig aus. Aber da Sie leicht sehen können, daß CP/M-Disketten mit COPYA zu vervielfältigen sind (was Sie aber nicht tun sollten, s.u.), liegen Sie mit Ihrer Vermutung ganz richtig, daß zwischen DOS 3.3 und CP/M Ähnlichkeiten bestehen. Diese sind so groß, daß Sie z.B. mit DOS 3.3 CP/M-Disketten lesen können. Im Heft 3/1985 vom Peeker wurde ein praktisches Beispiel vorgestellt.

Der Diskettenaufbau

Alle folgenden Erläuterungen beziehen sich auf das 16-Sektor-CP/M, Version 2.2 von Microsoft (Digital Research). Das FORMAT-Programm erzeugt eine Diskette mit 35 Spuren (Tracks) zu je 16 Sektoren. Alle Sektorinhalte werden dabei auf $E5 gesetzt. Eine solche Diskette enthält keine Systemspuren, kein Directory und auch keine Files, anders als beim INIT-Befehl von DOS. Mit dem COPY-Programm können Sie nun auf die Spuren 0–2 analog zu DOS den CP/M-Systemcode aufbringen. Der Befehl SYSGEN existiert in der Apple-Version nicht.

Die verbleibenden 32 Spuren werden vom CP/M in 128 **Blöcke** unterteilt, die von $00 bis $7F durchnumeriert sind. Jeder Block umfaßt 1024 Bytes (1K) und setzt sich seinerseits aus 8 **Records** zu je 128 Bytes zusammen. Da nun jeder Diskettensektor aber 256 Bytes umfaßt, gehören immer 2 Records zu einem Sektor und 4 Sektoren zu einem Block. Sie können sich einfach ausrechnen, daß 4 Blöcke gleich 16 Sektoren sind und damit eine Spur ergeben (**Tabelle 1**). Die Blöcke werden von außen nach innen gezählt, so daß z.B. die Blöcke 0 bis 3 zur Spur 3, die Blöcke 4 bis 7 zu Spur 4 usw. gehören (Spur = INT (Blocknummer/4) + 3). Ein Block ist die kleinste Einheit, die von CP/M auf einmal gelesen oder geschrieben werden kann, so daß selbst Miniprogramme immer mindestens 4 Sektoren belegen, was gegenüber DOS eine gehörige Platzverschwendung bedeutet.

Die Blöcke 0 und 1 der Diskette enthalten ihr Inhaltsverzeichnis (Directory), alle übrigen Blöcke stehen zur Abspeicherung von Daten bereit. In der **Abbildung 1** sind die Blöcke vereinfachend als zusammenhängende Bereiche dargestellt. Das ist aber in der Praxis nicht so, denn zur schnelleren Verarbeitung benutzt CP/M ein sogenanntes „Sector-Interleaving" (auch Skewing genannt). Näheres dazu steht in der angegebenen Literatur. Da auch DOS 3.3 nicht die physikalische Sektornumerierung benutzt, ist der Zusammenhang zwischen DOS-Sektoren und CP/M-Blöcken nur über eine Tabelle zu erfassen (**Tabelle 2**). Wegen des unterschiedlichen „Skewing" sollten Sie auch nicht COPYA für CP/M verwenden, da CP/M dann diese Kopien sehr langsam liest. „Relativer Block" in

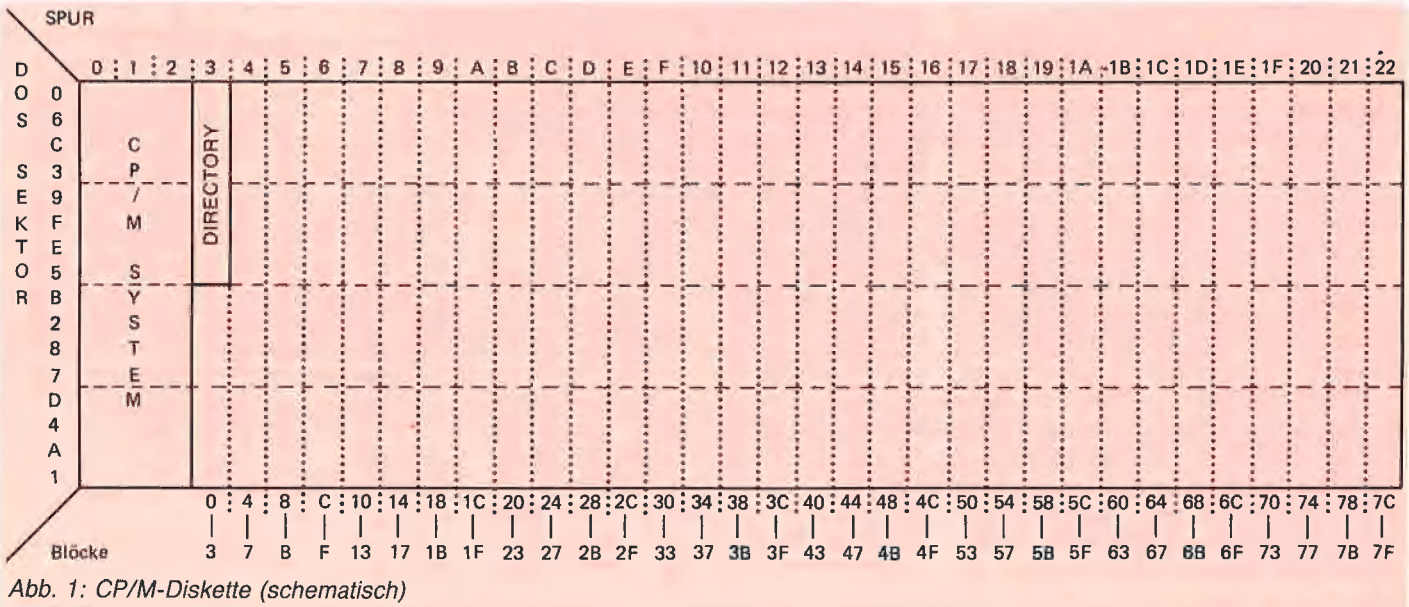


Abb. 1: CP/M-Diskette (schematisch)

Tabelle 1: Gleichungen

128 Byte	= 1	Record
2 Records	= 1	Sektor
8 Records	= 1	Block
1 Block	= 4	Sektoren
16 Sektoren	= 1	Spur
4 Blöcke	= 1	Spur

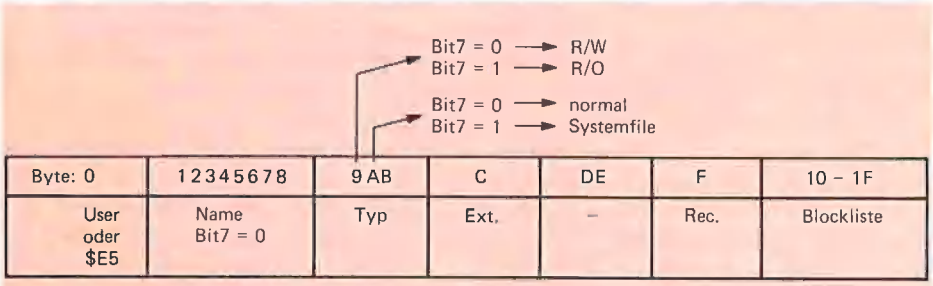
Tabelle 2: Skewing

Rel. Block	DOS-Sektoren
Block 0	0, 6, C, 3
Block 1	9, F, E, 5
Block 2	B, 2, 8, 7
Block 3	D, 4, A, 1

Tabelle 2 bedeutet dabei die Blockposition in einer Spur, so daß die Blöcke $00, $04, $08, ..., $7C z.B. alle den relativen Block 0 bilden (Rel. Block = Blocknummer MOD 4).

Das CP/M-Directory

Nachdem Sie nun wissen, wie die CP/M-Diskette organisiert ist, verstehen Sie auch leichter die Eintragungen im Directory.



Byte: 0	1 2 3 4 5 6 7 8	9 A B	C	D E	F	10 – 1 F
User oder $E5	Name Bit7 = 0	Typ	Ext.	–	Rec.	Blockliste

Abb. 2: Directory-Eintrag

Die beiden Blöcke 0 und 1 (2K) bestehen aus 64 Eintragungen von je 32 Bytes Länge, die alle nach demselben Schema aufgebaut sind und exakt die 2 Blöcke füllen. **Abbildung 2** zeigt den Aufbau eines Eintrags, **Abbildung 3** ist der erste Teil des „CP/M System Master Directory", wie ihn ein Sektoreditor listet. Byte $00 ist bei einem gelöschten File oder einem unbenutzten Eintrag gleich $E5, sonst entspricht es der „User-Number", die zwischen $00 und $0F liegt. Im allgemeinen finden Sie hier nur $00 wie in unserem Beispiel. Die Bytes $01 bis $08 enthalten den Filenamen mit gelöschtem Bit 7. Ist der Name kürzer als 8 Zeichen, wird der Eintrag mit $20 aufgefüllt. Byte $09 bis $0B enthalten die 3 Zeichen des Filetyps; gegebenenfalls sind auch sie mit $20 aufgefüllt. Normalerweise sind auch hier die Bit 7 gelöscht wie im Beispiel. Wäre das Bit 7 in Byte $09 gesetzt (hier also z.B. $C3 statt $43), hätte der File das R/O-Attribut, wäre also nur lesbar, aber nicht löschbar oder zu überschreiben. Ein gesetztes Bit 7 in Byte $0A kennzeichnet System-Files, die bei einem DIR-Befehl nicht gelistet werden. Die Bits werden mit dem STAT-Programm verändert. Das Byte $0C unterscheidet mehrfache Einträge für denselben File, was bei großen Files (z.B. GBASIC) notwendig ist. Mehr über diese Extension-Zahl später. Byte $0D und $0E sind immer gleich $00. Das Byte $0F bezeichnet die Zahl der Filerecords in diesem Eintrag, im Beispiel für FORMAT.COM $12 = 18.

Die restlichen 16 Bytes stellen eine Liste der vom File belegten Blöcke in der richtigen Reihenfolge dar, in unserem Beispiel die Blöcke $02, $03 und $04. Belegt ein File weniger als 16 Blöcke, so wird mit Nullen aufgefüllt. Sind es mehr als 16 Blöcke, wird ein zweiter Eintrag angelegt, der in der vorderen Hälfte ganz dem ersten entspricht, nur Byte $0C wird auf $01 hochgezählt und Byte $0F enthält natürlich die Records in diesem Eintrag (s. MBASIC oder GBASIC). Bis zu 8 Einträge (Extensions) sind bei einem File theoretisch möglich, der die ganze Diskette füllt. Außer bei Wörterbüchern bin ich solchen noch nicht begegnet. Sie könnten einen solchen File „am Stück" mangels Speicherplatz nicht mehr laden. Die mehrfachen Einträge für einen File liegen nur bei „neuen" Disketten sauber hintereinander, können aber nach vielen Lösch- und Schreiboperationen über das ganze Directory verteilt sein, wie ich durch unangenehme Erfahrung lernen mußte.

FORMAT.COM ist ein einfacher File. Er belegt $12 Records entsprechend 9 Sektoren, von denen die ersten vier in Block $02, die zweiten vier in Block $03 und der neunte in Block $04 liegen. Die restlichen 3 Sektoren in Block $04 sind leer. MBASIC.COM ist ein File mit 2 Extensions, denn er belegt $80 + $40 = 192 Records in den 24 Blöcken von $06 bis $1D. Jetzt wird es Ihnen sicherlich leicht fallen, auch die anderen Eintragungen zu entschlüsseln.

Wenn möglich, belegt CP/M aufeinanderfolgende Blöcke. An CONFIGIO.BAS sehen Sie, daß es hier auch Ausnahmen gibt.

Abbildung 3: Directory-Anfang

```
TRACK: 03    SECTOR: 00    CP/M    DRIVE 2

     00 01 02 03 04 05 06 07
     08 09 0A 0B 0C 0D 0E 0F      ASCII

00:  00 46 4F 52 4D 41 54 20     .FORMAT
     20 43 4F 4D 00 00 00 12      COM.....R
10:  02 03 04 00 00 00 00 00     BCD.....
     00 00 00 00 00 00 00 00     ........
20:  00 43 4F 50 59 20 20 20     .COPY
     20 43 4F 4D 00 00 00 08      COM.....H
30:  05 00 00 00 00 00 00 00     E.......
     00 00 00 00 00 00 00 00     ........
40:  00 4D 42 41 53 49 43 20     .MBASIC
     20 43 4F 4D 00 00 00 80      COM.....
50:  06 07 08 09 0A 0B 0C 0D     FGHIJKLM
     0E 0F 10 11 12 13 14 15     NOPQRSTU
60:  00 4D 42 41 53 49 43 20     .MBASIC
     20 43 4F 4D 01 00 00 40      COMA....§
70:  16 17 18 19 1A 1B 1C 1D     VWXYZÄÖÜ
     00 00 00 00 00 00 00 00     ........
80:  00 47 42 41 53 49 43 20     .GBASIC
     20 43 4F 4D 00 00 00 80      COM.....
90:  1F 20 21 22 23 24 25 26     _ !"#$%&
     27 28 29 2A 2B 2C 2D 2E     '()*+,-.
A0:  00 47 42 41 53 49 43 20     .GBASIC
     20 43 4F 4D 01 00 00 48      COMA....H
B0:  2F 30 31 32 33 34 35 36     /0123456
     37 00 00 00 00 00 00 00     7.......
C0:  00 43 4F 4E 46 49 47 49     .CONFIGI
     4F 42 41 53 00 00 00 3A     OBAS....:
D0:  1E 38 39 3A 3B 3C 3D 3E     ↑89:;<=>
     00 00 00 00 00 00 00 00     ........
E0:  00 50 49 50 20 20 20 20     .PIP
     20 43 4F 4D 00 00 00 3A      COM....:
F0:  40 41 42 43 44 45 46 47     §ABCDEFG
     00 00 00 00 00 00 00 00     ........

CTRL-B = BACK          FORWARD = CTRL-F
CTRL-S = SELECT          PRINT = CTRL-P
CTRL-X = EXIT             OPTIONS = ESC
```

Files entlöschen

Jetzt bin ich Ihnen nur noch die Auflösung schuldig, wie Sie gelöschte Files wiederbeleben. Mit einem Sektoreditor (z.B. ZAP) suchen Sie den Directory-Eintrag des gelöschten Files auf Spur 3. Haben Sie ihn gefunden, ersetzen Sie in Byte $00 die $E5 durch $00 und schreiben den Sektor zurück. Ihr File „lebt" wieder, wenn Sie nicht zwischenzeitlich seine Blöcke neu beschrieben haben. Bei einem File mit mehreren Extensions müssen Sie natürlich alle Eintragungen zurücksetzen.

Literatur:

Worth/Lechner: Beneath Apple DOS, Quality Software 1981

A.R. Miller: Programmieren mit CP/M, Sybex 1985

Apple-II-Monitor für den Mac

von Pit Capitain

Wie zu Wozniaks Zeiten

Als ich zum ersten Mal etwas vom Macintosh hörte, war ich ziemlich auf diesen Computer gespannt: MC68000-Prozessor, 128K RAM, gute Grafik, Maus, ähnliches Betriebssystem wie bei der LISA, all das versprach viel. Im Computergeschäft kam es dann auch zu den ersten Maus-Bewegungen, Bildverschiebungen und Malversuchen. Doch bald wollte ich als echter Computer-Freak dann auch einmal in den Mac hineinschauen, eigene Programme schreiben, das ROM untersuchen usw.

Später kam das Microsoft-Basic heraus, aber damit konnte ich auch nicht so richtig in den Mac „reinsehen", und Basic ist ohnehin nicht meine Lieblingssprache. Viel interessanter ist doch das Programmieren in Assembler! Immerhin gab es die Befehle „PEEK" und „POKE". Damit konnte ich mir wenigstens im Computergeschäft ein paar Bytes ausgeben lassen. Zu Hause wurde dann von Hand disassembliert, und siehe da: tatsächlich ein lauffähiges MC68000-Programm! Doch nachdem ich diese kurze Routine annähernd verstanden hatte, wollte ich wissen, wie sie weitergeht. Aber von Hand war dies alles viel zu umständlich. Deshalb sortierte ich den Befehlssatz des MC68000 und schrieb schließlich einen MC68000-Disassembler in Basic. Bald wollte ich dann selbst die ersten Maschinenprogramme schreiben und ausprobieren. Also wurde der Disassembler „aufgebohrt", und so ist dann schließlich dieses Monitorprogramm namens **MACMONITOR** entstanden. Es simuliert den Monitor, den ich vom Apple II her gewohnt war. Zudem sind einige Befehle funktionell erweitert worden. Mit diesem Programm kann man sich dann wieder auf sicherem, sprich gewohntem Boden bewegen und die ersten Gehversuche unternehmen. Es gibt aber auch einige Nachteile:

– Das Programm und das Basic-System belegen viel zuviel Platz im Speicher. Man ist immer irgendwie beengt. Und das bei 128K!

– Es gibt kaum Unterlagen über das ROM des Macintosh. Es ist erstens 64K lang und zweitens so komplex (Interrupt-Routinen, Treiber usw.), daß man mit dem Untersuchen nicht so richtig vorankommt. Wie wird beispielsweise ein Zeichen ausgegeben, wie liest man Zeichen von der Tastatur ein usw.? (Inzwischen weiß ich es; dazu an anderer Stelle mehr). Trotzdem ist der MACMONITOR bei mir das meistbenutzte Programm.

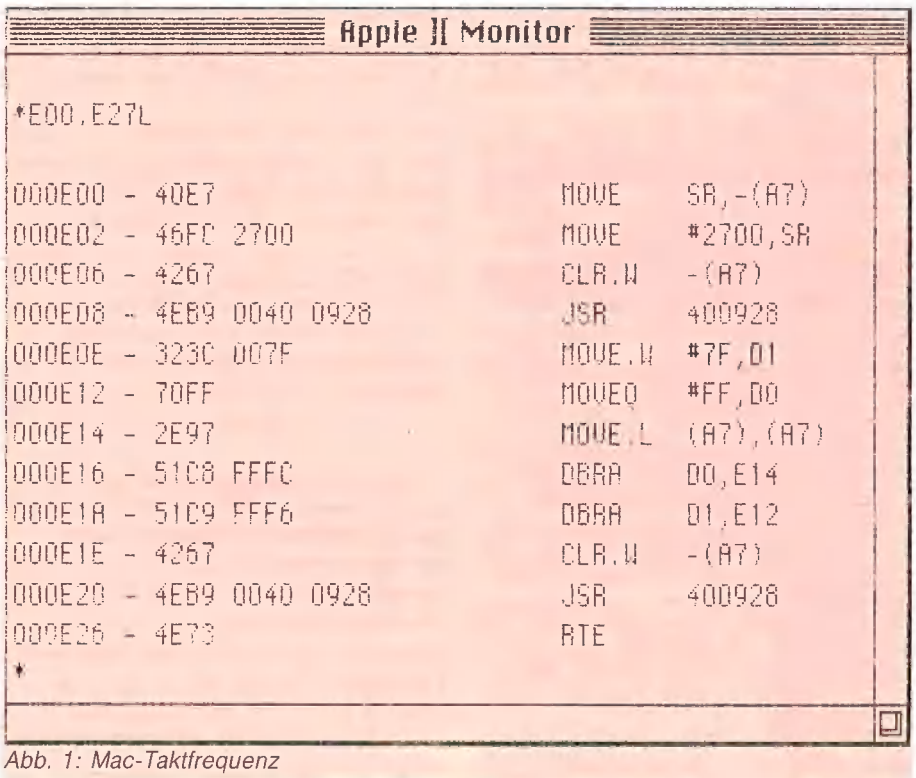


Abb. 1: Mac-Taktfrequenz

1. MACMONITOR-BEFEHLE

Zunächst beschreibe ich die implementierten Befehle:

Speicherinhalt ausgeben

{**adrs**} – gibt den Inhalt des Bytes mit der Adresse {adrs} aus.
{**adrs1**}.{**adrs2**} – zeigt den Inhalt von allen Bytes zwischen {adrs1} und {adrs2}.
RETURN-Taste – gibt den Inhalt von bis zu 16 weiteren Bytes aus, die nach der letztbenutzten Adresse folgen.

Speicherinhalt verändern

{**adrs**}:{**val**} {**val**} ... – speichert die angegebenen Bytes in aufeinanderfolgenden Adressen ab {adrs}.
:{**val**} {**val**} ... – speichert die Bytes ab der nächsten Adresse.

Verschieben und Vergleichen

{**dest**}<{**start**}.{**end**}**M** – kopiert den Inhalt des Bereichs {start}.{end} in den Bereich ab der Adresse {dest}.
{**dest**}<{**start**}.{**end**}**V** – vergleicht den Inhalt des Bereichs {start}.{end} mit dem des Bereichs ab der Adresse {dest}. Unterschiedliche Bytes werden ausgegeben.

Schreiben/Lesen auf/von Diskette

{**start**}.{**end**}**W** – schreibt den Inhalt des Bereichs {start}.{end} auf Diskette. Der Name der Datei wird vom Monitor erfragt.
{**start**}**R** – liest eine Datei von Diskette in den Bereich ab der Adresse {start}. Der Name der Datei wird vom Monitor erfragt. Es wird die gesamte Datei eingelesen.
R – wie der vorangehende Befehl, nur wird die Datei in den Bereich geladen, von dem aus sie gespeichert wurde. Dieser Bereich wird vom Monitor ausgegeben.

Starten und Auflisten eines Programms

{**adrs**}**G** – startet das Maschinenprogramm ab der Adresse {adrs}. In den Monitor muß mit einem „RTS"-Befehl zurückgekehrt werden.
{**adrs**}**L** – gibt die ersten 10 Befehle des Programms ab der Adresse {adrs} disassembliert aus. Nachfolgende „L"-Kommandos zeigen jeweils die nächsten 10 Befehle.
{**start**}.{**end**}**L** – gibt das Programm im Bereich {start}.{end} in disassemblierter Form aus.

Sonstige Befehle

N – löscht den Eingabemodus. Dies wird benötigt, um nach einem „:"-Kommando andere Monitorbefehle ausführen zu können, z.B.
E00:4E 75 NE00L
Q – beendet den Monitor und kehrt zum Basic zurück. (Beim Apple II: „Ctrl-C".)
{**val**}+{**val**} – addiert die beiden Werte und gibt das Ergebnis aus (bis zu 32 Bits).
{**val**}-{**val**} – subtrahiert den zweiten Wert vom ersten und gibt das Ergebnis aus (bis zu 32 Bits).
{**slot**}**P** – leitet die Ausgabe zu dem Ausgabegerät mit der Nummer {slot} um. Es gibt die folgenden Nummern:

0 – „SCRN:" (Bildschirm)
1 – „LPT1:" (Drucker)
2 – „COM1:" (serielle Schnittstelle)
3 – „CLIP:" (Zwischenspeicher)

Alle anderen Nummern aktivieren den Bildschirm. (Beim Apple II: „Ctrl-P".)

2. ARBEITSWEISE DES PROGRAMMS

Um das Programm im einzelnen verstehen zu können, sollte man vertraut sein
– mit dem Apple-II-Monitor,
– mit dem Befehlssatz des MC68000 und
– mit dem Basic vom Macintosh.
Der Apple-II-Monitor ist z.B. im „Apple II Reference Manual" aufgelistet und ausreichend erklärt. Alle Angaben zum MC68000 habe ich dem „MC68000 Benutzer-Handbuch" entnommen. Das Microsoft-Basic des Macintosh wird als Serie im „Peeker" vorgestellt.
Es folgen nunmehr Erläuterungen zu den einzelnen Abschnitten des Programms (mit den entsprechenden Zeilennummern):

100 – Initialisierung der Variablen
Im Speicher wird genügend Platz für alle Variablen geschaffen. Aus Platz- und Zeitgründen werden alle Variablen, die nicht mit „A" anfangen, als Integer-Variablen erklärt. Zwei Arrays werden definiert, der Bildschirm als Ausgabegerät bestimmt und die nicht-proportionale Schriftart „Monaco" ausgewählt. Zum Schluß wird noch der Bildschirm gelöscht. Anschließend ertönt der „$FF66-Beep". Dieser Teil entspricht dem Bereich $FF59 bis $FF68 des Apple-II-Monitors.

200 – Kommandozeile einlesen
Dies entspricht weitgehend der „GETLNZ"-Routine des Apple-II-Monitors ($FD67). Mit dem Befehl „Ctrl-X" kann man nun aber *nicht mehr* eine Zeile löschen.
Die gesamte Kommandozeile wird in CMD$ eingelesen. Als Endmarke wird ein „#" angehängt. Das „RETURN"-Kommando (Leerzeile) wird durch das Zeichen „&" ersetzt. Der Zustand des Monitors (MODE, $0031) in der Variablen MDE wird auf 0 gesetzt. Dieser Teil entspricht dem Bereich $FF69 bis $FF72 des Apple-II-Monitors.

300 – Nächste Zahl bestimmen
Entspricht der „GETNUM"-Routine beim Apple-II-Monitor ($FFA7). Es wird die nächste Hex-Zahl im Kommandostring gesucht. Diese Zahl steht in A2 ($003E/$003F). Falls MDE (Zustand) auf 0 gesetzt ist, wird die Zahl auch in A1 ($003C/$003D) und in A3 ($0040/$0041) kopiert. Die Variable DIG wird auf 1 gesetzt, falls mindestens eine Hex-Ziffer gefunden wurde, sonst bleibt DIG auf 0 gesetzt. Beim Apple-II-Monitor übernimmt diese Aufgabe das X-Register.
Das erste Zeichen, das keine Hex-Ziffer ist, wird in D$ übernommen. Beim Kommandostring CMD$ wird die Hex-Zahl und dieses letzte Zeichen gelöscht.
Falls das Ende der Kommandozeile erreicht wurde, wird das Leerzeichen-Programm ab Zeile 610 ausgeführt, und eine neue Zeile wird eingelesen. Dies entspricht der „CRMON"-Routine bei $FEF6.

400 – Kommando ausführen
Je nachdem, welches Zeichen in D$ steht, wird das dazugehörige Kommando ausgeführt. Dies entspricht dem Bereich $FF78 bis $FF89 beim Apple-II-Monitor.
Die Ausführung der Kommandos geschieht fast wie beim Apple. Die Kommandos „Leerzeichen", „RETURN" und „LIST" haben jeweils ein eigenes Unterprogramm, das weiter unten erklärt wird.
Zu den Befehlen R(ead und W(rite kann noch gesagt werden, daß die Maschinenprogramme wie ein „B-File" beim Apple-DOS abgespeichert werden, also zuerst die Anfangsadresse und dann die Länge des Bereichs, dahinter die Bytes selbst. Je zwei Bytes werden mit der „MKI$"-Funktion als String interpretiert und so auf die Datei geschrieben.

600 – Leerzeichen-Kommando
Dies entspricht der „BL1"-Routine beim Apple-II-Monitor ($FE00). Im einzelnen gilt folgende Zuordnung zwischen Basic-Zeilennummern und Monitor-Adressen:
610 – $FE04 (BLANK)
620 – $FE07 u. $FE0B (STOR)
630 – $FDC6 (XAMPM)
640 – $FDD1 (ADD)
650 – $FDD4 (Ausgabe des Ergebnisses)
660 – $FDAD (MOD8CHK)
670 – $FDB3 (XAM)
680 – $FDB6 (DATAOUT)
690 – $FDC0 (weiterzählen)

700 – Adresse in A1 ausgeben
Gibt die Adresse in A1 mit 6 Hex-Ziffern aus. Dies entspricht der „PRA1"-Routine beim Apple-II-Monitor ($FD92).

800 – Byte in WRD ausgeben
Gibt das Byte in der Variablen WRD mit 2 Hex-Ziffern aus. Dies entspricht der „PRBYTE"-Routine beim Apple-II-Monitor ($FDDA).

Das gesamte nachfolgende Programm ist nur für die Disassemblierung der MC68000-Befehle zuständig. Dabei wird der disassemblierte Befehl in der Variablen D$ schrittweise aufgebaut. Die Länge des Befehls (in Worten) steht in der Variablen LNG, die Adresse des Befehls (der Programmzähler) in der Variablen A1. Die Variable G gibt die Operationsgröße an: 1 bei Byte-, 2 bei Wort- und 4 bei Doppelwort-Befehlen.

900 – Sprungverteiler
Gibt den nächsten Befehl ab Adresse A1 disassembliert aus. Der Programmteil ab Zeile 900 dient dabei nur als Sprungverteiler: Die ersten 4 Bits des Befehls bestimmen, zu welcher Adresse gesprungen wird. Die Variable LNG, die die Länge des Befehls in Worten zählt, wird auf 0 gesetzt, ebenso die Variable Z, die angibt, mit wievielen Ziffern eine Hex-Zahl mindestens ausgegeben werden soll.

1000 – Nächstes Wort in WRD holen
Holt das nächste Wort ab der Adresse A1 (Programmzähler) in die Variable WRD. Dabei wird A1 um 2 weitergezählt. Die Variable LNG, die die Länge des augenblicklichen Befehls in Worten mitzählt, wird erhöht.

1100 – Nächstes Wort in O(0..5) und B(0..8) holen
Holt das nächste Wort ab der Adresse A1 (mit der Routine in Zeile 1000). Dieses Wort wird in Oktal- und Binärziffern aufgespalten. Die Oktalziffern stehen im Array O, die Binärziffern im Array B (die niederwertigsten Ziffern in O(0) bzw. B(0)). Diese Aufspaltung erleichtert später die Entschlüsselung der Befehlscodes.

1200 – Operand in A mit Z Ziffern in W$
Diese Routine wandelt den Operanden in A in eine Hex-Zahl um. Die Hex-Zahl wird in den String W$ übernommen. Falls sie weniger Ziffern hat, als in der Variablen Z gefordert wird, werden vorne Nullen hinzugefügt. Falls die Zahl mehr Ziffern hat, so wird sie aber *nicht* abgeschnitten.

1300 – Hole G Bytes in W$
Ab der Adresse A1 werden G Bytes gelesen. Diese Bytes werden als Hex-Zahl in W$ zurückgegeben. Dies dient dazu, Operanden darzustellen. Operanden, die 1 Byte lang sind (oder 3, 5, usw.), belegen im Speicher eine *gerade* Zahl von Bytes (Worteinteilung). Deshalb wird, falls z.B. G = 1 ist, nicht nur 1 Byte, sondern 2 Bytes geholt, von denen allerdings nur das zweite den Operanden enthält.

1400 – Größe der Operation in G
Bei den MC68000-Befehlen ist die Operationsgröße (Byte, Wort oder Doppelwort) sehr oft in den Bits 6 und 7 bzw. in der Oktalziffer 2 codiert. Diese Routine bestimmt nun in Abhängigkeit dieser Bits die Operationsgröße. Die Größe (1, 2 oder 4 Bytes) wird in der Variablen G gespeichert. Die Größenbezeichnung („.B", „.W" oder „.L") wird an den bisher erzeugten Befehlsstring in D$ angehängt.

1500 – Falscher Befehlscode
Diese Routine wird immer dann angesprungen, wenn ein ungültiger Befehlscode gefunden wurde. In diesem Fall wird die Disassemblierung abgebrochen. In der Variablen D$, die den bisher disassemblierten Befehl enthält, wird ein „???" gespeichert. Die Routine geht in den nächsten Programmteil über:

1600 – Zeile ausgeben
Der disassemblierte Befehl wird ausgegeben. In der Variablen A1 steht die Adresse des nächsten Befehls, in LNG die Länge des Befehls in Worten und in D$ der disassemblierte Befehl. Es wird zunächst die Adresse des Befehls, dann der Befehl in Hex-Darstellung und schließlich in disassemblierter Form ausgegeben.

1700 – Bestimme Registerliste für MOVEM
Bei dem Befehl „MOVEM" werden mehrere Register auf einmal transportiert. Welche der 16 Register transportiert werden, steht in einem besonderen Wort nach dem Befehl. Die Routine wandelt dieses Wort in lesbare Form um. Dabei wird die sogenannte „Registerliste" wie folgt ausgegeben:
<Axxx,Dxxx> bzw. <Dxxx,Axxx>,
wobei nach „A" die Nummern der Adreß- und nach „D" die Nummern der Datenregister stehen. Die Reihenfolge der Register richtet sich danach, mit welcher Adressierungsart die Register transportiert werden. Diese Registerliste wird an den String D$ angehängt.

1800 – Bedingungscode entschlüsseln.
In Abhängigkeit der Bits 8 bis 11 wird bestimmt, um welche Bedingung es sich handelt. Die Befehle „Bcc" und „DBcc" weisen ein paar Unregelmäßigkeiten bei den Bedingungen mit Code $0 und $1 auf. Die decodierte Bedingung wird in D$ zurückgegeben.

2000-5999 – Befehl decodieren
Die nächsten Routinen decodieren den MC68000-Befehl. Dies geschieht durch die Überprüfung einzelner Oktal- und Binärziffern sowie mit den bisher beschriebenen Unterprogrammen ab Zeile 900. Auf die einzelnen Befehle kann ich hier nicht eingehen. Wenn man die MC68000-Befehle in ihrer codierten Form sortiert, kommt man schließlich auf eine mehr oder weniger ähnliche Einteilung, wie sie hier in dem Programm vorgenommen wurde.

Noch ein Wort zu den „nicht implementierten" Befehlen, die in den ersten 4 Bits „1010" enthalten. Jedesmal, wenn der Prozessor auf solch einen Befehl trifft, wird ein „Trap" (Unterbrechung) ausgelöst. Der Prozessor springt zu einer besonderen Routine (Adresse s.u.), die dann je nach dem Rest des Befehls aus einer Tabelle eine Adresse im ROM holt und dorthin springt. Auf diese Weise werden fast alle Routinen im ROM des Macintosh aufgerufen. Man schiebt die Parameter auf den Stack, dann kommt der $Axxx-Befehl, und die Routine wird korrekt aufgerufen. Das geht zwar etwas langsamer als ein direkter Sprung, hat aber den Vorteil, daß bei einer (sicher noch öfter zu erwartenden) Änderung des Monitors nicht alle Adressen geändert werden müssen, sondern nur die Tabelle mit den Adressen.

6000 – Bestimme EA (Effektive Adresse)
Die vielfältigen Adressierungsarten des MC68000 sind fast alle sehr systematisch codiert. Deshalb benötigt man zur Decodierung nur diese Routine.
Meistens (bis auf die Zieladresse im „MOVE"-Befehl) steht die Adresse in den Oktalziffern 1 (Modus) und 0 (Register). Anhand dieser beiden Werte wird die vorliegende Adressierungsart entschlüsselt.

3. MAC-TAKTFREQUENZ

Ein kleines Maschinenprogramm soll als Beispiel angegeben werden, und zwar die Bestimmung der Taktfrequenz des Macintosh. Das Listing des Programms sieht man in **Abb. 1**. Es folgt eine kurze Erklärung der einzelnen Befehle:

E00 – Statusregister auf Stack retten
E02 – S-Flag setzen, Interruptmaske auf 7 setzen (keine Interrupts mehr zulässig)
E06 – 0 auf den Stack schreiben (Dauer des Tons)
E08 – Unterprogramm im ROM zum Piepsen
E0E – Zähler der äußeren Schleife (D1) auf $7F (128 Durchläufe)
E12 – Zähler der inneren Schleife (D0) auf $FFFF (65536 Durchläufe)
E14 – Dummy-Operation (macht nichts, aber benötigt Zeit)
E16 – inneren Zähler erniedrigen und wiederholen, falls Zähler nicht 0 war
E1A – äußeren Zähler erniedrigen und wiederholen, falls Zähler nicht 0 war
E1E – 0 auf den Stack schreiben (Dauer des Tons)
E20 – Unterprogramm im ROM zum Piepsen
E26 – altes Statusregister vom Stack holen, Rücksprung in den Monitor

Es handelt sich also um zwei ineinandergeschachtelte FOR-NEXT-Schleifen, die in großer Zahl durchlaufen werden. Vor und nach den Schleifen wird ein Ton ausgegeben, damit man die Zeit zwischen den Tönen stoppen kann.
Wenn man aus dem MC68000 Benutzerhandbuch die Zeiten für die einzelnen Befehle heraussucht und alles zusammenzählt (mit dem Unterprogramm zur Tonerzeugung!), so erhält man folgende Formel:
418 + 18 * D1 + 32 * D1 * D0 Takte.
Soviele Takte vergehen also vom Ausschalten des ersten Tons bis zum Einschalten des zweiten Tons. Die Variablen D0 und D1 sind die Werte, mit denen die Zählvariablen geladen werden. (Für die ganz genauen Leser: die Anfangswerte + 1, da die DBRA-Schleifen so lange durchlaufen werden, bis der Zähler -1 ($FFFF) erreicht hat.)
Mit den Werten aus **Abb. 1** (D0 = 65536, D1 = 128) ergibt sich die Zahl von 268438178 Takten. Die Zeit, die der Mac dafür benötigt, beträgt ca. 48,6s. Daraus ergibt sich eine Taktfrequenz von ca. 5,5 MHz beim Zugriff auf das RAM, also deutlich weniger als 8 MHz! (Beim Ausrechnen dieser Zahlen leistet der eingebaute Taschenrechner des Macintosh gute Dienste.)
Wer genügend Zeit und Geduld hat, kann den Anfangswert der äußeren Schleife in $E10/E11 vergrößern, um dadurch längere Zeiten und somit genauere Messungen zu erzielen.

Zum Abschluß noch ein paar nützliche Adressen:

$000400–0007FF – Tabelle mit den 1010-Trap-Adressen
$000E00–000FFF – (meistens) freier Speicherplatz (im System-Heap)
$01A700 – Anfang des Bildschirmspeichers
$400000 – Anfang des 64K-ROMs
$40002A – Kaltstart
$401018 –Adresse für Decodierung der 1010-Traps

MACMONITOR

```
 1Ø REM  Simulation des Apple-II-Monitors
 11 REM  Version 1.1
 12 REM  von Pit Capitain
1ØØ REM
1Ø1 REM --- Initialisierung der Variablen ----
1Ø2 REM
11Ø CLEAR, 29ØØØ, 1Ø24
12Ø DEFINT B-Z
13Ø DIM O(5), B(8)
14Ø OPEN "SCRN:" FOR OUTPUT AS #1
15Ø CALL TEXTFONT (4) : CALL TEXTSIZE (12)
16Ø CLS : WIDTH 7Ø : BEEP
2ØØ REM
2Ø1 REM -- Kommandozeile einlesen --
2Ø2 REM
21Ø PRINT#1, : LINE INPUT "*"; CMD$
22Ø IF CMD$="" THEN CMD$="&*" ELSE CMD$ = CMD$ + "*"
23Ø MDE=Ø
3ØØ REM
3Ø1 REM -- Nächste Zahl bestimmen --
3Ø2 REM
31Ø A2=Ø : DIG=Ø
32Ø D$=LEFT$(CMD$,1) : CMD$=MID$(CMD$,2)
33Ø IF D$="*" THEN I=MDE : GOSUB 61Ø : GOTO 21Ø
34Ø IF "a"<=D$ AND D$<="z" THEN D$=CHR$(ASC(D$)-32)
35Ø I=ASC(D$) XOR 48 : IF I<1Ø THEN 37Ø
    ELSE IF I<64 THEN 41Ø
36Ø I=I-1Ø3 : IF I<1Ø OR 15<I THEN 41Ø
37Ø A2=A2*16+I : DIG=1 : IF MDE=Ø THEN A1=A2 : A3=A2
38Ø GOTO 32Ø
4ØØ REM
4Ø1 REM -- Kommando ausführen --
4Ø2 REM
41Ø I=MDE : MDE=Ø
42Ø IF D$=" " THEN GOSUB 61Ø : GOTO 31Ø
43Ø IF D$="&" THEN A2=15+16*INT(A1/16) : GOSUB 66Ø :
    GOTO 21Ø
44Ø IF D$="+" OR D$="-" OR D$="." OR D$=":"
    THEN MDE=ASC(D$) : GOTO 31Ø
45Ø IF D$="<" THEN A4=A2 : GOTO 31Ø
46Ø IF D$="G" THEN CALL A1 : GOTO 31Ø
47Ø IF D$<>"M" THEN 48Ø
471 WRD=PEEK(A1) : POKE A4, WRD
472 A4=A4+1 : A1=A1+1 : IF A1<=A2 THEN 471 ELSE 31Ø
48Ø IF D$="N" THEN 31Ø
49Ø IF D$="Q" THEN CLOSE : END
5ØØ IF D$<>"V" THEN 51Ø
5Ø1 IF PEEK(A1) = PEEK(A4) THEN 5Ø4
5Ø2 GOSUB 71Ø : WRD=PEEK(A1) : GOSUB 81Ø
5Ø3 PRINT#1, " ("; : WRD=PEEK(A4) : GOSUB 81Ø :
    PRINT#1, ")";
5Ø4 A4=A4+1 : A1=A1+1 : IF A1<=A2 THEN 5Ø1 ELSE 31Ø
51Ø IF D$<>"L" THEN 52Ø
511 IF I<>46 THEN A2=-1Ø
512 GOSUB 91Ø : IF A1<=A2 THEN 512
513 IF A2<Ø THEN A2=A2+1 : IF A2<Ø THEN 512
514 GOTO 31Ø
52Ø IF D$<>"W" THEN 53Ø
521 IF A2<A1 THEN A2=A1
522 LINE INPUT "Name : "; D$
523 OPEN D$ FOR OUTPUT AS #2
524 PRINT#2, MKD$(A1); MKD$(A2-A1);
525 GOSUB 1Ø1Ø : PRINT#2, MKI$(WRD);
526 IF A1<=A2 THEN 525 ELSE CLOSE #2 : GOTO 31Ø
53Ø IF D$<>"R" THEN 54Ø
531 LINE INPUT "Name : "; D$ : OPEN D$ FOR INPUT AS #2
532 A2=CVD(INPUT$(8,#2)) : IF A1=Ø OR DIG=Ø THEN A1=A2
533 A=A1 : Z=6 : GOSUB 121Ø : PRINT W$ ".";
534 A2=CVD(INPUT$(8,#2)) : FOR I=Ø TO A2 STEP 2
535   WRD=CVI(INPUT$(2,#2)) : POKE A1, (WRD AND -256)\256
536   POKE A1+1, WRD MOD 256 : A1=A1+2
537 NEXT : CLOSE #2
538 A=A1-1 : Z=6 : GOSUB 121Ø : PRINT W$ : GOTO 31Ø
54Ø IF D$<>"P" THEN 59Ø
541 CLOSE #1 : ON A2 GOTO 543,544,545
542 D$="SCRN:" : GOTO 546
543 D$="LPT1:" : GOTO 546
544 D$="COM1:" : GOTO 546
545 D$="CLIP:"
546 OPEN D$ FOR OUTPUT AS #1 : GOTO 31Ø
59Ø BEEP : GOTO 21Ø
6ØØ REM
6Ø1 REM -- Leerzeichen-Kommando --
6Ø2 REM
61Ø IF DIG=Ø THEN MDE=I : RETURN
62Ø IF I=58 THEN WRD=A2-256*INT(A2/256) : POKE A3,WRD :
    A3=A3+1 : MDE=I : RETURN
63Ø IF I MOD 2=Ø THEN 67Ø ELSE IF I=45 THEN A2=-A2
64Ø A=A1+A2 : IF A<Ø THEN A=A+2↑32
65Ø Z=4 : GOSUB 121Ø : PRINT#1, "=" W$; : RETURN
66Ø IF A1>16*INT(A1/16) THEN 68Ø
67Ø GOSUB 71Ø
68Ø WRD=PEEK(A1) : GOSUB 81Ø : PRINT#1, " ";
69Ø A1=A1+1 : IF A1<=A2 THEN 66Ø ELSE RETURN
7ØØ REM
7Ø1 REM -- Adresse in A1 ausgeben --
7Ø2 REM
71Ø PRINT#1, : A=A1 : Z=6 : GOSUB 121Ø :
    PRINT#1, W$ " - "; : RETURN
8ØØ REM
8Ø1 REM -- Byte in WRD ausgeben --
8Ø2 REM
81Ø A=WRD : Z=2 : GOSUB 121Ø : PRINT#1, W$; : RETURN
9ØØ REM
9Ø1 REM -- Sprungverteiler --
9Ø2 REM
91Ø LNG=Ø : Z=Ø : GOSUB 111Ø : IF O(5) THEN 93Ø
92Ø ON O(4)+1 GOTO 2Ø1Ø,241Ø,251Ø,261Ø,271Ø,391Ø,411Ø,421Ø
93Ø ON O(4)+1 GOTO 431Ø,461Ø,481Ø,491Ø,5Ø1Ø,531Ø,541Ø,151Ø
1ØØØ REM
1ØØ1 REM -- Nächstes Wort in WRD holen --
1ØØ2 REM
1Ø1Ø WRD=PEEK(A1) : IF WRD>127 THEN WRD=WRD-256
1Ø2Ø WRD=WRD*256+PEEK(A1+1)
1Ø3Ø A1=A1+2 : LNG=LNG+1 : RETURN
11ØØ REM
11Ø1 REM -- Nächstes Wort in O(Ø..5) und B(Ø..8) holen --
11Ø2 REM
```

```
1110 GOSUB 1010
1120 FOR I=0 TO 5
1130   O(I)=WRD MOD 8 : IF I>2 THEN WRD=WRD\8 : GOTO 1170
1140   FOR DIG=0 TO 2
1150     B(DIG+3*I)=WRD MOD 2 : WRD=WRD\2
1160   NEXT DIG
1170 NEXT I
1180 RETURN
1200 REM
1201 REM -- Operand in A mit Z Ziffern in W$ --
1202 REM
1210 W$="" : IF A<0 THEN A=A+2↑16
1220 WRD=A-256*INT(A/256) : A=INT(A/256)
1230 W$=HEX$(WRD)+W$
1240 IF WRD<16 AND A>0 THEN W$="0"+W$
1250 IF A>0 THEN 1220
1260 IF LEN(W$)<Z THEN W$=STRING$(Z-LEN(W$),"0")+W$
1270 RETURN
1300 REM
1301 REM -- Hole G Bytes in W$ --
1302 REM
1310 A=0 : DIG=G
1320 GOSUB 1010 : IF DIG MOD 2=1 THEN WRD=WRD MOD 256 :
     DIG=DIG+1
1330 IF WRD<0 THEN A=A+1
1340 A=A*2↑16+WRD : DIG=DIG-2 : IF DIG>0 THEN 1320
     ELSE 1210
1400 REM
1401 REM -- Grösse der Operation in G --
1402 REM
1410 DIG=O(2)
1420 ON DIG GOTO 1440,1450,1460
1430 D$=D$+".B " : G=1 : RETURN
1440 D$=D$+".W " : G=2 : RETURN
1450 D$=D$+".L " : G=4 : RETURN
1460 D$=D$+".? " : RETURN
1500 REM
1501 REM -- Falscher Befehlscode --
1502 REM
1510 D$="???"
1600 REM
1601 REM -- Zeile ausgeben --
1602 REM
1610 A1=A1-2*LNG : GOSUB 710
1620 FOR DIG=LNG TO 1 STEP -1
1630   GOSUB 1010 : A=WRD : Z=4 : GOSUB 1210
1640   PRINT#1, W$ " ";
1650 NEXT
1660 I=INSTR(D$," ") : IF I=0 THEN PRINT#1, TAB(36) D$; :
     RETURN
1670 PRINT#1, TAB(36) LEFT$(D$,I-1) TAB(44)
     MID$(D$,I+1); : RETURN
1700 REM
1701 REM -- Bestimme Registerliste für MOVEM --
1702 REM
1710 DIG=0 : D$=D$+"<" : IF O(1)=4 THEN 1730
1720 FOR I=0 TO 15
1721   IF WRD MOD 2=0 THEN 1726 ELSE IF I<DIG THEN 1725
1722   IF I<8 THEN D$=D$+"D" : DIG=8 : GOTO 1725
1723   IF DIG>0 THEN D$=D$+","
1724   D$=D$+"A" : DIG=16
1725   D$=D$+HEX$(I MOD 8)
1726   WRD=WRD\2
1727 NEXT : GOTO 1740
1730 FOR I=0 TO 15
1731   IF WRD MOD 2=0 THEN 1736 ELSE IF I<DIG THEN 1735
1732   IF I<8 THEN D$=D$+"A" : DIG=8 : GOTO 1735
1733   IF DIG>0 THEN D$=D$+","
1734   D$=D$+"D" : DIG=16
1735   D$=D$+HEX$(7-I MOD 8)
1736   WRD=WRD\2
1737 NEXT
1740 D$=D$+">" : RETURN
1800 REM
1801 REM -- Bedingungscode entschlüsseln --
1802 REM
1810 DIG=B(8)+2*O(3)
1820 ON DIG+1 GOTO 1840,1842,1844,1845,1846,1847,1848,1849
1830 ON DIG-7 GOTO 1850,1851,1852,1853,1854,1855,1856,1857
1840 IF O(4)=6 THEN D$="RA" ELSE D$="T"
1841 RETURN
1842 IF O(4)=6 THEN D$="SR" ELSE IF O(1)=1 THEN D$="RA"
     ELSE D$="F"
1843 RETURN
1844 D$="HI" : RETURN
1845 D$="LS" : RETURN
1846 D$="CC" : RETURN
1847 D$="CS" : RETURN
1848 D$="NE" : RETURN
1849 D$="EQ" : RETURN
1850 D$="VC" : RETURN
1851 D$="VS" : RETURN
1852 D$="PL" : RETURN
1853 D$="MI" : RETURN
1854 D$="GE" : RETURN
1855 D$="LT" : RETURN
1856 D$="GT" : RETURN
1857 D$="LE" : RETURN
2000 REM
2001 REM -- $0xxx : unmittelbare Befehle --
2002 REM
2010 IF B(8)=1 THEN 2110
2020 ON O(3) GOTO 2022,2023,2024,2110,2025,2026,1510
2021 D$ = "ORI" : GOTO 2030
2022 D$ = "ANDI" : GOTO 2030
2023 D$ = "SUBI" : GOTO 2030
2024 D$ = "ADDI" : GOTO 2030
2025 D$ = "EORI" : GOTO 2030
2026 D$ = "CMPI"
2030 GOSUB 1410
2040 GOSUB 1310 : D$=D$+"#"+W$+"," : GOSUB 6010 :
     GOTO 1610
2100 REM
2101 REM -- $0xxx : Bit-Manipulation (statisch) --
2102 REM
2110 IF B(8)=1 OR O(3)<>4 THEN 2210
2120 G=1 : ON O(2) GOTO 2140,2150,2160
2130 D$="BTST " : GOTO 2040
2140 D$="BCHG " : GOTO 2040
2150 D$="BCLR " : GOTO 2040
2160 D$="BSET " : GOTO 2040
2200 REM
2201 REM -- $0xxx : MOVEP --
2202 REM
2210 IF O(1) <> 1 THEN 2310
2220 D$="MOVEP" : DIG=1+B(6) : GOSUB 1420
2230 G=2 : GOSUB 1310 : ON B(7) GOTO 2250
2240 D$=D$+W$+"(A"+HEX$(O(0))+"),D"+HEX$(O(3)) : GOTO 1610
2250 D$=D$+"D"+HEX$(O(3))+","+W$+"(A"+HEX$(O(0))+")" :
     GOTO 1610
2300 REM
2301 REM -- $0xxx : Bit-Manipulation (dynamisch) --
2302 REM
2310 ON O(2) MOD 4 GOTO 2330,2340,2350
2320 D$="BTST" : GOTO 2360
2330 D$="BCHG" : GOTO 2360
2340 D$="BCLR" : GOTO 2360
2350 D$="BSET"
2360 D$=D$+" D"+HEX$(O(3))+"," : GOSUB 6010 : GOTO 1610
2400 REM
2401 REM -- $1xxx : MOVE.B --
2402 REM
2410 D$="MOVE.B" : G=1
2420 D$=D$+" " : GOSUB 6010 : D$=D$+","
2430 MDE=O(2) : REG=O(3) : GOSUB 6020 : GOTO 1610
2500 REM
2501 REM -- $2xxx : MOVE.L --
2502 REM
2510 G=4 : IF O(2)=1 THEN D$="MOVEA.L" ELSE D$="MOVE.L"
2520 GOTO 2420
2600 REM
2601 REM -- $3xxx : MOVE.W --
2602 REM
2610 G=2 : IF O(2)=1 THEN D$="MOVEA.W" ELSE D$="MOVE.W"
2620 GOTO 2420
2700 REM
2701 REM -- $4xxx : LEA --
2702 REM
2710 IF O(2)<7 THEN 2810
2720 D$="LEA " : G=4 : GOSUB 6010 :
     D$=D$+",A"+HEX$(O(3)) : GOTO 1610
2800 REM
2801 REM -- $4xxx : CHK --
2802 REM
2810 IF O(2)<6 THEN 2910
2820 D$="CHK" : G=2 : GOSUB 6010 :
     D$=D$+" , D"+HEX$(O(3)) : GOTO 1610
2900 REM
2901 REM -- $4xxx : NEGX, CLR, NEG, NOT --
2902 REM
2910 IF B(8)=1 THEN 1510 ELSE IF O(3)>3 THEN 3110
     ELSE IF O(2)=3 THEN 3010
2920 ON O(3) GOTO 2940,2950,2960
2930 D$="NEGX" : GOTO 2970
```

```
2940 D$="CLR" : GOTO 2970
2950 D$="NEG" : GOTO 2970
2960 D$="NOT"
2970 GOSUB 1410 : GOSUB 6010 : GOTO 1610
3000 REM
3001 REM -- $4xxx : MOVE mit SR bzw. CCR --
3002 REM
3010 ON O(3) GOTO 1510,3030,3040
3020 D$="MOVE SR," : G=2 : GOSUB 6010 : GOTO 1610
3030 D$="MOVE " : G=2 : GOSUB 6010 : D$=D$+",CCR" :
     GOTO 1610
3040 D$="MOVE " : G=2 : GOSUB 6010 : D$=D$+",SR" :
     GOTO 1610
3100 REM
3101 REM -- $4xxx : SWAP, EXT --
3102 REM
3110 ON O(3)-4 GOTO 3310, 3410, 3510
3120 IF O(1)>0 OR O(2)=0 THEN 3210
3130 ON O(2) GOTO 3140,3150,3160
3140 D$="SWAP" : GOTO 3170
3150 D$="EXT.W" : GOTO 3170
3160 D$="EXT.L"
3170 D$=D$+" D"+HEX$(O(0)) : GOTO 1610
3200 REM
3201 REM -- $4xxx : NBCD, PEA, MOVEM (Reg)-->(EA) --
3202 REM
3210 ON O(2) GOTO 3230,3240,3240
3220 D$="NBCD " : G=1 : GOTO 3250
3230 D$="PEA " : G=4 : GOTO 3250
3240 D$="MOVEM" : DIG=1+B(6) : GOSUB 1420
3241 GOSUB 1010 : GOSUB 1710 : D$=D$+","
3250 GOSUB 6010 : GOTO 1610
3300 REM
3301 REM -- $4xxx : TST, TAS --
3302 REM
3310 ON O(2) GOTO 3320,3320,3330
3320 D$="TST" : GOSUB 1410 : GOTO 3250
3330 D$="TAS " : G=1 : GOTO 3250
3400 REM
3401 REM -- $4xxx : MOVEM (EA)-->(Reg) --
3402 REM
3410 IF B(7)=0 THEN 1510 ELSE GOSUB 1010 : I=WRD
3420 D$ = "MOVEM" : DIG=1+B(6) : GOSUB 1420 : GOSUB 6010
3430 D$=D$+"," : WRD=I : GOSUB 1710 : GOTO 1610
3500 REM
3501 REM -- $4xxx : JSR, JMP --
3502 REM
3510 IF B(7)=0 THEN 3610
3520 G=4 : IF B(6) THEN D$="JMP " ELSE D$="JSR "
3530 GOSUB 6010 : GOTO 1610
3600 REM
3601 REM -- $4xxx : RESET, NOP, STOP, RTE, RTS, TRAPV,
     RTR --
3602 REM
3610 IF B(6)=0 OR O(1)=7 THEN 1510 ELSE IF O(1)<6
     THEN 3710
3620 ON O(0) GOTO 3622,3623,3624,1510,3625,3626,3627
3621 D$="RESET" : GOTO 1610
3622 D$="NOP" : GOTO 1610
3623 G=2 : GOSUB 1310 : D$="STOP #"+W$ : GOTO 1610
3624 D$="RTE" : GOTO 1610
3625 D$="RTS" : GOTO 1610
3626 D$="TRAPV" : GOTO 1610
3627 D$="RTR" : GOTO 1610
3700 REM
3701 REM -- $4xxx : LINK, UNLK, MOVE mit USP --
3702 REM
3710 IF O(1)<2 THEN 3810
3720 ON O(1)-2 GOTO 3740,3750,3760
3730 G=2 : GOSUB 1310 : D$="LINK A"+HEX$(O(0))+",#"+W$ :
     GOTO 1610
3740 D$="UNLK A"+HEX$(O(0)) : GOTO 1610
3750 D$="MOVE A"+HEX$(O(0))+",USP" : GOTO 1610
3760 D$="MOVE USP,A"+HEX$(O(0)) : GOTO 1610
3800 REM
3801 REM -- $4xxx : TRAP --
3802 REM
3810 A=O(0)+8*B(3) : GOSUB 1210 : D$="TRAP #"+W$ :
     GOTO 1610
3900 REM
3901 REM -- $5xxx : ADDQ, SUBQ --
3902 REM
3910 IF O(2) MOD 4=3 THEN 4010
3920 IF B(8) THEN D$="SUBQ" ELSE D$="ADDQ"
3930 DIG=O(2) MOD 4 : GOSUB 1420
3940 DIG=O(3) : IF DIG=0 THEN DIG=8
3950 D$=D$+"#"+HEX$(DIG)+","
```

```
3960 GOSUB 6010 : GOTO 1610
4000 REM
4001 REM -- $5xxx : Scc, DBcc --
4002 REM
4010 GOSUB 1810 : IF O(1)<>1 THEN D$="S"+D$+" " : G=1 :
     GOTO 3960
4020 D$="DB"+D$+" D"+HEX$(O(0))+","
4030 GOSUB 1010 : A=A1-2+WRD : GOSUB 1210 : D$=D$+W$ :
     GOTO 1610
4100 REM
4101 REM -- $6xxx : Bcc --
4102 REM
4110 A=O(0)+8*O(1)+64*(O(2) MOD 4) : IF A>127 THEN A=A-256
4120 IF A=0 THEN GOSUB 1010 : A=A1-2+WRD ELSE A=A1+A
4130 GOSUB 1210 : GOSUB 1810 : D$="B"+D$+" "+W$ :
     GOTO 1610
4200 REM
4201 REM -- $7xxx : MOVEQ --
4202 REM
4210 IF B(8)=1 THEN 1510 ELSE D$="MOVEQ #"
4220 A=O(0)+8*O(1)+64*(O(2) MOD 4) : GOSUB 1210
4230 D$=D$+W$+",D"+HEX$(O(3)) : GOTO 1610
4300 REM
4301 REM -- $8xxx : DIVx --
4302 REM
4310 IF O(2) MOD 4<3 THEN 4410
4320 G=2 : IF B(8) THEN D$="DIVS " ELSE D$="DIVU "
4330 GOSUB 6010 : D$=D$+",D"+HEX$(O(3)) : GOTO 1610
4400 REM
4401 REM -- $8xxx : SBCD --
4402 REM
4410 IF O(2)<>4 OR O(1)>1 THEN 4510 ELSE D$="SBCD "
4420 IF B(3) THEN D$=D$+"-(A"+HEX$(O(0))+"),-(A"
     +HEX$(O(3))+")" : GOTO 1610
4430 D$=D$+"D"+HEX$(O(0))+",D"+HEX$(O(3)) : GOTO 1610
4500 REM
4501 REM -- $8xxx : OR --
4502 REM
4510 D$="OR"
4520 DIG=O(2) MOD 4 : GOSUB 1420
4530 IF B(8)=0 THEN 4330
4540 D$=D$+"D"+HEX$(O(3))+"," : GOSUB 6010 : GOTO 1610
4600 REM
4601 REM -- $9xxx : SUBA --
4602 REM
4610 IF O(2) MOD 4<3 THEN 4710 ELSE D$="SUBA"
4620 DIG=1+B(8) : GOSUB 1420 : GOSUB 6010
4630 D$=D$+",A"+HEX$(O(3)) : GOTO 1610
4700 REM
4701 REM -- $9xxx : SUBX, SUB --
4702 REM
4710 IF B(8)=0 OR O(1)>1 THEN D$="SUB" : GOTO 4520
4720 D$="SUBX" : DIG=O(2) MOD 4 : GOSUB 1420 : GOTO 4420
4800 REM
4801 REM -- $Axxx : "Linie 1010 Emulator" --
4802 REM
4810 A1=A1-2: LNG=LNG-1 : Z=4 : G=2 : GOSUB 1310
4820 D$="CALL ROM."+W$ : GOTO 1610
4900 REM
4901 REM -- $Bxxx : CMPA, CMP, EOR, CMPM --
4902 REM
4910 IF O(2) MOD 4=3 THEN D$="CMPA" : GOTO 4620
4920 IF B(8)=0 THEN D$="CMP" : GOSUB 1410 : GOTO 4330
4930 IF O(1)<>1 THEN D$="EOR" : GOTO 4520
4940 D$="CMPM" : DIG=O(2) MOD 4 : GOSUB 1420
4950 D$=D$+"(A"+HEX$(O(0))+")+,(A"+HEX$(O(3))+")+" :
     GOTO 1610
5000 REM
5001 REM -- $Cxxx : MULx --
5002 REM
5010 IF O(2) MOD 4<3 THEN 5110
5020 IF B(8) THEN D$="MULS " ELSE D$="MULU " :
5030 G=2 : GOTO 4330
5100 REM
5101 REM -- $Cxxx : ABCD --
5102 REM
5110 IF O(2)<>4 OR O(1)>1 THEN 5210
5120 D$="ABCD " : GOTO 4420
5200 REM
5201 REM -- $Cxxx : EXG, AND --
5202 REM
5210 IF O(2)<>5 OR O(1)>1 THEN 5250
5220 D$="EXG " : ON B(3) GOTO 5240
5230 D$=D$+"D"+HEX$(O(3))+",D"+HEX$(O(0)) : GOTO 1610
5240 D$=D$+"A"+HEX$(O(3))+",A"+HEX$(O(0)) : GOTO 1610
5250 IF O(2)<>6 OR O(1)<>1 THEN D$="AND" : GOTO 4520
5260 D$="EXG D"+HEX$(O(3))+",A"+HEX$(O(0)) : GOTO 1610
```

Fortsetzung Seite 50

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

BUCH-SHOP

Betriebssystem CP/M

Vom Monitorprogramm zum Mehrbenutzersystem.
Von Jürgen Plate.
1984, 351 Seiten, 30 Abb., 3 Tab., geb., DM 56,–

Das Buch beschreibt ausführlich die Kommandos, ihre genaue Syntax und die einzelnen Teilprogramme von CP/M wie BIOS (systemspezifischer Teil), ED (Editor), ASM (Asembler, inklusive einer Beschreibung des 8080-Befehlssatzes), SYSGEN und STAT.
Der Beschreibung von CP/M ist das Listing eines komfortablen Monitorprogramms für Z-80-Computer vorangestellt, das eine elementare Programmierung auf Maschinenebene erlaubt, solange man CP/M noch nicht geladen hat. Das kann z. B. zur Fehlersuche sehr nützlich sein. Am Schluß des Buches findet sich auch eine Kurzbeschreibung der Multitasking-/Multiuser-Betriebssysteme.

Das Buch zum Apple II

von Erich Esders
1985, 210 S., 119 Abb., geb, DM 54,–

Wenn hier vom Apple II gesprochen wird, so gilt das auch für den IIplus, den Ileuroplus und die IIe-Versionen sowie für den ganzen „Apple-Nachbau". Das Buch ist ein Wegweiser durch diesen Rechner, um mit ihm schneller und effektiver zu arbeiten. Es geht hier weniger um das elementare Programmieren des Rechners, sondern um Assemblerprogramme, die extensiv Monitor-ROM-Subroutinen benutzen. Diese hat der Autor nach Sachgebieten geordnet, z. B. Mathematik, Graphik, String-Bearbeitung + Disassembler-Listings und diese wiederum mit Erklärungen und Applikationen komplettiert. Eine ausreichende Dokumentation ist dabei immer gewährleistet. Sie geht schrittweise vor, von der Aufgabenstellung über die Programmentwicklung bis zum lauffähigen Maschinenprogramm. Die angebotenen Beispiele sind ausbaufähig und lassen der eigenen Kreativität reichlichen Spielraum. Viele neuartige Tips und Tricks wird auch der beschlagene Apple-Benutzer begrüßen.
Aus dem Inhalt:
Der Mikroprozessor des APPLE II. Der APPLE II und seine Speicheraufteilung. APPLESOFT und seine Arbeitsspeicher-Bereiche. Der MICROSOFT-Basic-Interpreter: Die Zeichen-Lese-Routine. Interpretierer und Lokalisierer. Handler-Routinen. BASIC/Maschinensprache-Interfaces. DISAS-Generator. Unterprogramme im APPLESOFT-Basic-Interpreter: Softschalter und -Flags. Ausdrucks-Interpretierer. Low-Resolution-Graphik. Fehler-Behandlung. Applikationen: Arithmetik-Demonstration „FP-CALC". Hex-Dumps der Applikationen. BASIC-Monitor BASMON/D: Vorstellung der neuen Kommandos. Das Programm „BASMON/D". Implementierung und Laufbeispiele. BASIC-Interpreter-Vergleich APPLE II – Commodore 64: Arithmetik-Demonstration „FP-CALC/64". Listen: Die Token des APPLESOFT-Basic.

Apple II ROM Listing

von Matthias Buck
1984, 116 S., Kart., DM 59,–

Das deutsche Apple-II-ROM-Listing ist da!
Einleitung zum prinzipiellen Ablauf des Applesoftinterpreters:
- Aufbau und Verarbeitung der/des Programmtextes – Variablentabelle – Stringspace – Fließkommaformate – Basicstacks (GDSUB, FOR-NEXT, ...)
- Beschreibung der wichtigsten Unterprogramme, z. B. Variablensuche, Garbage collection, Ausdrucksauswertung, CHRGET, ...
- Vollständig disassemblierte und sehr ausführlich deutsch kommentierte Auflistung des Applesoft-BASIC-Interpreters
- Übersichtliche Auflistung aller vom Interpreter benutzten RAM-Zeilen mit allen Verwendungszwecken
- Über 150 ausführlich dokumentierte Unterprogramme:
  – Funktion
  – Ein/Ausgabeparameter

Auch für Apple-IIe und c und Kompatible!

Apple II Pascal

Eine praktische Anleitung
von Arthur Luehrmann und Herbert Peckham
1982, 544 S., kart., DM 59,–

Dieses Buch ist unentbehrlich für alle, die die Programmiersprache PASCAL lernen wollen und Zugang zu einem Apple Computer haben.
Sie benötigen keinerlei Vorkenntnisse, sondern lernen an Hand von Beispielen und Übungen, wie man selbst PASCAL-Programme entwickelt und sie austestet und werden allmählich von Kapitel zu Kapitel vertrauter im Umgang mit dem Apple Computer.

Start mit Apple-Logo für II, IIe und IIc

Das kleine Logo-Einmaleins: Grafik • Text • Musik
Von D. Senftleben
1985, 222 Seiten, DM 35,–

Viele Mikrocomputer-Hersteller bieten für ihre Geräte neben BASIC und anderen Programmiersprachen zunehmend auch Logo an. Durch ihre Benutzerfreundlichkeit hat diese Sprache bereits viele Freunde im Ausbildungs- und Freizeitbereich gefunden. Dabei ist Logo eine mächtige Sprache, die auch dem anspruchsvollen Anwender kaum Wünsche offenläßt. Mittels Schildkrötengrafik wird das kleine Logo-Einmaleins in 12 Lektionen entwickelt. Große Bildschirmfotos begleiten den Leser durch die Lektionen. Das Buch verlangt aktive Mitarbeit. Es hat seinen Platz neben dem Computer und gibt Hilfen und Anregungen für eigenes Forschen. Dank des bausteinorientierten Konzepts kann jeder seine eigenen Teilbausteine erzeugen und sie zu neuen Blöcken zusammenfügen. Neben dem Einmaleins werden neue Einsatzbereiche für den Einsteiger erschlossen. Musik und Sound fehlen nicht.
In diesem Buch werden die beiden offiziellen Logo-Produkte der Firma Apple für die Rechnerfamilie Apple II, IIe und IIc behandelt und deren Unterschiede verdeutlicht. Weiterhin sind sämtliche Apple-Logo-Vokabeln übersichtlich zusammengestellt. Dieses Buch ist ideal zum problemlosen und vergnüglichen Start in die Apple-Logo-Welt.

Apple II Anwenderhandbuch

von Lon Poole
1982, 450 S., zahlr. Abb., kart., DM 56,–

Auch für diesen Computer haben wir den richtigen Leitfaden. Er erspart Ihnen zeitraubendes und nutzloses Suchen nach der wirklich verwendbaren Dokumentation für Ihren Computer. Das Anwenderhandbuch beschreibt zum einen den beliebten Apple II-Computer als solchen und gibt zum anderen ausführlich Auskunft über die normalen Peripheriebausteine und Zubehör einschließlich Disk-Laufwerken und Drucker. Mit Hilfe dieses Buches werden Sie Ihren Apple II erfolgreich einsetzen, denn der Informationsgehalt geht weit über das hinaus, was herstellerseitig an Literatur angeboten wird. Sie lernen BASIC auf zwei verschiedene Arten zu verwenden. Wie man den Gebrauch von Klang, Farbe und Grafik zum Optimum führt. Sie erhalten Tips für fortgeschrittene Programmerstellung. Sie erfahren die Verwendung des Maschinensprachen-Monitors u. v. m. Mit dem Apple II-Anwenderhandbuch werden Ihnen alle Möglichkeiten eröffnet, die in diesem Computer stecken.

Apple II Pascal

Sprache
1985, 197 S., DM 39,–

Apple II Pascal

Betriebssystem
1985, 256 S., DM 49,–

```
5300 REM
5301 REM -- $Dxxx : ADDA, ADD, ADDX --
5302 REM
5310 IF 0(2) MOD 4=3 THEN D$="ADDA" : GOTO 4620
5320 IF B(8)=0 OR 0(1)>1 THEN D$="ADD" : GOTO 4520
5330 D$="ADDX" : DIG=0(2) MOD 4 : GOSUB 1410 : GOTO 4420
5400 REM
5401 REM -- $Exxx : Speicher-Verschiebe-Befehle --
5402 REM
5410 IF 0(2) MOD 4<3 THEN 5510 ELSE IF 0(3)>3 THEN 1510
5420 ON 0(3) GOTO 5422,5423,5424
5421 D$="AS" : GOTO 5430
5422 D$="LS" : GOTO 5430
5423 D$="ROX" : GOTO 5430
5424 D$="RO"
5430 IF B(8) THEN D$=D$+"L " ELSE D$=D$+"R "
5440 G=2 : GOSUB 6010 : GOTO 1610
5500 REM
5501 REM -- $Exxx : Register-Verschiebe-Befehle --
5502 REM
5510 ON 0(1) MOD 4 GOTO 5512,5513,5514
5511 D$="AS" : GOTO 5520
5512 D$="LS" : GOTO 5520
5513 D$="ROX" : GOTO 5520
5514 D$="RO"
5520 IF B(8) THEN D$=D$+"L" ELSE D$=D$+"R"
5530 DIG=0(2) MOD 4 : GOSUB 1420 : ON B(5) GOTO 5560
5540 DIG=0(3) : IF DIG=0 THEN DIG=8
5550 D$=D$+"#"+HEX$(DIG)+",D"+HEX$(0(0)) : GOTO 1610
5560 D$=D$+"D"+HEX$(0(3))+",D"+HEX$(0(0)) : GOTO 1610
6000 REM
6001 REM -- Bestimme EA (Effektive Adresse) --
6002 REM
6010 MDE=0(1) : REG=0(0)
6020 ON MDE GOTO 6040,6050,6060,6070,6080,6100,6200
```

```
6030 D$=D$+"D"+HEX$(REG) : RETURN
6040 D$=D$+"A"+HEX$(REG) : RETURN
6050 D$=D$+"(A"+HEX$(REG)+")" : RETURN
6060 D$=D$+"(A"+HEX$(REG)+")+" : RETURN
6070 D$=D$+"-(A"+HEX$(REG)+")" : RETURN
6080 G=2 : GOSUB 1310 : D$=D$+W$ : GOTO 6050
6100 G=2 : Z=4 : GOSUB 1310 : DIG=ASC(MID$(W$,2,1))-48
6110 IF DIG MOD 8>0 THEN 6300 ELSE DIG=1+DIG\8
6120 IF MID$(W$,3,1)<>"0" THEN D$=D$+MID$(W$,3,1)
6130 D$=D$+RIGHT$(W$,1)+"("
6140 IF REG>7 THEN D$=D$+"PC" ELSE D$=D$+"A"+HEX$(REG)
6150 REG=ASC(LEFT$(W$,1))-48 : IF REG>9 THEN REG=REG-7
6160 IF REG<8 THEN D$=D$+",D" ELSE D$=D$+",A"
6170 D$=D$+HEX$(REG MOD 8) : GOSUB 1420 :
     MID$(D$,LEN(D$))=")" : RETURN
6200 ON REG GOTO 6210,6220,6230,6240,6300,6300,6300
6210 IF REG THEN G=4 ELSE G=2
6211 GOSUB 1310 : D$=D$+W$ : RETURN
6220 GOSUB 1010 : A=A1-2+WRD : GOSUB 1210 : D$=D$+W$ :
     RETURN
6230 REG=8 : GOTO 6100
6240 IF 0(5)+0(4)+B(8)>0 OR 0(3)>1 AND 0(3)<>5 THEN 6270
6250 IF G>2 THEN 6300 ELSE IF G=1 THEN D$=D$+"CCR"
     ELSE D$=D$+"SR"
6260 RETURN
6270 GOSUB 1310 : D$=D$+"#"+W$ : RETURN
6300 D$=D$+"?EA?" : RETURN
```

Hinweis: Der MACMONITOR befindet sich nicht auf den üblichen 5,25-Zoll-Peeker-Sammeldisketten. Sie können ihn jedoch für DM 30,- als 3,5-Sony-Macintosh-Diskette bestellen. Beachten Sie aber, daß diese Diskette nur das Programm und nicht zusätzlich das Microsoft-Basic enthält.

INPUT für alle Fälle

Tastatur – DOS – ProDOS

von Harald Grumser

INALL.DATA

```
10  FOR I = 768 TO 793: READ A: POKE I,A: NEXT
20  DATA 32,6,227,32,190,222,32,227,223,32,108,
    221,133,133,132,134,32,44,213,200,32,233,
    227,76,154,218
30  CALL 768, A$ : REM Beispiel
```

(Dieses im letzten Peeker bereits abgedruckte Programm enthielt 2 Tippfehler [277 statt 227]. Deshalb durcken wir es erneut ab.)

INALL

```
                    1   * Input-anything für Applesoft
                    2
                    3           ORG  $300
                    4           OBJ  *
                    5
                    6   FORPNT   EQU  $85
                    7   INLIN    EQU  $D52C
                    8   STRCPY   EQU  $DA9A
                    9   CHKSTR   EQU  $DD6C
                    10  CHKCOM   EQU  $DEBE
                    11  PTRGET   EQU  $DFE3
                    12  ERRDIR   EQU  $E306
                    13  STRLIT1  EQU  $E3E9
                    14
0300: 20 06 E3      15          JSR  ERRDIR     ;direkt?
0303: 20 BE DE      16          JSR  CHKCOM     ;Komma?
0306: 20 E3 DF      17          JSR  PTRGET     ;Var. suchen
0309: 20 6C DD      18          JSR  CHKSTR     ;String?
030C: 85 85         19          STA  FORPNT     ;Zeiger
030E: 84 86         20          STY  FORPNT+1   ; eintragen
0310: 20 2C D5      21          JSR  INLIN      ;Zeile holen
0313: C8            22          INY             ;$1FF -> $200
0314: 20 E9 E3      23          JSR  STRLIT1    ;Descriptor
0317: 4C 9A DA      24          JMP  STRCPY     ;eintragen

26 Bytes
```

Wer kennt nicht die häßliche „?EXTRA IGNORED"-Meldung des Applesoft-Interpreters nach Eingabe von Kommas oder Doppelpunkten beim INPUT-Befehl. Was im interaktiven Programm die schönsten Bildschirmmasken zerstört und beim unbedarften Anwender den Glauben an die gute deutsche Software ins Wanken bringt, wird beim Einlesen von Textdateien, die diese Zeichen enthalten (welcher Text beinhaltet kein Komma?) zum Programmierspektakel.

Eine Lösung stellt die hier vorgestellte INALL-Routine dar, die sowohl die Eingabe von Kommas und Doppelpunkten unterstützt als auch Hochkommas im Eingabestring erlaubt. Sie simuliert den INPUT-Befehl und kann aus jedem Applesoft-Programm, sowohl unter DOS als auch unter ProDOS, aufgerufen werden. Der Aufruf erfolgt durch „CALL 768,A$", wobei an Stelle von A$ jede Stringvariable (*keine* numerische Variable) benannt werden kann (max. Länge 239 Zeichen). Da auf die Benutzung des Ampersand-Vektors verzichtet wurde, kann die Routine mit dem **INALL.DATA**-Programm zu jeder Zeit in den Speicher gepokt oder direkt durch „BLOAD **INALL**" geladen werden, nachdem sie mit „BSAVE INALL,A$300,L26" gesichert wurde. Das Maschinenprogramm ist verschiebbar und läuft somit in jedem Speicherbereich, wobei 768 dann natürlich entsprechend zu ändern ist.

Block-Tracer für ProDOS

von Ulrich Stiehl

Im Gegensatz zu DOS 3.3 sind ProDOS-Datenträger blockmäßig organisiert, wobei 1 Block = 2 Sektoren = 2 mal 256 Bytes = 512 Bytes entspricht. Wenn man auf der Ebene des sog. MLI = Machine Language Interface programmieren möchte, ist es nützlich zu wissen, in welcher Form das in der Language Card befindliche PRODOS auf externe Datenträger (Disketten, RAM-Disks, Festplattenlaufwerke) zugreift. Hierzu dient das Analyseprogramm **Block-Tracer**, das in zwei Versionen geschrieben wurde: Die DISK-II-Version ist für die normalen Apple-Laufwerke und die RAM-Disk-Version für die 64K-Karte des Apple IIe/IIc gedacht. Die letztere Version ist ungefährlicher, zumal der RAM-Disk-Inhalt unter ProDOS ohnehin nur für unwichtige Dateien benutzt werden sollte, denn der Inhalt wird durch jedes Neubooten automatisch gelöscht.

Nach BRUN BLOCKTRACER wird bei jedem Diskettenzugriff der jeweilige Block in folgender Form angezeigt (Beispiele):

R0020 = Read = Lesen des Blocks $0020
W0100 = Write = Schreiben des Blocks $0100
S00FF = Seek = Suchen des Blocks $00FF
F0000 = Format = Formatieren des Blocks $0000

Mit Hilfe des Block-Tracers macht man einige interessante Entdeckungen. Beispielsweise kommt der Format-Befehl niemals vor, selbst dann nicht, wenn man von der Format-Option des FILER-Programms Gebrauch macht. (FILER und CONVERT können übrigens problemlos in Verbindung mit dem Block-Tracer benutzt werden, da der Bereich ab $0300 nicht überschrieben wird.) Ferner stellt man fest, daß der Seek-Befehl auch bei RAM-Disks benutzt wird, obgleich er eigentlich für das Positionieren des Lesekopfes gedacht ist und deshalb bei RAM-Disks sinnleer ist. Übrigens wird Seek sehr häufig auf Block $0000 angewandt, was bei DISK-II-Laufwerken zu den bekannten scharrenden Geräuschen führt.

Zu Testzwecken formatieren wir eine Leerdiskette, starten den Block-Tracer und führen anschließend folgende Diskettenoperationen durch:

BSAVE XXX, A$1000, L$200
(Erstspeicherung)

R0002 R0002 S0000 R0006 W0007 R0002 W0002 W0006 R0002 R0002 R0007 S0000 S0000 W0007 R0002 R0002 W0002 R0002 R0002 W0002

BSAVE XXX, A$1000, L$200
(Zweitspeicherung)

R0002 R0002 R0007 S0000 S0000 W0007 R0002 R0002 W0002 R0002 R0002 W0002

BSAVE XXX
(Drittspeicherung ohne Parameter)

R0002 R0002 R0007 S0000 W0007 R0002 R0002 W0002 R0002 R0002 W0002

BLOAD XXX
(Ladevorgang)

R0002 R0002 R0007

Wie ersichtlich, ist das erneute Speichern einer Datei erheblich weniger blockintensiv als der allererste BSAVE. Bei BLOAD/LOAD wird der Seek-Befehl offenbar niemals benutzt. Dies würde erklären, warum sich PRODOS-Disketten manchmal selbst nicht lesen können, z.B. wenn zuvor mit geschützten DOS-3.3.-Disketten gearbeitet wurde.

Den Block-Tracer kann man mit –PRODOS abstellen. Durch –FILER, –CONVERT, –BASIC.SYSTEM usw. wird er nicht zerstört, wohl aber, wenn ein anderes Maschinenprogramm in den Speicherbereich ab $0300 geladen wird.

Warnung:
Dieses Programm läuft **nur** unter ProDOS Version 1.0.1!

BLOCKTRACER

```
1              ORG  $300
2    *
3    * Block-Tracer von ProDOS für
4    * DISK-II-Driver von U.Stiehl
5    * August 1984
6    *
7    PRINT     EQU  $FDED
8    HEXOUT    EQU  $FDDA
9    DISKII    EQU  $F800
10   *
11   * ALT:
12   *
13   * $F800: A5 46  LDA $46
14   *        A6 47  LDX $47
0300: AD 8B C0 23             LDA  $C08B      ;READBK1
0303: AD 8B C0 24             LDA  $C08B      ;WRITEBK1
0306: A9 4C    25             LDA  #$4C       ;JMP
0308: 8D 00 F8 26             STA  DISKII
030B: A9 1C    27             LDA  #<VEKTOR
030D: 8D 01 F8 28             STA  DISKII+1
0310: A9 03    29             LDA  #>VEKTOR
0312: 8D 02 F8 30             STA  DISKII+2
0315: AD 81 C0 31             LDA  $C081      ;RDROM
0318: AD 81 C0 32             LDA  $C081      ;WRBK1
17   * NEU:
18   * $F800: JMP VEKTOR
19   *
20   * Block-Tracer-Vektor
21   * initialisieren
22   *
```

```
031B: 60          33            RTS
                  34  *
                  35  * Sprung vom DISK-II-Driver
                  36  * an diese Stelle
                  37  *
031C: 8C 8D 03    38  VEKTOR    STY  YSAVE
031F: 08          39            PHP
0320: 68          40            PLA
0321: 8D 8E 03    41            STA  PSAVE
                  42  *
0324: A2 00       43            LDX  #$00
0326: B5 40       44  RETTE1    LDA  $0040,X
0328: 9D 7D 03    45            STA  RETTE3,X
032B: E8          46            INX
032C: E0 10       47            CPX  #$10
032E: D0 F6       48            BNE  RETTE1
                  49  *
                  50  * Block-Nr. $HHLL anzeigen
                  51  *
0330: AD 81 C0    52            LDA  $C081       ;RDROM
0333: AD 81 C0    53            LDA  $C081       ;WRBK1
0336: A2 02       54            LDX  #$02        ;COMMAND
0338: BD 7D 03    55            LDA  RETTE3,X    ;S/R/W/F
033B: AA          56            TAX
033C: BD 8F 03    57            LDA  RDWR,X
033F: 20 ED FD    58            JSR  PRINT
0342: A2 07       59            LDX  #$07        ;HIGH
0344: BD 7D 03    60            LDA  RETTE3,X    ;BLOCK
0347: 20 DA FD    61            JSR  HEXOUT
034A: CA          62            DEX              ;LOW
034B: BD 7D 03    63            LDA  RETTE3,X    ;BLOCK
034E: 20 DA FD    64            JSR  HEXOUT
0351: A9 A0       65            LDA  #$A0        ;SPACE
0353: 20 ED FD    66            JSR  PRINT
0356: 20 ED FD    67            JSR  PRINT
0359: 20 ED FD    68            JSR  PRINT
                  69  *
035C: AD 8B C0    70            LDA  $C08B       ;RDBK1
035F: AD 8B C0    71            LDA  $C08B       ;WRBK1
                  72  *
0362: A2 00       73            LDX  #$00
0364: BD 7D 03    74  RETTE2    LDA  RETTE3,X
0367: 95 40       75            STA  $0040,X
0369: E8          76            INX
036A: E0 10       77            CPX  #$10
036C: D0 F6       78            BNE  RETTE2
                  79  *
036E: AC 8D 03    80            LDY  YSAVE
0371: AD 8E 03    81            LDA  PSAVE
0374: 48          82            PHA
0375: 28          83            PLP
0376: A5 46       84            LDA  $0046
0378: A6 47       85            LDX  $0047
                  86  *
                  87  * Zurück zum DISK-II-Driver
                  88  *
037A: 4C 04 F8    89            JMP  DISKII+4
                  90  *
                  91  RETTE3    DS   $10
038D: 00          92  YSAVE     HEX  00
038E: 00          93  PSAVE     HEX  00
                  94  *
                  95  * S = Seek
                  96  * R = Read
                  97  * W = Write
                  98  * F = Format (kommt nicht vor)
                  99  *
038F: D3 D2 D7   100  RDWR      ASC  "SRWF"
0392: C6
```

BLOCKTRACER1

```
                   1            ORG  $300
                   2  *
                   3  * Block-Tracer von ProDOS für
                   4  * RAM-Disk-Driver von U.Stiehl
                   5  * August 1984
                   6  *
                   7  PRINT     EQU  $FDED
                   8  HEXOUT    EQU  $FDDA
                   9  RAMDISK   EQU  $FF00
                  10  *
                  11  * ALT:
                  12  *
                  13  * $FF00: CLD
                  14  * $FF01: LDX #$0B
                  15  * $FF03: LDA $3C
                  16  *
                  17  * NEU:
                  18  *
                  19  * $FF00: JMP VEKTOR
                  20  * $FF03: LDA $3C
                  21  *
                  22  * Block-Tracer-Vektor
                  23  * initialisieren
                  24  *
0300: AD 8B C0    25            LDA  $C08B       ;READBK1
0303: AD 8B C0    26            LDA  $C08B       ;WRITEBK1
0306: A9 4C       27            LDA  #$4C        ;JMP
0308: 8D 00 FF    28            STA  RAMDISK
030B: A9 1C       29            LDA  #<VEKTOR
030D: 8D 01 FF    30            STA  RAMDISK+1
0310: A9 03       31            LDA  #>VEKTOR
0312: 8D 02 FF    32            STA  RAMDISK+2
0315: AD 81 C0    33            LDA  $C081       ;RDROM
0318: AD 81 C0    34            LDA  $C081       ;WRBK1
031B: 60          35            RTS
                  36  *
                  37  * Sprung vom RAM-Disk-Driver
                  38  * an diese Stelle
                  39  *
031C: 8D 8E 03    40  VEKTOR    STA  ASAVE
031F: 8E 8F 03    41            STX  XSAVE
0322: 8C 90 03    42            STY  YSAVE
                  43  *
0325: A2 00       44            LDX  #$00
0327: B5 40       45  RETTE1    LDA  $0040,X
0329: 9D 7E 03    46            STA  RETTE3,X
032C: E8          47            INX
032D: E0 10       48            CPX  #$10
032F: D0 F6       49            BNE  RETTE1
                  50  *
                  51  * Block-Nr. $HHLL anzeigen
                  52  *
0331: AD 81 C0    53            LDA  $C081       ;RDROM
0334: AD 81 C0    54            LDA  $C081       ;WRBK1
0337: A2 02       55            LDX  #$02        ;COMMAND
0339: BD 7E 03    56            LDA  RETTE3,X    ;S/R/W/F
033C: AA          57            TAX
033D: BD 91 03    58            LDA  RDWR,X
0340: 20 ED FD    59            JSR  PRINT
0343: A2 07       60            LDX  #$07        ;HIGH
0345: BD 7E 03    61            LDA  RETTE3,X    ;BLOCK
0348: 20 DA FD    62            JSR  HEXOUT
034B: CA          63            DEX              ;LOW
034C: BD 7E 03    64            LDA  RETTE3,X    ;BLOCK
034F: 20 DA FD    65            JSR  HEXOUT
0352: A9 A0       66            LDA  #$A0
0354: 20 ED FD    67            JSR  PRINT
0357: 20 ED FD    68            JSR  PRINT
035A: 20 ED FD    69            JSR  PRINT
                  70  *
035D: AD 8B C0    71            LDA  $C08B       ;RDBK1
0360: AD 8B C0    72            LDA  $C08B       ;WRBK1
                  73  *
0363: A2 00       74            LDX  #$00
0365: BD 7E 03    75  RETTE2    LDA  RETTE3,X
0368: 95 40       76            STA  $0040,X
036A: E8          77            INX
036B: E0 10       78            CPX  #$10
036D: D0 F6       79            BNE  RETTE2
                  80  *
036F: AD 8E 03    81            LDA  ASAVE
0372: AE 8F 03    82            LDX  XSAVE
0375: AC 90 03    83            LDY  YSAVE
                  84  *
0378: A2 0B       85            LDX  #$0B
037A: D8          86            CLD
                  87  *
                  88  * Zurück zum RAM-Disk-Driver
                  89  *
037B: 4C 03 FF    90            JMP  RAMDISK+3
                  91  *
                  92  RETTE3    DS   $10
038E: 00          93  ASAVE     HEX  00
038F: 00          94  XSAVE     HEX  00
0390: 00          95  YSAVE     HEX  00
                  96  *
                  97  * S = Seek
                  98  * R = Read
                  99  * W = Write
                 100  * F = Format (kommt nicht vor)
                 101  *
0391: D3 D2 D7   102  RDWR      ASC  "SRWF"
0394: C6
```

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.

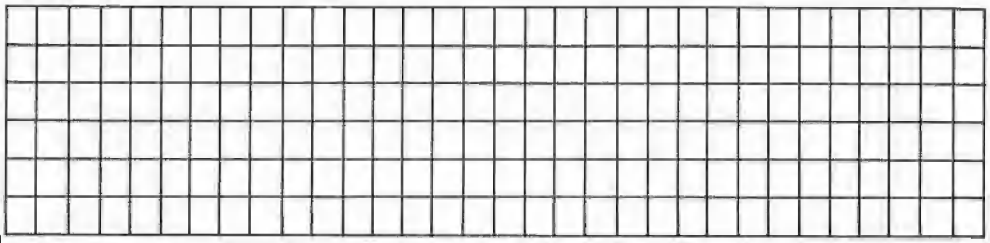
peeker MAGAZIN FÜR APPLE-COMPUTER

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

Hüthig-FACHBUCH-TIP

Apple ProDOS für Aufsteiger

Band 1

von Ulrich Stiehl
1984, 202 S., kart., DM 28,-
ISBN 3-7785-1027-4
Hüthig Verlag, Heidelberg

ProDOS ist das neue „professionelle DOS" (Professional Disk Operating System) für den Apple IIe sowie den mit einer Language Card ausgestatteten Apple II Plus. Reine Applesoft-Programmierer, die bereits unter DOS 3.3 programmiert haben, werden sich schnell an ProDOS gewöhnen, da die diesbezüglichen Unterschiede zwischen DOS 3.3 und ProDOS weniger gravierend sind.
Sinngemäß liegt das Schwergewicht in dem ersten Band von „ProDOS für Aufsteiger" auf der Assembler-Programmierung, da Assembler-Programmierer unter ProDOS völlig umdenken müssen. Insbesondere sind alle früheren Assemblerprogramme unter ProDOS nicht mehr lauffähig und bedürfen einer intensiven Überarbeitung. Band 1 befaßt sich überwiegend 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

BESTELLCOUPON

Bitte ausfüllen und an Hüthig Vertriebsservice, Postfach 10 28 69 · 6900 Heidelberg schicken.

Buchtitel

Name

Straße

Ort

Unterschrift

FORMAT-Befehl für ProDOS

von U. Stiehl

Der FORMAT- oder INIT-Befehl ist im ProDOS-Betriebssystem u.a. aus Platzgründen nicht enthalten. Statt dessen wird von Firma Apple empfohlen, stets eine größere Anzahl bereits initialisierter Disketten bereitzuhalten, da normale Anwenderprogramme eine Formatierung aus dem Programm heraus nicht zulassen. Zur Formatierung muß nämlich eigens der FILER geladen werden, der jedes residente Programm einschließlich des BASIC.SYSTEMs zerstören würde. Mit der nachstehend beschriebenen Utility FORMAT.LC ist es nunmehr möglich, nicht nur aus einem Anwenderprogramm heraus, sondern auch über die Tastatur den FORMAT-Befehl aufzurufen.

Mittlerweile wurden von mir bereits zwei Formatierungsroutinen für ProDOS geschrieben:

PRODOS.INIT, das in dem Buch „ProDOS für Aufsteiger" enthalten ist, basiert noch auf der DOS-RWTS-Routine (RWTS = Read Write Track Sector) und ist völlig in sich abgeschlossen, d.h. benutzt keinerlei programmexterne Routinen, so daß es sowohl nach dem Booten von ProDOS wie auch nach dem Booten von DOS 3.3 eingesetzt werden kann. Es empfiehlt sich stets dann, wenn Fremdlaufwerke, etwa 80-Spur-Drives, erstmals für ProDOS konfiguriert werden sollen, denn bekanntlich gibt es bislang keine bootfähigen ProDOS-Systemdisketten für Fremdlaufwerke. Gegebenenfalls könnte PRODOS.INIT auch in Anwenderprogramme eingebunden werden, doch ist es hierfür nicht konzipiert worden, zumal es den RAM-Bereich von $8000 bis $90FF einnimmt.

FORMAT.LC basiert auf dem ProDOS-System selbst und setzt deshalb die Existenz des in der Language Card Bank 1 befindlichen Teilprogramms PRODOS voraus. FORMAT.LC liegt selbst in der Language Card, und zwar in Bank 2 ab $D300 und nimmt damit keinen unnötigen Speicherraum in den unteren 48K weg, so daß es bestens in Anwenderprogramme eingebunden werden kann. Wie mit PRODOS.INIT können auch mit FORMAT.LC Laufwerke unterschiedlichen Typs (35-, 40- und 80-Spur-Laufwerke für 5,25-Zoll-Disketten sowie beispielsweise 80-Track-Drives für 3,5-Zoll-Disketten formatiert werden, wenn die üblichen Controller-Konventionen bezüglich der Schrittmotorsteuerung usw. eingehalten werden. Als Faustregel gilt: Wenn die unveränderte alte DOS-RWTS (von der Spurenanzahl abgesehen) auf ein Fremdlaufwerk angewendet werden kann, dann sind auch unsere Formatierungsprogramme lauffähig.

Nachfolgend sind das Maschinenprogramm FORMAT.LC sowie das übergeordnete Applesoft-Programm FORMAT.LC.START gelistet. Wegen des ungewöhnlichen Umfangs des Maschinenprogramms kann hier nur der Hex-Dump abgedruckt werden.

1. Anwendung von FORMAT.LC

Wenn man mit
RUN FORMAT.LC.START
das Applesoft-Programm startet, erscheint ein kurzes Menü, das nach
– Slot (1-7)
– Drive (1-2)
– Spuren (35-80)
fragt. Diese Parameter werden in das Maschinenprogramm gepokt, das dann aufgerufen wird. Es meldet sich mit
INIT STIEHL, D1 J/N oder
INIT STIEHL, D2 J/N
Mit „N" oder „n" könnte man jetzt noch das Formatierungsprogramm verlassen. Anderfalls wird es nach „J" oder „j" ohne weitere Warnung gestartet. Man muß also bereits zuvor eine Leerdiskette in das entsprechende Laufwerk gelegt haben. „STIEHL" steht für den Volume-Namen, den man später mit
RENAME /STIEHL,/MAYER o.ä.
in den eigenen Namen abändern kann.
Wenn während der Formatierung keine

FORMAT.LC

```
Assembler-Hex-Dump

BSAVE FORMAT.LC, A$80BB, L$0CA0
Starten mit BRUN FORMAT.LC
Programm verschiebt sich auto-
matisch in die LC Bank 2
$D100-$DD5A

$80B8: 00 00 00 AD 83 C0 AD 83
$80C0: C0 A0 00 A9 81 8D CF 80
$80C8: A9 D1 8D D2 80 B9 00 81
$80D0: 99 00 D1 C8 D0 F7 EE CF
$80D8: 80 EE D2 80 AD D2 80 C9
$80E0: E0 D0 EA AD 81 C0 AD 81
$80E8: C0 B9 F5 80 99 40 BF C8
$80F0: C0 09 D0 F5 60 AD 83 C0
$80F8: AD 83 C0 4C 00 D1 D5 D3
$8100: 4C 3D DB 06 01 23 C9 00
$8108: D0 02 18 60 C9 02 D0 05
$8110: A9 2B 4C 21 D1 C9 01 D0
$8118: 05 A9 27 4C 21 D1 18 69
$8120: 30 38 60 0A 0E 24 D5 8D
$8128: 36 D5 8A 4A 4A 4A 4A A8
$8130: AD 36 D5 20 C6 D2 4E 24
$8138: D5 60 AA 29 70 8D 23 D5
$8140: 8A AE 23 D5 2A A9 00 2A
$8148: D0 06 BD 8A C0 4C 53 D1
$8150: BD 8B C0 BD 89 C0 A9 D7
$8158: 85 DA A9 50 8D 24 D5 A9
$8160: 00 20 23 D1 A5 DA F0 06
$8168: 20 3A D4 4C 64 D1 A9 01
$8170: 85 D3 A9 AA 85 D0 AD 20
$8178: D5 18 69 02 85 D4 A9 00
$8180: 85 D1 A5 D1 AE 23 D5 20
$8188: 23 D1 AE 23 D5 BD 8D C0
$8190: BD 8E C0 A8 BD 8E C0 BD
$8198: 8C C0 98 10 05 A9 02 4C
$81A0: F9 D1 20 63 D4 90 0E A9
$81A8: 01 A4 D4 CC 1F D5 B0 02
$81B0: A9 04 4C F9 D1 A4 D4 CC
$81B8: 1F D5 B0 05 A9 04 4C F9
$81C0: D1 CC 20 D5 90 05 A9 03
$81C8: 4C F9 D1 AD 22 D5 8D 25
$81D0: D5 CE 25 D5 D0 05 A9 01
$81D8: 4C F9 D1 AE 23 D5 20 6A
$81E0: D2 B0 EE A5 D8 D0 EA AE
$81E8: 23 D5 20 07 D2 B0 E2 E6
$81F0: D1 A5 D1 C9 23 90 8B A9
$81F8: 00 48 AE 23 D5 BD 88 C0
$8200: A9 00 20 23 D1 68 60 A0
$8208: 20 88 F0 5C BD 8C C0 10
$8210: FB 49 D5 D0 F4 EA BD 8C
$8218: C0 10 FB C9 AA D0 F2 A0
$8220: 56 BD 8C C0 10 FB C9 AD
$8228: D0 E7 A9 00 88 84 D5 BD
$8230: 8C C0 10 FB C9 96 D0 30
$8238: A4 D5 D0 F0 84 D5 BD 8C
$8240: C0 10 FB C9 96 D0 21 A4
$8248: D5 C8 D0 F0 BD 8C C0 10
$8250: FB C9 96 D0 13 BD 8C C0
$8258: 10 FB C9 DE D0 0A EA BD
$8260: 8C C0 10 FB C9 AA F0 5C
$8268: 38 60 A0 FC 84 DC C8 D0
$8270: 04 E6 DC F0 F3 BD 8C C0
$8278: 10 FB C9 D5 D0 F0 EA BD
$8280: 8C C0 10 FB C9 AA D0 F2
$8288: A0 03 BD 8C C0 10 FB C9
$8290: 96 D0 E7 A9 00 85 DB BD
$8298: 8C C0 10 FB 2A 85 DD BD
$82A0: 8C C0 10 FB 25 DD 99 D7
$82A8: 00 45 DB 88 10 E7 A8 D0
$82B0: B7 BD 8C C0 10 FB C9 DE
$82B8: D0 AE EA BD 8C C0 10 FB
$82C0: C9 AA D0 A4 18 60 8E 37
$82C8: D5 8D 36 D5 CD 24 D5 F0
$82D0: 5C A9 00 8D 38 D5 AD 24
$82D8: D5 8D 39 D5 38 ED 36 D5
$82E0: F0 37 B0 07 49 FF EE 24
$82E8: D5 90 05 69 FE CE 24 D5
$82F0: CD 38 D5 90 03 AD 38 D5
$82F8: C9 0C B0 01 A8 38 20 1D
$8300: D3 B9 4B D4 20 3A D4 AD
$8308: 39 D5 18 20 20 D3 B9 57
$8310: D4 20 3A D4 EE 38 D5 D0
```

Störungen auftraten, kehrt FORMAT.LC zum Applesoft-Programm ohne Meldung zurück. Im Falle eines Fehlers wird die Routine sofort abgebrochen und ein „E“ für Error angezeigt. Ferner wird die Fehler-Nummer in der Speicherstelle 767 abgelegt, die durch das Applesoft-Programm abgerufen werden kann. Die häufigste Fehler-Nummer ist 39 = I/O-Error, d.h. keine Diskette im Laufwerk, nicht-belegter Slot und/oder Drive usw.

Das Applesoft-Programm FORMAT.LC-.START ist entbehrlich, wenn keine Parameter geändert werden müssen. FORMAT.LC ist eingestellt auf
Slot 6
Drive 1
Spuren 35
Diese Ersatzwerte lassen sich wie folgt ändern (Beispiel):
BLOAD FORMAT.LC
CALL – 151
8103: 05 (Slot 5)
8104: 02 (Drive 2)
8105: 50 (80 Spuren)
Ctrl-C
BSAVE FORMAT.LC
Wenn man FORMAT.LC ohne das Applesoft-Programm gestartet hat, so kann man es jederzeit mit
CALL 48960
aufrufen, und zwar sowohl über die Tastatur wie auch aus einem Programm heraus.

2. Technisches zu FORMAT.LC

FORMAT.LC wird zunächst in den Speicherbereich $80BB-$8D5A geladen, da unter ProDOS Programme nicht direkt in die Language Card Bank 2 gelegt werden können. Ein kurzes Move-Programm verschiebt erstens das eigentliche FORMAT.LC ab $8100 in die Bank 2 ab $D100-$DD5A und installiert zweitens einen Sprungbefehl in der ProDOS Global Page ab $BF40 (= dezimal 48960). Hier steht normalerweise der Copyrightvermerk „COPR. APPLE,1983“, der zugleich die einzige noch freie Stelle in der Global Page darstellt. FORMAT.LC kann durch CALL 48960 oder durch JSR $BF40 aufgerufen werden. Es ist damit völlig benutzertransparent und funktioniert auch dann, wenn sich das BASIC.SYSTEM nicht im Speicher befindet.

FORMAT.LC überschreibt den Bereich des sog. REBOOT-Programms in der Bank 2, das in „ProDOS für Aufsteiger“ erstmals erwähnt wurde und nach der Installierung von FORMAT.LC nicht mehr per MLI (Machine Language Interface) aufgerufen werden darf. Ferner ändert FORMAT.LC die folgende Speicherstelle:
$F809: CA D0 2A alt
$F809: 18 90 04 neu
Damit wird die Disk-Driver-Routine gepatcht, welche normalerweise die zulässige Block-Gesamtzahl von 280 überprüft. Dieser Patch bleibt auch nach der Formatierung bestehen.

FORMAT.LC rettet zunächst den benötigten Teil der Zero-Page $0030-$00FF in den Eingabepuffer $0230-$02FF, initialisiert dann die Diskette in der gewünschten Spurenanzahl und legt schließlich die Blocks 0-6 an (Urlader, Volume-Directory und Volume Bit Map). Dann wird die Zero-Page wieder hergestellt, eine mögliche Fehler-Nummer nach $02FF übertragen ($02FF: 00 bedeutet keinen Fehler) und schließlich das Programm über eine kurze Routine ab $0230 verlassen, welche die Language Card wieder abstellt. Man sollte also niemals FORMAT.LC durch Ctrl-Reset gewaltsam verlassen, da sonst die Zero-Page nicht mehr die alten Werte enthält. Im Zweifelsfall öffne man die Laufwerkklappe, worauf die Formatierungsroutine sofort abgebrochen wird.

FORMAT.LC enthält zwei interne Puffer, nämlich erstens einen 2-Block-Puffer mit dem Urlader für Block 0 und 1, der unverändert der ProDOS-Systemdiskette als Kleinzitat entnommen ist, sowie zweitens einen 1-Block-Puffer zur Anlage der Diskettenblocks 2-6. Ferner wird der Eingabepuffer ab $0230 – wie zuvor erwähnt – benutzt.

Aufgrund eines Fehlers im BASIC.SYSTEM empfiehlt es sich, die Input-Output-Vektoren im FORMAT.LC.START-Programm vorübergehend vom BASIC.SYSTEM abzuhängen, da Applesoft sonst in den Trace-Modus gelangt, was mit der Anzeige des Mini-Menüs „INIT STIEHL, D1“ und der Belegung der Zero-Page zusammenhängt.

FORMAT.LC benutzt direkt den Disk-Driver ab $F800 und nicht das MLI im engeren Sinne. Letzteres würde nämlich gar nicht zulassen, daß ein sich in der Bank 2 befindlicher Datenpuffer auf die Diskette geschrieben wird, von den Bank-Switching-Konflikten einmal ganz abgesehen. Wie man den Disk-Driver direkt benutzt, wird anhand des nachstehenden Beispielprogramms **DISKDRIVER.DEMO** gezeigt. us

FORMAT.LC

```
$8318: BD 20 3A D4 18 AD 24 D5
$8320: 29 03 2A 0D 37 D5 AA BD
$8328: 80 C0 AE 37 D5 60 20 0E
$8330: D5 BD 8D C0 BD 8E C0 A9
$8338: FF 9D 8F C0 DD 8C C0 48
$8340: 68 EA A0 04 48 68 20 A5
$8348: D3 88 D0 F8 A9 D5 20 A4
$8350: D3 A9 AA 20 A4 D3 A9 AD
$8358: 20 A4 D3 A0 56 EA EA EA
$8360: D0 03 20 0E D5 EA EA A9
$8368: 96 9D 8D C0 DD 8C C0 88
$8370: D0 F0 24 00 EA 20 0E D5
$8378: A9 96 9D 8D C0 DD 8C C0
$8380: A9 96 EA C8 D0 EF 20 A4
$8388: D3 A9 DE 20 A4 D3 A9 AA
$8390: 20 A4 D3 A9 EB 20 A4 D3
$8398: A9 FF 20 A4 D3 BD 8E C0
$83A0: BD 8C C0 60 EA 48 68 9D
$83A8: 8D C0 DD 8C C0 60 38 BD
$83B0: 8D C0 BD 8E C0 30 5E A9
$83B8: FF 9D 8F C0 DD 8C C0 48
$83C0: 68 20 1B D4 20 1B D4 9D
$83C8: 8D C0 DD 8C C0 EA 88 D0
$83D0: F0 A9 D5 20 2D D4 A9 AA
$83D8: 20 2D D4 A9 96 20 2D D4
$83E0: A5 D3 20 1C D4 A5 D1 20
$83E8: 1C D4 A5 D2 20 1C D4 A5
$83F0: D3 45 D1 45 D2 48 4A 05
$83F8: D0 9D 8D C0 BD 8C C0 68
$8400: 09 AA 20 2C D4 A9 DE 20
$8408: 2D D4 A9 AA 20 2D D4 A9
$8410: EB 20 2D D4 18 BD 8E C0
$8418: BD 8C C0 60 48 4A 05 D0
$8420: 9D 8D C0 DD 8C C0 68 EA
$8428: EA EA 09 AA EA EA 48 68
$8430: 9D 8D C0 DD 8C C0 60 00
$8438: 00 00 A2 11 CA D0 FD E6
$8440: D9 D0 02 E6 DA 38 E9 01
$8448: D0 F0 60 01 30 28 24 20
$8450: 1E 1D 1C 1C 1C 1C 1C 70
$8458: 2C 26 22 1F 1E 1D 1C 1C
$8460: 1C 1C 1C AD 21 D5 85 D6
$8468: A0 80 A9 00 85 D2 4C 73
$8470: D4 A4 D4 AE 23 D5 20 AE
$8478: D3 90 03 4C 0E D5 AE 23
$8480: D5 20 2E D3 E6 D2 A5 D2
$8488: C9 10 90 E5 A0 0F 84 D2
$8490: AD 22 D5 8D 25 D5 99 26
$8498: D5 88 10 FA A5 D4 38 E9
$84A0: 05 A8 20 0E D5 20 0E D5
$84A8: 48 68 EA EA 88 D0 F3 AE
$84B0: 23 D5 20 6A D2 B0 3C A5
$84B8: D8 F0 13 C6 D4 A5 D4 CD
$84C0: 1F D5 B0 2F 38 60 AE 23
$84C8: D5 20 6A D2 B0 1A AE 23
$84D0: D5 20 07 D2 B0 12 A4 D8
$84D8: B9 26 D5 30 0B A9 FF 99
$84E0: 26 D5 C6 D2 10 E0 18 60
$84E8: CE 25 D5 D0 D9 C6 D6 D0
$84F0: 02 38 60 AD 22 D5 0A 8D
$84F8: 25 D5 AE 23 D5 20 6A D2
$8500: B0 06 A5 D8 C9 0F F0 07
$8508: CE 25 D5 D0 ED 38 60 A2
$8510: D6 20 0E D5 20 0E D5 24
$8518: 00 CA D0 F5 4C 68 D4 10
$8520: 1B 03 10 60 80 00 00 00
$8528: 00 00 00 00 00 00 00 00
$8530: 00 00 00 00 00 00 00 60
$8538: A0 B1 01 38 B0 03 4C 32
$8540: A1 86 43 C9 03 08 8A 29
$8548: 70 4A 4A 4A 4A 09 C0 85
$8550: 49 A0 FF 84 48 28 C8 B1
$8558: 48 D0 3A B0 0E A9 03 8D
$8560: 00 08 E6 3D A5 49 48 A9
$8568: 5B 48 60 85 40 85 48 A0
$8570: 63 B1 48 99 94 09 C8 C0
$8578: EB D0 F6 A2 06 BC 1D 09
$8580: BD 24 09 99 F2 09 BD 2B
$8588: 09 9D 7F 0A CA 10 EE A9
$8590: 09 85 49 A9 86 A0 00 C9
$8598: F9 B0 2F 85 48 84 60 84
$85A0: 4A 84 4C 84 4E 84 47 C8
$85A8: 84 42 C8 84 46 A9 0C 85
$85B0: 61 85 4B 20 12 09 B0 68
$85B8: E6 61 E6 61 E6 46 A5 46
$85C0: C9 06 90 EF AD 00 0C 0D
$85C8: 01 0C D0 6D A9 04 D0 02
```

```
$85D0: A5 4A 18 6D 23 0C A8 90
$85D8: 0D E6 4B A5 4B 4A B0 06
$85E0: C9 0A F0 55 A0 04 84 4A
$85E8: AD 02 09 29 0F A8 B1 4A
$85F0: D9 02 09 D0 DB 88 10 F6
$85F8: 29 F0 C9 20 D0 3B A0 10
$8600: B1 4A C9 FF D0 33 C8 B1
$8608: 4A 85 46 C8 B1 4A 85 47
$8610: A9 00 85 4A A0 1E 84 4B
$8618: 84 61 C8 84 4D 20 12 09
$8620: B0 17 E6 61 E6 61 A4 4E
$8628: E6 4E B1 4A 85 46 B1 4C
$8630: 85 47 11 4A D0 E7 4C 00
$8638: 20 4C 3F 09 26 50 52 4F
$8640: 44 4F 53 20 20 20 20 20
$8648: 20 20 20 20 A5 60 85 44
$8650: A5 61 85 45 6C 48 00 08
$8658: 1E 24 3F 45 47 76 F4 D7
$8660: D1 B6 4B B4 AC A6 2B 18
$8668: 60 4C BC 09 A9 9F 48 A9
$8670: FF 48 A9 01 A2 00 4C 79
$8678: F4 20 58 FC A0 1C B9 50
$8680: 09 99 AE 05 88 10 F7 4C
$8688: 4D 09 AA AA AA A0 D5 CE
$8690: C1 C2 CC C5 A0 D4 CF A0
$8698: CC CF C1 C4 A0 D0 D2 CF
$86A0: C4 CF D3 A0 AA AA AA A5
$86A8: 53 29 03 2A 05 2B AA BD
$86B0: 80 C0 A9 2C A2 11 CA D0
$86B8: FD E9 01 D0 F7 A6 2B 60
$86C0: A5 46 29 07 C9 04 29 03
$86C8: 08 0A 28 2A 85 3D A5 47
$86D0: 4A A5 46 6A 4A 4A 85 41
$86D8: 0A 85 51 A5 45 85 27 A6
$86E0: 2B BD 89 C0 20 BC 09 E6
$86E8: 27 E6 3D E6 3D B0 03 20
$86F0: BC 09 BC 88 C0 60 A5 40
$86F8: 0A 85 53 A9 00 85 54 A5
$8700: 53 85 50 38 E5 51 F0 14
$8708: B0 04 E6 53 90 02 C6 53
$8710: 38 20 6D 09 A5 50 18 20
$8718: 6F 09 D0 E3 A0 7F 84 52
$8720: 08 28 38 C6 52 F0 CE 18
$8728: 08 88 F0 F5 BD 8C C0 10
$8730: FB 00 00 00 00 00 00 00
$8738: 00 00 4C 6E A0 53 4F 53
$8740: 20 42 4F 4F 54 20 20 31
$8748: 2E 31 20 0A 53 4F 53 2E
$8750: 4B 45 52 4E 45 4C 20 20
$8758: 20 20 20 53 4F 53 20 4B
$8760: 52 4E 4C 49 2F 4F 20 45
$8768: 52 52 4F 52 08 00 46 49
$8770: 4C 45 20 27 53 4F 53 2E
$8778: 4B 45 52 4E 45 4C 27 20
$8780: 4E 4F 54 20 46 4F 55 4E
$8788: 44 25 00 49 4E 56 41 4C
$8790: 49 44 20 4B 45 52 4E 45
$8798: 4C 20 46 49 4C 45 3A 00
$87A0: 00 0C 00 1E 0E 1E 04 A4
$87A8: 78 D8 A9 77 8D DF FF A2
$87B0: FB 9A 2C 10 C0 A9 40 8D
$87B8: CA FF A9 07 8D EF FF A2
$87C0: 00 CE EF FF 8E 00 20 AD
$87C8: 00 20 D0 F5 A9 01 85 E0
$87D0: A9 00 85 E1 A9 00 85 85
$87D8: A9 A2 85 86 20 BE A1 E6
$87E0: E0 A9 00 85 E6 E6 86 E6
$87E8: 86 E6 E6 20 BE A1 A0 02
$87F0: B1 85 85 E0 C8 B1 85 85
$87F8  E1 D0 EA A5 E0 D0 E6 AD
$8800: 6C A0 85 E2 AD 6D A0 85
$8808: E3 18 A5 E3 69 02 85 E5
$8810: 38 A5 E2 ED 23 A4 85 E4
$8818: A5 E5 E9 00 85 E5 A0 00
$8820: B1 E2 29 0F CD 11 A0 D0
$8828: 21 A8 B1 E2 D9 11 A0 D0
$8830: 19 88 D0 F6 A0 00 B1 E2
$8838: 29 F0 C9 20 F0 3E C9 F0
$8840: F0 08 AE 64 A0 A0 13 4C
$8848: D4 A1 18 A5 E2 6D 23 A4
$8850: 85 E2 A5 E3 69 00 85 E3
$8858: A5 E4 C5 E2 A5 E5 E5 E3
$8860: B0 BC 18 A5 E4 6D 23 A4
$8868: 85 E2 A5 E5 69 00 85 E3
$8870: C6 E6 D0 95 AE 4F A0 A0
$8878: 1B 4C D4 A1 A0 11 B1 E2
$8880: 85 E0 C8 B1 E2 85 E1 AD
```

```
$8888: 66 A0 85 85 AD 67 A0 85
$8890: 86 20 BE A1 AD 68 A0 85
$8898: 85 AD 69 A0 85 86 AD 00
$88A0: 0C 85 E0 AD 00 0D 85 E1
$88A8: 20 BE A1 A2 07 BD 00 1E
$88B0: DD 21 A0 F0 08 AE 64 A0
$88B8: A0 13 4C D4 A1 CA 10 ED
$88C0: A9 00 85 E7 E6 E7 E6 86
$88C8: E6 86 A6 E7 BD 00 0C 85
$88D0: E0 BD 00 0D 85 E1 A5 E0
$88D8: D0 04 A5 E1 F0 06 20 BE
$88E0: A1 4C 8A A1 18 AD 6A A0
$88E8: 6D 08 1E 85 E8 AD 6B A0
$88F0: 6D 09 1E 85 E9 6C E8 00
$88F8: A9 01 85 87 A5 E0 A6 E1
$8900: 20 79 F4 B0 01 60 AE 32
$8908: A0 A0 09 4C D4 A1 84 E7
$8910: 38 A9 28 E5 E7 4A 18 65
$8918: E7 A8 BD 29 A0 99 A7 05
$8920: CA 88 C6 E7 D0 F4 AD 40
$8928: C0 4C EF A1 00 00 00 00

(Blockpuffer freilassen in
diesem Bereich)
$8B38: 00 00 00 01 E0 A9 00 8D
$8B40: 3A DB A2 30 B5 00 9D 00
$8B48: 02 E8 D0 F8 A2 F0 A0 00
$8B50: B9 8F DB 95 00 C8 E8 D0
$8B58: F7 AD 04 D1 09 B0 8D AE
$8B60: DB A2 00 BD 9F DB F0 06
$8B68: 20 F0 00 E8 D0 F5 2C 10
$8B70: C0 AD 00 C0 10 FB 2C 10
$8B78: C0 C9 CA F0 3A C9 EA F0
$8B80: 36 C9 CE F0 07 C9 EE F0
$8B88: 03 4C 71 DB 4C FE DB 8D
$8B90: 81 C0 8D 81 C0 20 ED FD
$8B98: 8D 83 C0 8D 83 C0 60 8D
$8BA0: C9 CE C9 D4 A0 D3 D4 C9
$8BA8: C5 C8 CC AC A0 C4 B2 A0
$8BB0: CA AF CE A0 20 8D 00 A9
$8BB8: 18 8D 09 F8 A9 90 8D 0A
$8BC0: F8 A9 04 8D 0B F8 AD 3B
$8BC8: DB 8D 6F D1 AD 05 D1 8D
$8BD0: F4 D1 AD 03 D1 AE 04 D1
$8BD8: 0A 0A 0A 0A E0 01 F0 02
$8BE0: 09 80 8D 3C DB AD 3C DB
$8BE8: 08 78 20 3A D1 28 20 06
$8BF0: D1 B0 03 4C 42 DC 8D 3A
$8BF8: DB A9 C5 20 F0 00 A2 30
$8C00: BD 00 02 95 00 E8 D0 F8
$8C08: BD 1C DC 9D 30 02 E8 E0
$8C10: 07 D0 F5 AD 3A DB 8D FF
$8C18: 02 4C 30 02 AD 81 C0 AD
$8C20: 81 C0 60 84 44 85 45 A9
$8C28: 02 85 42 AD 3C DB 85 43
$8C30: A9 00 85 47 60 A9 00 AA
$8C38: 9D 3A D9 9D 3A DA E8 D0
$8C40: F7 60 A0 3A A9 D5 20 23
$8C48: DC A9 00 85 46 20 00 F8
$8C50: 90 03 4C F6 DB A0 3A A9
$8C58: D7 20 23 DC A9 01 85 46
$8C60: 20 00 F8 90 03 4C F6 DB
$8C68: 20 35 DC A9 03 8D 3C D9
$8C70: A2 00 18 BD 82 DC 69 10
$8C78: 9D 3E D9 E8 E0 07 D0 F2
$8C80: F0 09 E6 43 44 39 35 38
$8C88: 3C 00 00 A9 00 8D 8A DC
$8C90: AD 05 D1 8D 89 DC A2 03
$8C98: 0E 89 DC 2E 8A DC CA D0
$8CA0: F7 A9 C3 8D 5C D9 A9 27
$8CA8: 8D 5D D9 A9 0D 8D 5E D9
$8CB0: A9 06 8D 61 D9 AD 89 DC
$8CB8: 8D 63 D9 AD 8A DC 8D 64
$8CC0: D9 A0 3A A9 D9 20 23 DC
$8CC8: A9 02 85 46 20 00 F8 90
$8CD0: 03 4C F6 DB 20 35 DC A9
$8CD8: 02 8D 3A D9 A9 04 8D 3C
$8CE0: D9 A0 3A A9 D9 20 23 DC
$8CE8: A9 03 85 46 20 00 F8 90
$8CF0: 03 4C F6 DB 20 35 DC A9
$8CF8: 03 8D 3A D9 A9 05 8D 3C
$8D00: D9 A0 3A A9 D9 20 23 DC
$8D08: A9 04 85 46 20 00 F8 90
$8D10: 03 4C F6 DB 20 35 DC A9
$8D18: 04 8D 3A D9 A0 3A A9 D9
$8D20: 20 23 DC A9 05 85 46 20
$8D28: 00 F8 90 03 4C F6 DB 20
```

```
$8D30: 35 DC A9 FF AE 05 D1 CA
$8D38: 9D 3A D9 CA D0 FA A9 01
$8D40: 8D 3A D9 A0 3A A9 D9 20
$8D48: 23 DC A9 06 85 46 20 00
$8D50: F8 90 03 4C F6 DB 4C FE
$8D58: DB D5 D3 00 00 00 00 00
```

FORMAT.LC.START

```
Applesoft-Listing

10  PRINT  CHR$ (4) "BRUN FORMAT.LC":
    REM FORMAT.LC in Language Card
    schieben
15  PRINT  CHR$ (21): POKE 242,0:
    PR# 0: IN# 0: REM BASIC.SYSTEM
    abhängen
20  TEXT : HOME : VTAB 5: INVERSE :
    PRINT "PRODOS FORMAT.LC":
    NORMAL : PRINT
25  X = PEEK (49281):X = PEEK (49281):
    REM Bank 2 schreibfähig machen
30  INPUT "Slot    (1-7) : ";S
35  POKE 53507,S
40  INPUT "Drive   (1-2) : ";D
45  POKE 53508,D
50  INPUT "Spuren (35-80): ";T
55  POKE 53509,T
60  HOME : VTAB 10: CALL 48960:
    REM Formatieren
65  IF  PEEK (767) <  > 0 THEN  PRINT:
    PRINT "Fehler-Nr. "; PEEK (767):
    GOTO 90
70  HOME : VTAB 11: PRINT "Erneut J/N ";
75  GET X$: IF X$ = "N" OR X$ = "n"
    THEN 90
80  IF X$ = "J" OR X$ = "j" THEN 60
85  GOTO 75
90  CALL 39447: REM BASIC.SYSTEM
    wieder anhängen
```

Warnung
FORMAT.LC läuft **nur** unter ProDOS 1.01!

DISKDRIVER DEMO

```
                        ORG  $300
                *
                * DISKDRIVER.DEMO
                * ==============
                *
                * Beispiel für direkten Zugriff
                * auf ProDOS-Disk-Driver ab $F800
                *
                * Vorteil:
                * Datenpuffer darf sich
                * in den RAM-Bereichen
                * $0200-$BEFF sowie in
                * $D000-$DFFF von Bank 2
                * befinden.
                *
                * Vor JSR $F800 müssen die Para-
                * meter in die Zero-Page
                * $0042-$0047 gepokt werden.
                *
                * $0042: Command: 1=read, 2=write
                * $0043: Unitnumber: Slot/Drive
                * $0044: Low Byte Puffer
                * $0045: High Byte Puffer
                * $0046: Low Byte Blocknummer
                * $0047: High Byte Blocknummer
                *
                * Nach JSR $F800 befindet sich
                * eine mögliche Fehlernummer
                * im Akkumulator, falls das
                * Carry-Flag gesetzt ist.
                *
                *-------------------------
                *
                * Bank 2 der LC lese- und
                * schreibfähig machen
                *
0300: AD 83 C0          LDA  $C083      ;RD Bank2
0303: AD 83 C0          LDA  $C083      ;WR Bank2
                *
                * Speicherbereich $D000-$D1FF
                * mit Hex-Muster füllen = 1 Block
                *
0306: A9 01             LDA  #1         ;1
0308: A2 00             LDX  #0
030A: 9D 00 D0  EINS    STA  $D000,X
030D: E8                INX
030E: D0 FA             BNE  EINS
                *
0310: A9 02             LDA  #2         ;2
0312: A2 00             LDX  #0
0314: 9D 00 D1  ZWEI    STA  $D100,X
0317: E8                INX
0318: D0 FA             BNE  ZWEI
                *
                * Command: 01 Read, 02 Write
                *
031A: A9 02             LDA  #2         ;Write
031C: 85 42             STA  $42        ;Command
                *
                * Unit-Nummer: hier für S6, D1
                *
031E: A9 60             LDA  #%01100000 ;D1,S6
0320: 85 43             STA  $43        ;Unitno
                *
                * Puffer-Adresse: hier $D000
                *
0322: A9 00             LDA  #$00       ;$D000
0324: 85 44             STA  $44        ;Low
                *
0326: A9 D0             LDA  #$D0
0328: 85 45             STA  $45        ;High
                *
                * Block-Nummer: hier $0117 = 279
                *
032A: A9 17             LDA  #$17       ;$0117
032C: 85 46             STA  $46        ;Low
                *
032E: A9 01             LDA  #$01
0330: 85 47             STA  $47        ;High
                *
                * Disk-Driver aufrufen: $F800
                *
0332: 20 00 F8          JSR  $F800
                *
                * Language Card abstellen
                *
0335: AE 81 C0          LDX  $C081      ;RD ROM
0338: AE 81 C0          LDX  $C081      ;WR Bank 2
                *
033B: 90 03             BCC  ENDE       ;okay
                *
                * Fehler als Hex-Zahl anzeigen
                *
033D: 20 DA FD          JSR  $FDDA      ;PRBYTE
0340: 60        ENDE    RTS

65 Bytes
```

Neologismen in der Computersprache

von Ulrich Stiehl

Da die Computerwissenschaft rasant wächst, werden fast täglich neue Begriffe gebildet. Diese Neologismen sind zumeist amerikanischer Provenienz; nur selten hat man den Mut, „eigene" Wörter zu schaffen.

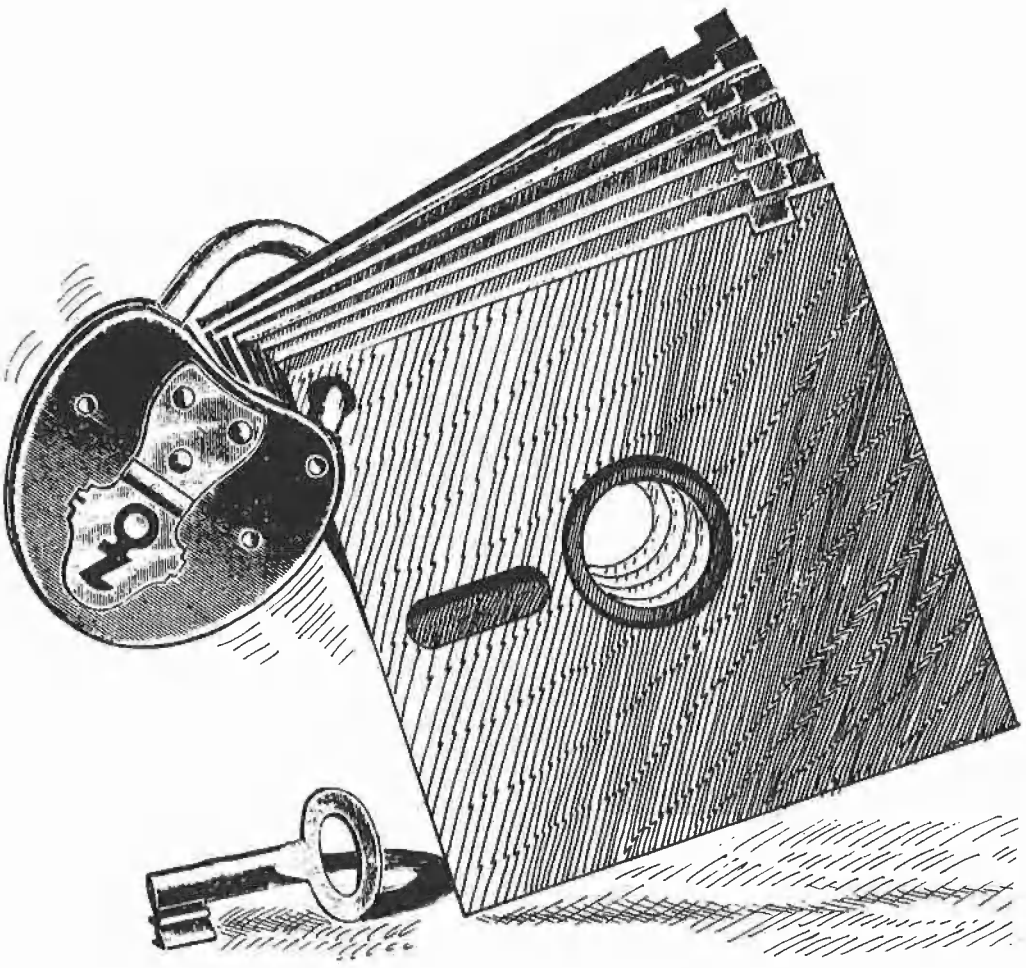
1. Etymologieprobleme
Falsche Wortbildungen können durch sorglose „wörtliche" Übersetzungen zustande kommen. Zwei Beispiele:
editieren: Erstens bedeutet „to edit" herausgeben (lateinisch edere = herausgeben: e = heraus, dare = geben) und würde damit dem deutschen „edieren" entsprechen. Zweitens bedeutet „to edit" bearbeiten, redigieren, korrigieren und müßte dann auch so übersetzt werden. „editieren" ist dagegen ein unglücklicher Zwitter und würde genausowenig wie „redaktieren" (statt richtig „redigieren") oder „korrektieren" (statt richtig „korrigieren") passen, denn – im Gegensatz zum Englischen – werden im Deutschen Verben lateinischen Ursprungs dadurch gebildet, daß man „ieren" an den Präsensstamm (und *nicht* an den Perfektstamm) anhängt (redig/o wird redigieren, corrig/o wird korrigieren, ed/o wird edieren usw.)
disassemblieren: Betrachten wir zunächst einige andere Wörter. Der Gegensatz zu „assoziieren" lautet „dissoziieren" und nicht „disassoziieren". Das Gegenstück zu „assimilieren" ist „dissimilieren" und nicht „disassimilieren". Folglich müßte es „dissemblieren" und *nicht* „disassemblieren" heißen.
„Editieren" und „disassemblieren" wird man inzwischen nicht mehr ausrotten können, da sie bereits eingeführt sind, doch würde uns Deutschen sicherlich gut anstehen, wenn wir nicht alles kritiklos übernehmen würden.

2. Orthographieprobleme
In vielen Fällen übernimmt man einfach die englischen Computerfachausdrücke unverändert ins Deutsche. Dies bringt jedoch eine Fülle orthographischer und grammatikalischer Probleme mit sich.
2.1. Divis und Großschreibung: Ein nicht zusammengesetztes Wort (z.B. Slot als lexikalisches Simplex) oder bereits im Englischen zusammengeschriebenes Wort (z.B. Softswitch) schreibt man im Deutschen stets mit großem Anfangsbuchstaben. Vielfach liegen jedoch Komposita (zusammengesetzte Wörter, hier meist Substantiv + Substantiv) vor, die im Englischen getrennt und ohne Bindestrich (Divis) geschrieben werden, z.B. „directory entry", „volume directory" usw. Der DUDEN gibt für diese Problemfälle keine verbindlichen Regeln an. Im einzelnen sind zulässig:
a) Beat generation (zweites Wort klein, ohne Divis)
b) Cash-flow (zweites Wort klein, mit Divis)
c) Cold Cream (zweites Wort groß, ohne Divis)
d) Country-Music (zweites Wort groß, mit Divis)
Man beachte: Language Card (möglich), aber Language-Card-Schreibschutz (nur so, weil letzter Bestandteil deutsches Wort ist).
2.2. Genus: Da englische Wörter kein Geschlecht kennen, steht man bei neuen Anglizismen vor der Frage, für welches Genus man sich entscheiden soll. Heißt es „der" oder „die" oder „das" Directory? Ich würde hier für „das" plädieren, da das englische „directory" auf das lateinische Neutrum „directorium" zurückgeführt werden kann. Andere Beispiele: der Catalog, weil „der Katalog"; das Volume, weil „das Volumen"; der Slot, weil „der Schlitz"; die Language-Card, weil „die Karte"; die Garbage-Collection, weil „die Kollektion"; der Controller, weil „der Kontrolleur" usw. In vielen Fällen steht man jedoch „auf dem Schlauch" und muß dann das Genus willkürlich festlegen. Ich sage beispielsweise „der File", weil französisch „le fil"; genausogut könnte man jedoch hier auch „das File" wegen lateinisch „filum" sagen.
2.3. Deklination: Wenn man das Genus erst einmal kennt bzw. festgelegt hat, kann man das Problem der Deklination in Angriff nehmen. Wie lautet z.B. bei „das Directory" der Genitiv Singular („des Directory oder des Directorys") und der Nominativ Plural („die Directories oder die Directorys")? Zunächst einige „normale Beispiele":
– *der* Whisky, des Whiskys; Pl. die Whiskys; aber: der Softy, des Softies; Pl. die Softies; ferner: der Pool, des Pools; Pl. die Pools
– *die* Party, der Party; Pl. die Partys oder die Parties
– *das* Pony, des Ponys; Pl. die Ponys; ferner das Happening, des Happenings; Pl. die Happenings; aber das Playback, des Playback; Pl. die Playbacks
Diese Beispiele zeigen, daß englische Fremdwörter im Plural (fast) immer ein „s" bzw. bei y-Wörtern oft ein „ies" haben. Im Singular findet man bei den Maskulina und Neutra häufig ein „s" im Genitiv, während die Feminina stets endungslos sind. Da die DUDEN-Grammatik keine festen Regeln vorschreibt, muß man im Zweifelsfall den Deklinationstyp „willkürlich" festlegen, was zu erheblichen Unsicherheiten führt.
2.4. Verben: Bei Verben englischer Herkunft gibt es weniger Probleme. Normalerweise hängt man an den „Stamm" (poke, peek, move) „(e)n" an (poken, peeken, moven) und konjugiert im übrigen schwach: ich poke, du pokst, er pokt; wir poken; ich habe gepokt usw. Analog: Der File wird gebloadet (oder zur Verdeutlichung geBLOADet).

3. Eindeutschungsprobleme
Wenn es darum geht, neue Wörter zu schaffen, sind die Deutschen einfach nicht mutig genug. Der Engländer nimmt ohne viel Federlesens ein bereits vorhandenes Wort und verleiht ihm eine neue Bedeutung. Beispiele hierfür sind „catalog" (eigentlich „Bücherverzeichnis"), „volume" (eigentlich „Band" eines mehrbändigen Buches), „string" (eigentlich „Kordel"), „boot" (eigentlich „Stiefel") usw. Der Engländer denkt sich nichts dabei, wenn er Wendungen wie „to home the cursor" benutzt, während wir uns totlachen würden, wenn jemand HOME mit HEIM übersetzen würde. Es wäre an der Zeit, hier etwas mehr Mut zu zeigen. In der Neuerscheinung „DOS 3.3" aus dem Franzis-Verlag ist der Autor B. Ruhland teilweise sehr mutig gewesen. Einige Beispiele (bitte nicht lachen!): Bezugsadresse (für offset address), Belegungsrichtung (für allocation direction), Steuerkarte (für controller), Einsteckschlitz (für slot). Allerdings sind auch einige Wortungetüme geprägt worden, mit denen ich mich nicht anfreunden kann, z.B. „Sektorinitialisierungsmerkbereich".

Kopierschutz für den Apple II

von Matthias Greve



Softwarehersteller arbeiten heutzutage mit einer Vielzahl ausgefeilter Kopierschutzverfahren, um die illegale Verbreitung von Raubkopien zu verhindern. Der nachfolgende Artikel gibt einen allgemeinen Einblick in die Entwicklung und die zur Zeit angewandten Schutzmethoden, die wertfrei, d.h. ohne Berücksichtigung der kulturpolitischen und urheberrechtlichen Implikationen des Schützens dargestellt werden.

Nicht-formatierte Spuren: Die ersten Kopierschutzmethoden waren sehr einfacher Natur. Eine Spur (Track) auf der Diskette, meistens Track 3, wurde nicht formatiert. Die anfänglichen Kopierprogramme duplizierten die ganze Diskette Track für Track. Durch die fehlende Formatierung trat bei Track 3 ein Lesefehler auf, und der Kopiervorgang wurde abgebrochen.

Andere Catalog-Spur: Einige Softwarehersteller verlegten zum Schutz ihrer Programme den DOS-3.3-Catalog von Track 17 auf einen anderen Track. Solche Disketten konnten nur noch mit einem modifizierten DOS gelesen werden.

Spur 35: Auch Hardware-Eigenschaften der Disk-II-Laufwerke wurden zum Schützen der Originaldisketten ausgenutzt. Außer den üblicherweise benutzten Tracks 0–34 kann der Schreib/Lesekopf meist noch auf Track 35 positioniert werden. Standard-DOS ignoriert diesen zusätzlichen Track mit allen hier abgelegten Daten. Dies war damals ein sehr effektvoller Kopierschutz, da nur wenigen Benutzern dieser technische Kniff bekannt war. Die Verwendung von Track 35 besitzt aber einen gravierenden Nachteil. Nicht alle Laufwerke können ihren Kopf exakt genug positionieren, um Track 35 einwandfrei zu lesen. Knarren, Lesefehler und nicht bootende Originale waren oft die Folge.

Andere Prüfsummenbildung: Neue Schutzverfahren mußten gefunden werden, die auf allen Rechnern und Laufwerken einwandfrei arbeiteten. Bis jetzt wurden alle Programme unter Standard-DOS-Format auf der Diskette abgespeichert. Programmierer unternahmen erste Versuche, das Betriebssystem und das verwendete Format zu ändern. Während das DOS die 256 Datenbytes eines Sektors auf die Diskette schreibt, wird aus allen Bytes eine Prüfsumme gebildet, die zusätzlich auf der Diskette gespeichert wird. Beim Lesevorgang bildet das Betriebssystem erneut diese Prüfsumme und vergleicht sie mit dem abgespeicherten Wert. Stimmen die beiden Werte nicht überein, ist der betreffende Sektor fehlerhaft gelesen worden. Zum Schützen wird in einem eigenen DOS die Art der Prüfsummenbildung geändert. Versucht der Benutzer, mit Standard-DOS auf die geschützte Diskette zuzugreifen, stellt die Prüfsummenroutine eine Abweichung fest und meldet einen Lesefehler.

Geänderte Header: Oft wurden das Diskettenformat und der Aufbau einzelner Sektoren geändert. Der Sektor einer Appleldiskette besteht aus einem Adreß- und einem Datenfeld. Adreßfelder werden vom Betriebssystem zur Orientierung auf der Diskette benötigt, da diese Informationen über die Track-, Sektor- und Volume-Nummer des betreffenden Sektors enthalten. Auf das Adreßfeld folgt das Datenfeld, in dem sich die eigentlichen Datenbytes befinden. Um eine einwandfreie Erkennung zu ermöglichen, beginnt jedes der beiden Felder mit einer Folge von reservierten Bytes, den sog. Headern. Der Header des Adreßfeldes besteht beim Standard-DOS aus 3 Bytes ($D5 $AA $96). Wenn das Betriebssystem einen Sektor lesen will, sucht es auf der Diskette zuerst nach diesen 3 Bytes, die den

Beginn eines neuen Sektors kennzeichnen. Konnte das Adreßfeld gelesen werden und handelt es sich um den gewünschten Sektor, sucht das DOS jetzt nach dem Header des Datenfeldes. Der Datenheader besteht wie der Adreßheader aus 3 Bytes ($D5 $AA $AD). Direkt auf den Datenheader folgen die codierten Datenbytes und ihre Checksum (= Prüfsumme). Zum Schützen der Diskette wird die Formatierungsroutine des DOS so geändert, daß andere Adreß- und Datenheader benutzt werden. Schreib- und Leseroutinen werden an das neue Format angepaßt. Der Vorteil dieses Schutzverfahrens liegt in der unveränderten Benutzerschnittstelle. Anwenderprogramme müssen nicht an das neue Betriebssystem angepaßt werden. Da diese Änderung im Betriebssystem relativ leicht zu bewerkstelligen ist, gehören neue Daten- und Adreßheader inzwischen zum Standardkopierschutz für Disketten. Ein neues Format allein ist jedoch noch kein zuverlässiger Schutz. Viele Nibble-Kopierprogramme können heutzutage solche Disketten ohne Probleme duplizieren.

Gerade Sektornummern: Die Adreßfelder bieten jedoch noch einen weiteren Ansatzpunkt für Schutzverfahren, nämlich die hier abgelegte Sektornummer. Standard-DOS 3.3 verwendet eine Sektornumerierung von 0–15. Es ist jedoch möglich, alle Sektornummern in gerade Zahlen zu verwandeln. Ein derart geändertes Format besitzt eine Sektornumerierung von 0–30, die ein unverändertes DOS nicht mehr verwalten kann.

Geänderte Timingbytes: Einige Softwarehersteller ersannen noch grundlegendere Formatänderungen. Die gesamten Informationen eines Tracks bestehen aus einer riesigen Folge von Bits. Sollen Daten gelesen werden, so befindet sich der Schreib/Lesekopf irgendwo auf einem der Sektoren der gewünschten Spur, wobei zunächst noch nicht bekannt ist, um welchen Sektor es sich handelt. Es ist nicht einmal sichergestellt, daß der Lesevorgang mit einem Byte beginnt. Um zu gewährleisten, daß alle acht gelesenen Datenbits auch zu demselben Byte gehören, muß das DOS den Lesevorgang mit der Kopfposition synchronisieren. Hierfür werden die „Timing-" oder „Sync-Bytes" verwendet. Diese speziellen Bytes bestehen im Gegensatz zu allen anderen Datenbytes aus zehn Bits. Die beiden niederwertigsten Bits sind jedoch auf Null gesetzt. DOS 3.3 verwendet als Timingbyte $FF (%1111111100). Nullbits zu Anfang eines Bytes sind nicht zugelassen. Deshalb startet der Controller erst mit dem Einlesen eines neuen Bytes, wenn er eine 1 als Datenbit auf der Diskette erkennt. Diese Hardware-Eigenschaft wird in Verbindung mit den Timingbytes zum Synchronisieren benutzt. Auf einem Track befinden sich mindestens fünf Timingbytes vor jedem Adreß- und Datenfeld. Startet das DOS den Lesevorgang, so erreicht der Kopf irgendwann ein Feld von Timingbytes. Da die Leseroutinen noch nicht synchronisiert sind, beginnt der Controller irgendwo innerhalb des ersten Timingbytes Datenbits von der Diskette zu lesen. Da diese Spezialbytes eine Länge von 10 Bits besitzen, jedoch nur 8 Bits gelesen werden, ist der Lesevorgang des nächsten Timingbytes genau um zwei Bits verschoben. Dieser Vorgang wiederholt sich so lange, bis eines der beiden Nullbits zu Anfang des neuen Bytes steht. Nullbits am Beginn eines Bytes werden jedoch vom Controller ignoriert. Daher ist das nächste gelesene Byte ein komplettes Timingbyte. Von da an werden die acht Einserbits der Timingbytes gelesen und die beiden folgenden Nullbits übersprungen. Der Controller und die Leseroutinen sind jetzt synchronisiert. Bei Datenbytes, die diesen Sync-Bytes folgen, ist sichergestellt, daß alle acht Bits wirklich zum gleichen Byte gehören. Sync-Bytes müssen bezüglich ihrer Bitkombination besonderen Ansprüchen genügen. Daher kommen außer $FF nur noch wenige andere Bytes in Frage. Wird nun statt $FF ein anderes Byte als Timingbyte benutzt, ist das Standard-DOS nicht mehr in der Lage, seine Leseroutine mit der Diskette zu synchronisieren. Kein einziges Byte kann von einer so geschützten Diskette gelesen werden.

Spezielle Datenbytes: Diese Eigenheit des Controllers wird auch bei anderen Schutzverfahren ausgenutzt. Spezialroutinen schreiben ein Byte innnerhalb des Datenfeldes im Format eines Timingbytes, d.h. zwei zusätzliche Nullbits am Ende des Bytes. Übernimmt der Controller dieses Byte von der Diskette, liest er die ersten acht Bits und überspringt die beiden folgenden Nullbits. Sie werden einfach ignoriert, da der Lesevorgang erst beim nächsten Einserbit wieder beginnt. Kopierprogramme übertragen dieses Spezialbyte als normales Datenbyte auf die Kopie. Die beiden Nullbits fehlen. Standardbytes werden innerhalb von 32 Maschinenzyklen auf die Diskette geschrieben. Zum Schreiben von Timingbytes benötigt das DOS auf Grund der Länge von zehn Bits 40 Zyklen. Werden normale Datenbytes gelesen, erhält die Leseroutine spätestens alle 32 Zyklen ein neues Byte von der Diskette. Ist die Leseroutine jedoch an dem Spezialbyte angekommen, tritt durch das Auslassen der zwei Nullbits eine kleine Verzögerung auf. Bei geschützten Programmen mißt ein mitlaufender Zähler die zum Lesen eines Bytes benötigte Zeit. Sollte diese Zeitverschiebung nicht auftreten, muß es sich um eine Kopie handeln.

Synchronisierte Tracks: Diese bildeten die Grundlage der nächsten Methode. Bei der Erstellung der Originaldisketten formatieren geeignete Routinen die Diskette so, daß die einzelnen Tracks eine bestimmte physikalische Anordnung zueinander besitzen, was z.B. bedeutet, daß jeweils die ersten Sektoren eines Tracks – von der Diskettenmitte aus gesehen – in einer Reihe liegen. Beim Bootvorgang liest das Programm Track 0 bis zu einer bestimmten Stelle ein und verschiebt dann den Schreib/Lesekopf auf Track 1. Ohne auf die Sektornummer zu achten, wird jetzt auf Track 1 der nächste Sektor gelesen. Waren Track 0 und Track 1 einwandfrei synchronisiert, hat die Software auf Track 1 auch den richtigen Sektor gelesen. Auf diese Weise läßt sich ein ganzes Anwenderprogramm in den Speicher laden. Synchronisiert ein Kopierprogramm beim Formatieren die Kopie nicht, läßt sich das Original zwar einwandfrei kopieren, jedoch geht die physikalische Anordnung der Tracks zueinander verloren. Diese Verschiebung der Tracks auf der Kopie im Gegensatz zum Original bewirkt, daß beim Bootvorgang falsche Sektoren gelesen werden.

Nibble-Counting: Auf einer anderen Hardware-Eigenschaft beruht das „Nibble-Counting". Da nur sehr wenige Laufwerke mit genau der gleichen Umdrehungsgeschwindigkeit arbeiten, differiert beim Formatieren die Tracklänge von Laufwerk zu Laufwerk. Nibble-Counting benutzt diese Tatsache, um Original und Kopie zu unterscheiden. Bei Erstellung der Originaldisketten werden durch nochmaliges Einlesen nach der Formatierung für jeden Track die Längen notiert und Timingbytes sowie Daten- und Adreßfelder mitgezählt. Daher auch der Name „Nibble-Counting". Die so erhaltenen Informationen werden auf der Diskette abgelegt. Während des Bootvorgangs überprüft die Software die Längen der Tracks und vergleicht sie mit den auf der Diskette gespeicherten. Dies ist ein eindeutiges Entscheidungskriterium, da es recht unwahrscheinlich ist, daß die Tracklängen von Kopie und Original übereinstimmen.
Synchronisierte Tracks und Nibble-Counting gelten heute bereits als veraltet, da viele Nibble-Kopierprogramme inzwischen in der Lage sind, diese beiden Schutzmethoden zu umgehen.

Halbspuren: Laut Handbuch besitzen die Laufwerke des Apple II 35 Spuren, auf denen der Schreib/Lesekopf positioniert werden kann. Doch die Diskettenstation ist durchaus in der Lage, 70 Spuren anzufahren. Diese Spuren – Halbspuren genannt – besitzen im Gegensatz zu den 35 Standard-Spuren nur den halben Abstand zueinander. Wie ist das möglich? In den Laufwerken befindet sich ein Schrittmotor, der zur Bewegung des Kopfschlittens dient. Um den Schlitten von einer der 35 Spuren zur nächsten zu verschieben, werden normalerweise vier Schritte ausgeführt. Werden statt dessen nur zwei ausgeführt, befindet sich der Kopf in einem Bereich zwischen zwei der 35 Spuren. (Technische Details hierzu im Peeker, 3/85, S. 10ff.)
Halbspuren können jedoch nur bedingt verwendet werden. Durch den Aufbau des Schreib/Lesekopfes ist es nicht möglich, eine Spur und die unmittelbar benachbarte Halbspur zu beschreiben. Übersprechen und Datenverlust wären die Folgen. Trotzdem ist es möglich, Halbspuren zur Datenspeicherung zu benutzen. Voraussetzung hierfür ist, daß direkt benachbarte Tracks nicht benutzt werden. Eine Möglichkeit wäre, alle Spuren

einfach um eine Halbspur nach innen zu verschieben. Die Diskette besitzt dann zwar immer noch 35 Spuren, aber normales DOS findet keine einzige mehr.

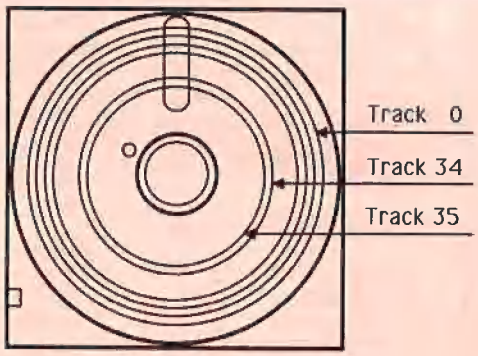
Spiraltracks: Halbspuren sind die Grundlage eines ausgefeilten Kopierschutzes, der es erlaubt, alle 70 Spuren der Disk-II-Laufwerke durch die Verwendung von Spiraltracks auszunutzen: Um Übersprechen zu vermeiden und Datensicherheit zu garantieren, muß der Abstand zwischen zwei verwendeten Spuren mindestens eine ganze Spur betragen. Zu diesem Zweck übertragen spezielle Routinen die Daten nach einem ganz bestimmten Schema auf die Originaldisketten. Bei der Erstellung des Originals wird ca. 1/3 der ersten Halbspur beschrieben und der Kopf auf die benachbarte Halbspur gefahren. Hier wird nun wieder 1/3 der Spur beschrieben. Dieser Vorgang wiederholt sich solange, bis alle Daten auf die Diskette übertragen sind. Die Halbspuren scheinen sich spiralförmig zur Diskettenmitte hin zu winden. Daher auch der Name Spiraltracks. Dieses Schutzverfahren besitzt bereits eine hohe Kopiersicherheit gegenüber modernen Nibble-Kopierprogrammen, da diese zwar Halbspuren lesen, jedoch meist nur komplette Halbspuren schreiben können. Wird eine Diskette Halbspur für Halbspur kopiert, zerstört der Kopf durch seine Breite beim Schreiben jeweils die benachbarten Halbspuren. Einige wenige Programme können nach umfangreichen Parameteränderungen oder Programmierung auf dieses Format eingestellt werden. Der Zeitaufwand für diese Änderungen ist jedoch sehr hoch. Die Implementierung dieses Schutzverfahrens ist recht aufwendig, da neue Schreib/Leseroutinen erforderlich sind und die Anwendersoftware speziell hierfür angepaßt werden muß. Die Kopiersicherheit, die Spiraltacks und ähnliche Verfahren bieten, rechtfertigt jedoch den hohen Aufwand.

Pseudo-Sektoren: Neue Diskettenformate liegen auch der Verwendung von Pseudo-Sektoren zugrunde. Ein Pseudo-Sektor besteht wie ein normaler DOS-Sektor aus einem Adreßfeld und dem darauffolgenden Datenfeld. 342 codierte Datenbytes sind in einem Standarddatenfeld abgelegt. Im Gegensatz dazu kann das Datenfeld eines Pseudo-Sektors bis zu 5K an codierten Daten enthalten. Ein Sektor dieser Länge belegt dann einen vollständigen Track. Es ist fraglich, ob sich der Aufwand für diese Art von Kopierschutz lohnt. Zum einen ist die Programmierung der neuen Schreib/Leseroutinen sehr zeitintensiv. Zum anderen stellen Pseudo-Sektoren kaum noch eine Schwierigkeit für ein gutes Nibble-Kopierprogramm dar.

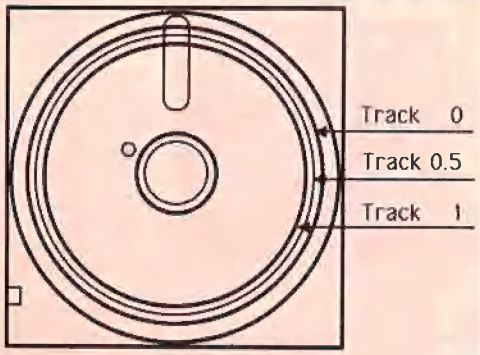
Magnetschicht-Zerstörung: Zeitweise versuchten Softwarehäuser, Kopierschutzverfahren durch Beschädigung des Originals zu realisieren. An einer bestimmten Stelle der Diskette wurde durch einen Kratzer die Magnetschicht des Datenträgers zerstört. Dadurch entstand ein Bereich auf der Diskette, der sich weder initialisieren noch beschreiben ließ. Die geschützte Software versuchte einen der zerstörten Tracks zu formatieren. Gelang dies ohne Probleme, mußte es sich folglich um eine Kopie handeln. Die zerstörten Tracks des Originals hätten nicht einwandfrei initialisiert werden können. Die zerkratzte und dadurch sehr unebene Magnetschicht der Diskette griff jedoch die empfindlichen Schreib/Leseköpfe der Laufwerke an. Laute Schabgeräusche und hohe Abnutzung der Köpfe waren die Folgen. Grund genug, dieses Schutzverfahren wieder aufzugeben.
Der Trick mit der Unformatierbarkeit mancher Teile der Originaldiskette wurde jedoch in einem anderen Kopierschutzsystem wieder aufgegriffen. Vor der Erstellung des Originals magnetisiert eine spezielle Hardware einige Tracks der verwendeten Diskette so stark, daß alle Formatierungsversuche mit dem schwachen Magnetfeld des Schreib/Lesekopfes erfolglos bleiben. Geschützte Programme versuchen diese Spuren auf der Diskette, von der sie gestartet wurden, zu formatieren. Ein Schreib- oder Lesefehler bei diesem Vorgang bestätigt, daß es sich um das Original handelt.

Protection-Keys: Einen ganz anderen Weg gehen manche Softwarehersteller, indem sie sog. „Protection-Keys" verwenden. Bei diesen Keys handelt es sich um spezielle Bausteine, in denen eine bestimmte Information fest „verdrahtet" ist. Der Key muß in den Game-I/O-Port des Apple II gesteckt werden, damit das Programm diese Information abfragen kann. Mit jedem Originalprogramm wird genau *ein* Key geliefert. Damit ist die Frage nach der Kopierbarkeit der Software hinfällig. Das Programm läuft nur mit diesem einen Key oder gar nicht. Oft sind diese Bausteine in ein Gehäuse eingeschweißt oder mit Epoxydharz ausgegossen. Protection-Keys sind durch ihren Aufbau eine teure, aber auch sichere Kopierschutzmethode. Das Hauptanwendungsgebiet für diese Art des Schutzes liegt bei kommerzieller Software, die in geringeren Stückzahlen zu höheren Preisen produziert wird.

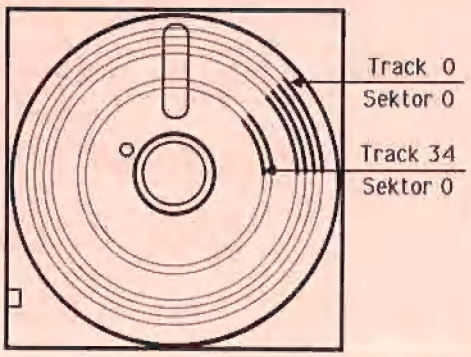
Physikalische Lage von Track 35



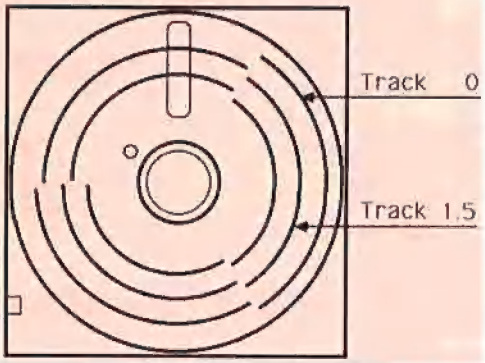
Position der Halbspuren



Synchronisierte Tracks



Spiralförmige Tracks



Peeker-Sammeldisketten

Einzelbezug DM 28,–
Fortsetzungsbezug DM 20,–
(Jederzeit kündbar, jedoch mindestens 6 Disketten)
(* = nur auf Diskette, nicht im Peeker gelistet! Seitenangaben beziehen sich auf Beginn des Listings)
Hüthig Software Service
Postfach 10 28 69 · 6900 Heidelberg 1

Disk# 1
(Heft 1+2, 1984)

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

T.ACCEL.WAIT (1/84, S. 22)
ACCEL.WAIT
T.ACCEL.BOOT
ACCEL.BOOT
ACCEL.LC.KOPIERER
T.ACCEL.LC.KOPIE
ACCEL.LC.KOPIE
T.ACCEL.ROM.KOPIE1
ACCEL.ROM.KOPIE1
T.ACCEL.ROM.KOPIE2
ACCEL.ROM.KOPIE2

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

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

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

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

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

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

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

PRIMZAHLEN.OVERMEYER * (2/84, S. 70)
PRIM.OBJ0 *
PRIM.OBJ1 *
PRIM.TEST *
PRIM.TOOLKIT.SOURCE *

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

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

T.IBS.RAMDISKDRIVER (1-2/85, S. 20)
IBS.RAMDISKDRIVER
T.AP20.RAMDISKTEST
AP20.RAMDISKTEST

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

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

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

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

T.VIDEXT
VIDEXT.LISA.SOURCE
VIDEXT

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

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

Disk #3
(Heft 1-2, 1985, CP/M-Format)

STEUER.84 (1-2/85, S. 47)
PASS.BAS
MENUE.BAS
HELP.BAS *
A.BAS
B.BAS
C.BAS
D.BAS
E.BAS
F.BAS
G.BAS
H.BAS
I.BAS
J.BAS
K.BAS
L.BAS
M.BAS
N.BAS

Disk #4
(Heft 3-4, 1985)

TESTGENERATOR (3/85, S. 26)
SAETZE
BAHNFAHRT *
ZU *
TUN.UND.SOLLEN *
IRGEND *

MULTIPRECISION (3/85, S. 32)

T.WS.TRANSFER (3/85, S. 36)
WS.TRANSFER
T.WS.TRANSFER.2 *
WS.TRANSFER.2 *
GETCPM

PRIM.0.SC.SOURCE (3/85, S. 62)
PRIM.0.BIN
PRIM.1.SC.SOURCE
PRIM.1.BIN
PRIM.FP

ACCELERATOR.ABSTELLEN (3/85, S. 66)

T.WILDCARD.TEST * (3/85, S. 72)
WILDCARD.TEST1 *
T.WILDCARD.TEST2 *
WILDCARD.TEST2 *

XPLOT.DEMO (4/85, S. 18)
XPLOT.ROUTINE
T.XPLOT.ROUTINE

MENUE.GENERATOR (4/85, S. 22)

T.MACROS.65C02 (4/85, S. 31)

TERMINAL (4/85, S.36)
TERMINAL.B
T.TERMINAL.B

CAT.ARRAY (4/85, S. 44)
CAT.SAVER
EINTRAG.SUCHER
EINTRAG.ANALYSE
PRODOS.READER
T.PRODOS.READER.OBJ
PRODOS.READER.OBJ

MOUSESTUFF.PASCAL.SOURCE (4/85, S. 51)
MOUSE.ASS.PASCAL.SOURCE
TESTMOUSE.PASCAL.SOURCE
DRAWMOUSE.PASCAL.SOURCE

INALL.DATA (4/85, S. 70)
SCREEN80.DATA (4/85, S. 33)
SCREEN80.SAVER (4/85, S. 76)

Disk #5
(Heft 5, 1985, DOS-Format)

T.FM.BSP (5/85, S. 9)
FM.BSP

T.SLOTRAMDISK (5/85, S. 13)
SLOTRAMDISK
SLOTRAMDISK.HELLO

PLOT.2.0 (5/85, S. 20)
T.PLOT.B
PLOT.B
PLOT.PROTECTOR

T.CONVERT560 (5/85, S. 26)
CONVERT560
CONVERT560.DEMO

T.EDA (5/85, S. 33)
EDA

TRANSCEND.PASCAL.SOURCE (5/85, S. 36)

T.BLOCKTRACER (5/85, S. 51)
BLOCKTRACER
T.BLOCKTRACER1
BLOCKTRACER1

FORMAT.LC (5/85, S. 56)
FORMAT.LC.START
T.DISKDRIVER.DEMO
DISKDRIVER.DEMO

RANDOM.DEMO (5/85, S. 69)
COLUMN80.DEMO

SUPERDUMP.EPSON (6/85!)
SUPERDUMP.IMAGEWRITER
SUPERDUMP.BILD
T.SUPERDUMP
SUPERDUMP
EPSON
IMAGEWRITER

Der nächste Peeker
Heft 6/1985
erscheint am
20.5.1985

Apple-Kompatible

von Hans-Kurt Kuhn

Ananas, Banana, Boscop oder weniger botanisch und etwas weniger an Apple erinnernd Atlas, Lazar, Pluto oder manchmal auch schlicht Computer – so heißen einige Beispiele aus der „Familie" der Apple-Nachbauten oder vornehmer der „Apple-Kompatiblen". Kompatibel wird dabei in dem Sinne gebraucht, daß auf Original-Apple-II-(Plus)-Geräten verwendbare Software hier ohne Änderungen lauffähig ist.

Wie Kompatible entstehen

Als Steven Jobs und Stephen Wozniak 1976 der große Wurf gelang, mit gängigen Mikrochips und einem handelsüblichen Prozessor auf einer Platine von kaum mehr als DIN-A4-Größe einen universell erweiterbaren Personalcomputer namens Apple II zu realisieren, war ihnen wohl nicht bewußt, daß dieser genial-einfache Aufbau einen hohen Tribut fordern würde: Wenn jemand die Kosten der verwendeten Bausteine zusammenrechnet, kommt er sehr leicht darauf, daß sich ein solcher Apple II für sehr viel weniger Geld nachbauen läßt, als Apple für das Original benötigt. Man fotografiere also eine Original-Apple-Platine (den Apple-Firmenaufdruck kann man dabei abdekken), stelle auf die heute jedem Elektronikfan geläufige Art die neue = alte Platine her, bestücke diese nach Vorbild, baue eventuell auch noch das Apple-Netzteil nach und setze das Ganze in eine auf der Abkantbank hergestellte Aluminiumkiste mit Fernosttastatur – und fertig ist der Urvater des „Apple-Kompatiblen", also eines in elektronischer Hinsicht sklavischen Original-Nachbaus.
Offensichtlich haben auch bei uns in Deutschland einige in der Elektronikszene beheimatete (Neu-) Unternehmer diese Gelegenheit ergriffen, denn bei etlichen der heute erhältlichen „Kompatiblen" handelt es sich noch um solche nachempfundene Apple II, deren rein elektronische Nachgestaltung wohl zulässig ist.

F8-ROM

Kritisch wird es bei der in PROMs oder EPROMs untergebrachten Firmware aus dem Hause Apple bzw. Microsoft (das F8-ROM ist von Apple, der Applesoft-Interpreter von Microsoft), da hier möglicherweise das Urheberrecht tangiert wird: So bieten die Hersteller der „Kompatiblen" ihr Produkt meist ausdrücklich „mit leeren EPROMs, ohne Firmware" an. Dem zum „Kompatiblen" fest Entschlossenen ist auch dies kein Hindernis, sofern er über einen Bekannten mit Original-Apple-II und EPROM-Brenner sowie Kenntnissen in der Maschinenprogrammierung verfügt. Die erforderlichen Speicheradressen verrät freundlicherweise das Apple-Handbuch, das auch ein vollständiges Assemblerlisting des Autostart-F8-ROMs enthält.

Sofern es sich also um einen solchen Kompatiblen handelt: Was sollte an ihm schlechter sein als an einem Original? Höchstens die nach einiger Zeit zum Prellen neigende Billigtastatur. Aber auch hier schafft der Import Abhilfe: Seit einiger Zeit sind Originaltastaturen bei uns erhältlich, und das so preiswert, daß sich die Verwendung anderer Tastaturen kaum noch lohnt.

Die Versuche, selbstverfaßte Firmware in den EPROMs abzulegen, scheinen bei der Käuferschaft auf nicht sehr viel Gegenliebe zu stoßen, da sie dann z.B. auf das liebgewordene selbststartende F8-ROM oder gar auf Applesoft-Basic verzichten und sich mit FORTH oder ähnlichem bescheiden müssen. Hier tut sich ein Feld für Assemblerprogrammierer und Hardwarekenner auf, die das F8-ROM so umschreiben, daß es selbststartend wird, ohne als Autostart-ROM-Kopie gelten zu müssen.

Disk-Controller

Wenden wir uns den Peripheriegeräten und Erweiterungen zu: Die von Apple verwendeten Shugart-Laufwerke sind bis auf die Ansteuerungselektronik des Schreib/Lesekopfes als preiswerte Import-Allzwecklaufwerke erhältlich und brauchen lediglich mit der richtigen Steuerung, dem Analogboard, ausgerüstet zu werden. Diese Analogboards wurden anfangs auf die gleiche Weise wie die Motherboards kopiert. Mithin sind diese Laufwerke, bis auf den Namen, Apple-Drives. Anders verhält es sich bei den Japan-Importen von Billig-Drives, meist als Slimline-Typen erhältlich: Hier ist das Analogboard in die Antriebselektronik integriert, wie auch die Fernost-Importe der Disk-Drive-Controller anscheinend Eigenentwicklungen sind. Dies gilt auch für die importierten Zusatzkarten, also 16K-Erweiterungen, 80-Zeichenkarten, Printer-Interfaces und Z80-Karten, wobei allerdings einige nicht ganz geklärte Inkompatibilitäten auftreten: So funktioniert z.B. der in der ersten Peeker-Ausgabe geschilderte ProDOS-Patch ab Adresse $2659: EA EA, der die Namensprüfroutine lahmlegen soll, in Verbindung mit einem Original-Apple-F8-ROM (!) auf meinem Kompatiblen durchschlagend: Derart gepatchtes Original-ProDOS steigt aus, während es auf dem Apple II läuft! (Es mußte $265B heißen und

wurde im Peeker 2/84, S. 10 korrigiert. Anm.d.Red.)

80-Zeichenkarte
Ein Problem ganz besonderer Art hinsichtlich der Kompatibilität wirft die 80-Zeichenkarte auf: Sie wird fast ausnahmslos als videxkompatibel angepriesen, ist es aber in den seltensten Fällen.
Nach meiner Zählung gibt es mindestens vier Versionen:
– Die erste Karte hat keinen Softswitch, so daß für die Grafik das Ausgangskabel umgesteckt werden muß.
– Die zweite Version besitzt einen sog. Softswitch, ist aber deswegen nicht zur Videxkarte mit Softswitch kompatibel, weil sie nicht durch die Befehlsfolge Ctrl-Z,1, sondern nur durch Reset bzw. durch PR#0: IN#0: HGR in den 40-Zeichenmodus zurückschaltet, was sehr seltsam ist. Außerdem verwendet sie einen anderen Adreßbereich zur Bildschirmspeicherung, der die Programmierung von Hardcopy-Routinen erschwert. Hinzu kommt, daß durch eine Taktfrequenz in der Nähe des Motherboardquarzes Störungen erzeugt werden, die sich als schwingende Bildränder bemerkbar machen.
– In der dritten Fassung ist die Taktfrequenz auf 18 MHz erhöht; der Speicherbereich ist noch immer verlegt mit der Folge, daß in Pascal die Funktion „Keypress" nicht funktioniert, und auch Ctrl-Z,1 geht noch nicht.
– In der mittlerweile erhältlichen vierten Version sollen nun auch diese Nachteile behoben sein; Genaueres kann ich im Moment noch nicht sagen, da ich bei dem Versuch, eine solche Karte zu Testzwecken zu erwerben, mit drei Karten aus drei Quellen beliefert worden bin, die eine Gemeinsamkeit haben: Sie sind sämtlich defekt!
Die vorangehenden Schilderungen zeigen, daß der 80-Zeichenkarte bei der Anschaffung eines Kompatiblen besondere Aufmerksamkeit zu widmen ist, da sie einen besonders neuralgischen Punkt darstellt.

Disk-Drives
Um gleich bei unliebsamen Erfahrungen zu bleiben: Eine weitere Schwachstelle sind die Billigdrives älterer Bauart. Die Laufwerke des Typs FD 525-A von Shonan Data (von der Firma Lech-Technics früher beim Atlas 1 eingesetzt) in Zusammenhang mit den alten Eigenbau-Controllern dieser Firma machen die Spitze der Reparaturen aus (der mir zugängliche Erfahrungskreis umfaßt 10 Geräte). Im Gegensatz dazu lassen die Geräte mit Shugart-Drives oder die neueren Fernost-Importe mit Zentrierknebel hinsichtlich Zuverlässigkeit keine Wünsche offen. So hat obige Firma ihr Folgemodell Atlas 1 mit Slimlines des Typs FD-LDD 103 von Resco ausgerüstet; dieses von mir getestete Gerät arbeitet seit vier Monaten ebenso einwandfrei wie der mir zugängliche Kompatible Circle 100 mit Distar-Slimlines oder dessen größerer Bruder Nova II mit einem TEAC- und einem Siemens-Laufwerk, beide letzteren vertrieben von der Firma Schmitz Datentechnik in Geilenkirchen.

Netzteil
Bei den meisten Kompatiblen, die heute erhältlich sind, werden Hardwareverbesserungen durchgeführt, die über das Original hinausgehen:
Das im Original-Apple-II wie in älteren Nachbauten als Ofen wirkende längsgeregelte Netzteil, das in Verbindung mit sommerlichen Temperaturen häufig wegen der gleich benachbarten 16K-Karte eine Quelle von unerklärlichen „Unregelmäßigkeiten" darstellt (Hitzestau), ist normalerweise durch ein kaltes Taktnetzteil ersetzt.

Motherboard-Änderungen
Ein Versuch, die mechanisch anfälligen Slots bzw. Kartenanschlüsse auszumerzen, wird in neueren Kompatiblen in der Form unternommen, daß der Z80-Prozessor mitsamt Zubehör und der komplette Disk-Controller sowie die 16K-Erweiterung mit auf das Motherboard übernommen werden. Da der Z80 den Platz der EPROMs auf dem Board übernimmt, werden diese bedauerlicherweise auf einer Einsteckarte in Slot 0 angeschlossen, was diese Lösung teilweise wieder aufhebt.
Die wohl modernste Variante des Apple-Kompatiblen ist irgendeine der obigen Motherboardversionen mitsamt Controller in einem IBM-ähnlichen Gehäuse mit externer Speichertastatur und fest gespeicherten Basic-, DOS- und CP/M-Befehlen (s.o. Atlas 1 und Nova II). Ein so modernisierter Apple oder Kompatibler stellt zusammen mit der verfügbaren, qualitativ hochwertigen Software, insbesondere den Compilern zu allen gängigen Sprachen, eine Anlage dar, die es sich auch auf längere Sicht nicht zu ersetzen lohnt, zumal sich die Preise für eine ähnlich universelle Verbesserung in schwindelnden Höhen bewegen.

Reparatur-Service
Zum Abschluß sei daran erinnert, daß ein Kompatibler auch gelegentlich den Geist aufgibt, sogar weit öfter als das Original, wenn er von einem Billiganbieter mit notwendig hohem Umsatz stammt, der keine besondere Sorgfalt auf eine Endkontrolle legt. Da man mit dem Kompatiblen nicht zum Apple-Händler um die Ecke gehen kann, ist es gerade hier besonders wichtig, den Verkäufer und Reparaturdienst in nächster Nähe zu haben, sich vor dem Kauf von seiner Seriosität und Zugänglichkeit zu den eigenen Problemen zu überzeugen und so kurze Reparaturzeiten sicherzustellen, damit möglichst alle auftretenden Fehler schon in der Gewährleistungsfrist behoben werden.

Versandgeschäft
Unter den im Versandgeschäft tätigen Händlern von Kompatiblen sind einige schwarze Schafe zu finden, die nach dem Motto versenden: „Ob mein Gerät noch Fehler hat, wird der Kunde schon herausfinden." Der hat dann allerdings Mühe, bei vierwöchigen (und längeren) Reparaturzeiten alle Fehler innerhalb der ersten sechs Monate nach Kauf beheben zu lassen. Wenn man nicht selbst betroffen ist, so ist die Lektüre der Hilfe- und Entrüstungsschreie in der AUGE-Zeitschrift zu diesem Thema mitunter sehr erheiternd, für den Betroffenen allerdings eine teuer bezahlte Erfahrung.
Die in letzter Zeit zunehmende Konkurrenz unter den Händlern von Kompatiblen wird wohl ein entsprechendes Verhalten begünstigen und das Geschäft beleben, so wie die Konkurrenz der Kompatiblen in der Vergangenheit mit dazu beigetragen hat, daß die Äpfel (-Preise) nicht in den Himmel geschossen sind.

Redaktioneller Nachtrag
Der obige Beitrag deckt sich nicht in jeder Hinsicht mit der Meinung der Peeker-Redaktion. Das Thema Kompatible läßt sich m. E. auf 3 Punkte reduzieren:

Kompatibilitätsbegriff: Der Begriff der Kompatiblität ist durch die neueren Modelle IIe und IIc aufgeweicht worden, weil Apple II Plus, IIe und IIc untereinander nicht mehr 100% kompatibel sind. Beispielsweise läuft der Applewriter IIe weder auf dem II Plus noch dem IIc und ist somit weder aufwärts- noch abwärtskompatibel. Zieht man Zusatzkarten in die Betrachtung mit ein (z.B. 80-Zeichenkarten), so ist das Ende der Kompatibilität sehr schnell erreicht. Beispielsweise läuft der Applewriter IIe nicht mit der Ultraterm, und in den IIc passen mangels Steckplätzen ohnehin keine Zusatzkarten. Ein Apple-Kompatibler ist deshalb bestenfalls ein Apple-II- oder Apple-II-Plus-Kompatibler (obgleich

ich inzwischen auch einen Apple-IIe-Kompatiblen gesehen habe).

Rechtslage: „Abkupfern" galt schon immer als verpönt oder zumindest als wenig kreativ. Insofern werden Kompatible moralisch stets einen schweren Stand haben. Demgegenüber ist die juristische Situation in der Bundesrepublik nicht immer eindeutig, weil es zwei prinzipielle Typen von Kompatiblen gibt, nämlich erstens die „Nur-Kompatiblen" als exakte Kopien von Gehäuse, Platine und ROMs sowie zweitens die „Auch-Kompatiblen", auf denen man zwar *auch* Apple-II-Plus-Programme laufen lassen kann, die jedoch zahlreiche zusätzliche Features aufweisen (z.B. Prometric). Die Nur-Kompatiblen sind wahrscheinlich unzulässig, während die Auch-Kompatiblen insbesondere wegen abweichenden Gehäuses, neuer Platine und anderer ROMs wohl unstrittig sind.

Preissituation: Das Original wird einer Kopie stets vorgezogen. Wohl jeder würde Butter kaufen, wenn die Margarine nicht billiger wäre. Mancher Kompatibler (aus der Kategorie der Nur-Kompatiblen) kostet nur ein Drittel oder gar ein Viertel dessen, was heute der Apple II Plus kosten würde. Wer einen derart preiswerten Nur-Kompatiblen erwirbt, darf sich nicht wundern, wenn erstens die Verarbeitung weniger gut ist und zweitens der Service teils zu Wünschen übrigläßt. Irgendwo muß notwendigerweise gespart worden sein. Trotzdem bin ich felsenfest davon überzeugt, daß es kaum Nur-Kompatible geben würde, wenn die Preiskluft zwischen Original und Nachbau nicht derart eklatant wäre. Hier bewahrheitet sich die alte Werbeweisheit „Es war schon immer etwas teurer, einen besonderen Geschmack zu haben."

us

Hotline

von Ulrich Stiehl

Aufgrund der in den USA üblichen Gepflogenheiten von Computerfirmen und -clubs sowie durch Veröffentlichungen in anderen deutschen Zeitschriften und nicht zuletzt aufgrund eigener Testanrufe nahm ich irrtümlicherweise an, daß auch *Endabnehmer* (= Nicht-Wiederverkäufer) die Hotline der Firma Apple in München schriftlich oder telefonisch in Anspruch nehmen dürften. Dies ist jedoch nicht der Fall, wie mir Apple-München in einem persönlichen Gespräch mitteilte, auch wenn – wie aus den nachfolgenden Zuschriften ersichtlich ist – aus Kulanzgründen gelegentlich Anfragen von Privatpersonen beantwortet wurden. Insoweit bitte ich die Irreführung unserer Leser zu entschuldigen und stelle im Einvernehmen mit der Firma Apple folgendes klar:

Endabnehmer sollten die Firma Apple *nicht* direkt anrufen, sondern sich vielmehr indirekt an einen der ca. 300 lizenzierten *Händler* (= Vertragshändler) wenden, die über das ganze Bundesgebiet verteilt sind. Welcher Händler lizenziert ist, können Sie einem regelmäßig aktualisierten Händlerverzeichnis entnehmen, das beispielsweise am Apple-Stand auf den Computer-Messen erhältlich ist. Die in einschlägigen Apple-Anzeigen veröffentlichte Telefonnummer („Apple Aktuell" usw.), auf die sich eine Vielzahl unserer Leserzuschriften bezog, ist nicht im Sinne einer technischen, telefonischen Auskunft zu verstehen. Der Ausdruck „irgendwelche" in dem Satz „Falls Sie irgendwelche Fragen haben, Apple Aktuell hat ein Ohr für Sie" ist restriktiv im Sinne „irgendwelcher nicht-technischer" und und nicht im Sinne „irgendwelcher technischer" Fragen zu interpretieren.

Zusammenfassend gilt also: Wenn Sie eine Fachfrage haben, so wenden Sie sich an denjenigen Vertragshändler, bei dem Sie den Apple kaufen wollen oder gekauft haben. Sollte der Vertragshändler keine Antwort wissen, so darf er und *nur er allein* sich an die ausschließlich für Händler gedachte Stelle von Apple-München (= Händler-Hotline) wenden.

Im übrigen sollten nur solche Fragen gestellt werden, die sich auf *Original*-Apple-Hardware und -Software beziehen. Dies ist verständlich, denn weder die Apple-Händler noch die Firma Apple sind für Fremdprodukte zuständig. Ferner muß betont werden, daß man keinen Rechtsanspruch auf die Beantwortung von Fachfragen hat. Dies läßt sich schon daraus ableiten, daß Original-Appleprodukte meist (bzw. bei den mir vorliegenden Produkten immer) mit einem „Ausschluß jeder Garantie und Haftung" (Überschrift im Klappentext) verkauft werden. So heißt es beispielsweise im „Benutzer-Handbuch" zum Apple IIc: „Apple Computer gibt keine Garantien...in bezug auf die in diesem Handbuch beschriebene Software, ihre Qualität, Durchführbarkeit, Verkäuflichkeit oder Verwendbarkeit für einen bestimmten Zweck... Das Risiko für die Qualität und Durchführbarkeit liegt ganz beim Käufer. Sollten sich die Programme nach ihrem Kauf als fehlerhaft herausstellen, trägt der Käufer (und nicht Apple-Computer, deren Distributor oder Einzelhändler) die gesamten Kosten für alle notwendigen Arbeiten, Reparaturen oder Korrekturen und jedwede daraus gefolgten oder daraus resultierenden Schäden." Die Lösung technischer Probleme durch den Apple-Händler ist damit eine Frage des Wollens und nicht des Müssens.

Problematischer wird es, wenn Sie bei einem Apple-Händler ein Nicht-Apple-Produkt, z.B. einen Erphi-Controller als Fremdhardware oder Visicalc als Fremdsoftware, erwerben. Da das Nicht-Apple-Produkt in Verbindung mit dem Apple-Produkt verwendet wird, können auftretende technische Probleme an der Interaktion zwischen beiden liegen. Ein einfaches Beispiel: In einer anderen Redaktion unseres Verlages steht ein Original-Apple-II-Plus mit Original-Disk-II-Drives der Firma Apple, aber mit einer Nicht-Original-Language-Card, alles zusammen erworben bei der Firma Süß in Ludwigshafen, die damals noch Vertragshändler war. Auf dieser Konfiguration läßt sich das Original-ProDOS nicht booten. Mit dem *Nachbau*-Patch aus Peeker 1-2, 85 ließ sich dann ProDOS auf dem *Original*-Apple starten! In solchen Fällen sollte der Händler, bei dem Sie das Fremdprodukt erwerben, am besten vor dem Kauf in die Pflicht genommen werden, da sich nachträglich keiner mehr zuständig fühlt. Darüber hinaus beachten Sie, daß Fremdproduzenten gele-

```
                    1             D0   Ø
                    2             ORG  $254C
                    3   H254C    TAY             ;Alte
                    4             LDA  ($28),Y    ;Routine
                    5   H254F    EOR  #$8Ø
                    6   H2551    STA  ($28),Y
                    7             RTS
                    8   *
                    9   * Applewriter IIe Patch für IIc
                    1Ø  * a) Bei $524F Sprung zum Patch
                    11  * b) Bei $52FØ Patch-Routine
                    12  *
                    13            ORG  $524F      ;Sprung zum
                    14            JMP  H52FØ      ;Patch
                    15            FIN
                    16  *
                    17            ORG  $52FØ
52FØ: CD ØA 53      18  H52FØ     CMP  CURSOR     ;Neue
52F3: DØ Ø6         19            BNE  NORMAL     ;Routine
52F5: AD Ø9 53      2Ø  INVERSE   LDA  CHAR
52F8: 91 28         21            STA  ($28),Y
52FA: 6Ø            22            RTS
52FB: 8D Ø9 53      23  NORMAL    STA  CHAR
52FE: AD ØA 53      24            LDA  CURSOR
53Ø1: 91 28         25            STA  ($28),Y
53Ø3: AD Ø9 53      26            LDA  CHAR
53Ø6: 49 8Ø         27            EOR  #$8Ø
53Ø8: 6Ø            28            RTS
53Ø9: ØØ            29  CHAR      HEX  ØØ
53ØA: 2Ø            3Ø  CURSOR    HEX  2Ø         ;definierbar
```

Den Applewriter-IIe-Patch für den IIc bin ich Ihnen noch aus Peeker 3/85, S. 35 schuldig. Während man die inversen Menüleisten mit einem Diskeditor ändern kann, muß die Cursorroutine wegen des Mauszeichensatzes umgeschrieben werden. Dies ist nicht schwer. Problematisch ist lediglich, wo man diese Routine hinlegt. Bei $52FØ ist noch Platz im Falle der Version OBJ.APWRTG2E. U.Stiehl

gentlich echte Garantien abgeben. So heißt es beispielsweise in den „Gewährleistungs-Bedingungen" der Firma Microsoft zu dem Microsoft-Basic für den Macintosh: „...Für die auf der Diskette gespeicherten Informationen gewährleistet Microsoft ihre Übereinstimmung mit den veröffentlichten und bei Abnahme des Programms gültigen Programm-Spezifikationen. ...Wird ein die Gebrauchs-Tauglichkeit des Microsoft-Produktes im Vergleich zur jeweils gültigen Programm-Spezifikation bzw. Leistungsbeschreibung und Bedienungsanleitung wesentlich einschränkender Mangel innerhalb dieser Frist (d.h. 60 Tage nach Eingang des mangelhaften Produktes) nicht behoben, so kann der Käufer nach seiner Wahl die Herabsetzung des Kaufpreises oder die Rückerstattung des Kaufpreises gegen Rückgabe des mangelhaften Microsoft-Produktes fordern." Hier ist also die Beseitigung wesentlicher technischer Mängel nicht eine Frage der Kulanz, sondern der Gewährleistungspflicht.
Aus der Fülle der Zuschriften zum Thema Hotline greifen wir nur diejenigen Briefe heraus, zu denen wir unmittelbare Problemlösungen anbieten können.

Apple-IIc-Dienstprogramme

Ich möchte Ihrem Aufruf folgen und meine Erfahrungen mit der Apple-Hotline kundtun. Im September des letzten Jahres habe ich mir einen Apple IIc zugelegt und fing an, mich damit zu beschäftigen. Natürlich hatte ich einige Fragen und ging damit zuerst zu meinem Applehändler:
– Wie programmiere ich die Maus in Assembler?
– Wie konfiguriere ich die seriellen Schnittstellen ohne Systemdienstprogramme?
– Wie benutze ich die Maus unter DOS 3.3 zusammen mit der 80-Zeichenkarte?
Der Händler bekam eine Liste der Fragen von mir und versprach mir, die Antworten zu besorgen. Nach drei Wochen erhielt ich die Nachricht, daß er bei Apple keine Antworten auf diese Fragen erhalten habe. Inzwischen habe ich mich mit den Büchern von Ulrich Stiehl beholfen und lasse die Maus Maus sein.
Einen Tip kann ich anderen Apple-IIc-Besitzern noch geben: Wenn man nur ein Laufwerk hat, dann sollte man zum Kopieren einzelner Files unter ProDOS den FILER und unter DOS 3.3 FID zusammen mit einem RAM-Disk-Driver benutzen. Die Apple-Systemdienstprogramme booten aus dem ProDOS-Rebootprogramm einschließlich PRODOS in ca. 31 Sekunden (ab Return). Der FILER benötigt als Systemprogrammm mit PRODOS ca. 10 Sekunden, also ein Drittel der Zeit. *Ulrich Kußmann, Bonn*
(Anm.: Ein schnelles PRODOS-.FID erscheint in einem er nächsten Peeker-Hefte. Aufgrund einer flüchtigen Analyse kann man die zeitaufwendige Konfigurierung der Schnittstellen offenbar umgehen, indem man die Routinen und die „Screenhole"-Parameter direkt in den Bereich $0000-$1FFF der 64K-Karte schiebt. Eine spezielle Utility ist in Vorbereitung. Einstweilen kann man sich mit dem Programm AUXMOVER der „MMU 2.0"-Diskette behelfen. us)

Imagewriter-Sonderzeichen

Ein Freund von mir besitzt, genau wie ich, einen Imagewriter. Jetzt soll dieser ja (laut Handbuch) einen frei definierbaren Zeichensatz von ca. 170 Zeichen besitzen. Also probiert man die Steuersequenzen aus dem Handbuch aus und ist deprimiert, weil nichts läuft. Mein Freund erkundigte sich beim Händler, woran das liegt, worauf dieser antwortete, man könne nur ca. 16 Zeichen definieren. Nächster Anruf: Hotline. Antwort: Man weiß es auch nicht genau, aber irgendeine Privatfirma habe für DM 198,- ein Programm herausgebracht, mit dem man wenigstens 96 Zeichen definieren kann. Im übrigen schickte der Hotline-Herr noch ein Programm für die Hires-Grafik des Imagewriter. *Gerhard Schneider, Unterleinleiter*
(Anm.: Für den Imagewriter gibt es u.a. von der Firma Hunstig, Münster, ein Programm zur Erzeugung von Sonderzeichen. Es heißt DMP-Charger. Die Anzahl der zulässigen Zeichen hängt vom Speicherplatz im Druckerpuffer = Zeichenpuffer ab. Aufwendige Zeichensätze wie z.B. der neue Fraktur-Zeichensatz der Firma Hunstig benötigen mehr Speicherraum (1 Frakturzeichen = 2 normale Zeichen), womit dann insgesamt weniger Zeichen definiert werden können. us)

RANDOM.DEMO

```
10Ø REM Random-Zahlen-Generator
11Ø REM von T.Hare, J.Russ, G.Faulkner, David Sparks
12Ø REM Zitiert aus 'Call Apple', 1/1983, S. 3Øff.
13Ø REM Gestrafft und ab $Ø3ØØ neu assembliert von
       U.Stiehl
14Ø REM a) Zufallszahl initialisieren mit CALL 768
15Ø REM b) Neue Zufallszahl mit R = USR (B) abrufen
16Ø REM c) Wenn B = 1, dann R = Fließkommazahl
           im Bereich Ø bis 1
17Ø REM d) Wenn B > 1, dann R = Ganzzahl
           im Bereich Ø bis B-1
18Ø DATA 169,76,133,1Ø,169,27,133,11,169,3
19Ø DATA 133,12,16Ø,4,185,2Ø1,Ø,57,198,3
2ØØ DATA 153,194,3,136,2Ø8,244,96,165,157,24Ø
21Ø DATA 8Ø,165,162,16,3,76,153,225,16Ø,3
22Ø DATA 162,2Ø3,32,43,235,162,243,14,195,3
23Ø DATA 46,196,3,46,197,3,46,198,3,56
24Ø DATA 173,195,3,233,155,72,173,196,3,233
25Ø DATA 219,72,173,197,3,233,255,72,173,198
26Ø DATA 3,233,3,72,144,18,1Ø4,141,198,3
27Ø DATA 1Ø4,141,197,3,1Ø4,141,196,3,1Ø4,141
28Ø DATA 195,3,176,4,1Ø4,1Ø4,1Ø4,1Ø4,232,2Ø8
29Ø DATA 192,169,3,133,158,169,255,133,159,169
3ØØ DATA 219,133,16Ø,169,155,133,161,169,Ø,133
31Ø DATA 162,133,17Ø,133,171,169,129,133,165,133
32Ø DATA 157,173,198,3,133,166,173,197,3,133
33Ø DATA 167,173,196,3,133,168,173,195,3,133
34Ø DATA 169,32,46,232,32,1Ø5,234,173,2Ø3,3
35Ø DATA 56,233,129,13,2Ø4,3,13,2Ø5,3,13
36Ø DATA 2Ø6,3,2Ø8,1,96,16Ø,3,169,2Ø3,32
37Ø DATA 127,233,76,35,236,Ø,Ø,Ø,Ø,154
38Ø DATA 219,255,3,Ø,Ø,Ø,Ø,Ø
39Ø RESTORE : FOR X = 768 TO 975: READ Y: POKE X,Y: NEXT
4ØØ REM *** Demo ***
41Ø CALL 768:Z = 5ØØØØØ:B = 1ØØ: DIM R%(B - 1): REM Z+1
    Random-Zahlen im Bereich Ø bis B-1
42Ø HOME : PRINT "5ØØØØØ ZUFALLSZAHLEN...": FOR X = Ø TO
    Z:R = USR (B):R%(R) = R%(R) + 1: NEXT
43Ø FOR X = Ø TO 99:X$ = STR$ (R%(X)):L = LEN (X$): PRINT
    LEFT$ ("     ",5 - L);: PRINT X$;: NEXT : REM
    Prozentuale Verteilung
44Ø PRINT : FOR X = Ø TO 99:P = R%(X) * 1ØØ * Z / B:P =
    INT (P):S = SGN (P):Y = P - S * 1ØØ: PRINT "    ";Y:
    NEXT
45Ø REM Abweichung -3,74% bis +2,86% vom Mittelwert 5ØØØ
    bei 5ØØØØØ Zahlen
```

COLUMN80.DEMO

```
1ØØ REM 8Ø-Z/Z-IIe/IIc-VTAB-HTAB-Demo/14.Ø3.85/U.Stiehl
11Ø AH = 36:AV = 37: REM HTAV + VTAB 4Ø Z/Z
12Ø BH = 14Ø3:BV = 1531: REM HTAB + VTAB 8Ø Z/Z
13Ø DIM A1%(23,79): DIM A2%(23,79)
14Ø DIM B1%(23,79): DIM B2%(23,79)
15Ø PRINT CHR$ (4)"PR#3": HOME
16Ø REM Cursor-Position peeken
17Ø FOR V = Ø TO 23: FOR H = Ø TO 79
18Ø A1%(V,H) = PEEK (AH):A2%(V,H) = PEEK (AV)
19Ø B1%(V,H) = PEEK (BH):B2%(V,H) = PEEK (BV)
2ØØ PRINT "*";: NEXT H: NEXT V
21Ø REM 4Ø Z/Z + 8Ø Z/Z gegenüberstellen
22Ø HOME : PRINT : PRINT "4Ø-CH     4Ø-CV     8Ø-CH
    8Ø-CV": POKE 34,2
23Ø FOR V = Ø TO 23: FOR H = Ø TO 79
24Ø PRINT A1%(V,H);: POKE BH,1Ø: PRINT A2%(V,H);: POKE
    BH,2Ø: PRINT B1%(V,H);: POKE BH,3Ø: PRINT B2%(V,H)
25Ø NEXT H: PRINT : NEXT V
26Ø TEXT : HOME : PRINT : PRINT "Von unten nach oben...":
    REM Cursor-Position poken
27Ø FOR V = 24 TO 1 STEP - 1: FOR H = 8Ø TO 1 STEP - 1
28Ø VTAB V: POKE BH,H - 1: PRINT "*";: REM So wird's
    gemacht!
29Ø NEXT H: NEXT V
3ØØ POKE 2Ø39, ASC ("*") + 128: REM VTAB 24, HTAB 8Ø
```

Echte Zufallzahlen
Seit der Gründung der Zeitschrift Peeker bin ich regelmäßiger Leser. Über keine andere Publikation zum Apple-Computer bin ich so erfreut wie durch Ihre. Endlich eine Computerzeitschrift, die nicht mangelhaftes Wissen von anderen abkupfert und versucht, die Leser durch Basic-Müll in Rage zu versetzen, sondern vernünftig dokumentierte Assemblerlistings mit guten Beschreibungen präsentiert, von denen man zu guter Letzt auch etwas lernen kann. Ich kaufte mir dann das von Ihnen geschriebene Buch Apple Assembler, vernünftigerweise nicht zu einem Horrorpreis wie viele andere Fachbücher. Der Inhalt konnte mir schon weiterhelfen, obwohl ich noch nicht sehr lange in Assembler programmiere. Allerdings komme ich mit dem Fließkommaakkumulator nicht zurecht, was auch mein Hauptanliegen darstellt. Ich benötige für ein Maschinenprogramm Zufallszahlen; so suchte ich mir die RND-Routine bei $EFAE heraus und konnte nachlesen, daß die ermittelte Zufallszahl in FAC gespeichert wird. Nach langen Versuchen auf meinem Apple IIe konnte ich noch immer keine Zufallszahl berechnen und später verwenden. So rief ich die Apple-Hotline an, die mich nur mit Telefonnummern von Apple-Vertragshändlern abspeiste. Nun glaube ich, daß nur Sie, der Autor des Buches, mir die Funktionsweise und Handhabung des FAC erklären kann, da ja sehr viele Routinen mit ihm arbeiten. Ich hoffe, daß sie mir diesbezüglich helfen können, und wünsche Ihnen weiterhin viel Erfolg mit Ihrer Zeitung. *Oliver Mehl, Berlin*
(Anm.: Die RND-Funktion liefert nur Zufallszahlen für den „Hausgebrauch", d.h. nach einiger Zeit treten immer wieder dieselben Folgen auf. In „Call A.P.P.L.E.", Heft 1/83, S. 30ff. hat ein Mathematiker-Team einen definitiven Algorithmus vorgestellt. Wir zitieren nebenstehend diesen Algorithmus in einer gestrafften und leicht umgeschriebenen Form. Das **RANDOM-DEMO** berechnet 500.000 Zufallszahlen, die nahezu optimal verteilt sind. us)

FLASH und POKE 36, HTAB nach PR#3
Zu ihrer Umfrage in Peeker 3/85 über die Benutzung der Hotline möchte ich Ihnen mitteilen, daß ich diese Hotline schon in Anspruch genommen habe und sehr davon enttäuscht bin. Ich besitze seit Ende letzten Jahres einen Apple IIc, der wie bekannt einen 40- und einen 80-Zeichen-Modus hat. Nun versuchte ich Material über Applesoft-Basic zu bekommen, was auch gelang. In dem Buch über Applesoft-Basic wird beschrieben, daß auf Speicheradresse 36 (dezimal) die horizontale Cursorposition abzufragen ist. Nun gab mein Apple mir für jede Position auf dem Monitor die horizontale Cursorposition 0 an. Der Händler konnte auf meine Anfrage nicht befriedigend antworten und verwies mich an die Apple-Hotline. Dort meldete sich zunächst ein automatischer Anrufbeantworter, der mich nach meinen Problemen und nach meiner Telefonnummer, zwecks Rückruf, fragte. Etwas später kam dann auch der Rückruf. Nach einigen Unklarheiten wußte man nun endlich, was ich wollte, und nachdem ich ein kurzes Prüfprogramm für die Abfrage der horizontalen Cursorposition durchgegeben hatte, stellte sich heraus, daß die Speicherbelegung bei 40- und bei 80-Zeichen-Modus geringfügig unterschiedlich ist. Wo die horizontale Cursorposition im 80-Zeichen-Modus abzufragen ist, konnte mir allerdings nicht gesagt werden. Ebenso warum der FLASH-Befehl beim Apple IIc nicht arbeitet. Man konnte mich nur an die AUGE-Vereinigung und auf die Tatsache, daß die Hotline eigentlich nur für Händlerfragen gedacht wäre, verweisen. *Ulrich Kulemann, Aachen*

(Anm. zum FLASH-Problem: Dieser ist beim „alternativen" Zeichensatz, den man bei 80-Zeichendarstellung normalerweise stets benutzt, zugunsten inverser Kleinbuchstaben nicht vorgesehen. FLASH in Großbuchstaben ist trotzdem möglich durch folgende Routine:

```
10 PRINT CHR$ (4) "PR#3"
20 POKE 49166, 0: REM Aktiviert
   80-Z/Z-FLASH-Zeichensatz
30 POKE 50, 127: PRINT
   "FLASH";: POKE 50, 255:
   PRINT "NORMAL"
```

Anm. zum HTAB-Problem: Eine definitive Lösung des HTAB-Problems nach PR#3 können Sie dem nebenstehend abgedruckten **COLUMN80.DEMO** entnehmen. us)

Leserbriefe

Indizes und Exponenten beim Imagewriter
Die DIP-Schalter-Stellung, die ich normalerweise beim Imagewriter benutze, ist sw1: -1 und -2 open, -3 bis -6 closed, -7 und -8 open; sw2: -1 und -2 closed, -3 und -4 open. Schalter sw1 -5 wird nun bei abgeschaltetem Gerät in Position open gebracht, dann das Gerät anschalten. Nun boote ich mit der Imagewriter-Toolkit-Diskette (wird beim Kauf des Geräts mitgeliefert) und wähle Punkt 5 des Menüprogramms aus (Download super and subscripts). Der Schreibkopf im Imagewriter reagiert darauf mit einem kurzen Ruck. Mit Option 6 (Quit) verläßt man das Toolkit-Menü (es erscheint das Basic-Ü auf dem Bildschirm) und bootet jetzt mit der Applewriter-IIe-Diskette (Imagewriter natürlich nicht abschalten, da die eingespeicherten Sonderzeichen sonst gelöscht werden).
Aus dem Schreibprogramm können jetzt die Sonderzeichen über die Sequenz Ctrl-V ESC ' (Apostroph neben der Delete-Taste, ohne Shift) angesteuert werden. Zum normalen Zeichenfont kehrt man mit der Sequenz ESC $ Ctrl-V zurück.
Beispiel: Chemische Formel für Wasser: H_2O. Einzutippen: „H Ctrl-V ESC ' 2 ESC $ Ctrl-V O". Das ESC erscheint auf dem Bildschirm als schwarzes Ä auf hellem Grund. Welche Tasten man drükken muß, um die kleinen Buchstaben und Ziffern zu erhalten, kann man im Imagewriter-Handbuch, Teil 2, S. 25 und 26 nachlesen. Das gleiche gilt auch für Exponenten. Hier muß man wirklich die Tabelle im Handbuch konsultieren, denn die Zahlen erscheinen nicht als Zahlen, sondern als Zeichen, die man erhält, wenn man beispielsweise Shift-1, Shift-2 etc. drückt. Wenn die Zahlen von 1 bis 0 z.B. als Exponenten erscheinen sollen, muß man !, ", #, $, %, &, ', (,), § eintippen, und so sieht das dann auch auf dem Bildschirm aus. Die oben beschriebenen Kontrollsequenzen sind wieder verwendbar.
Das ist natürlich für einen Mathematiker das absolute Chaos, wenn er am Bildschirm versucht, z.B. einen wissenschaftlichen Text mit vielen Formeln Korrektur zu lesen. Auf dem Bildschirm sieht man nämlich xÄ'!"#Ä$/xÄ')§Ä$ = xÄ'!"# =)§Ä$ für die zu druckende Formel

$x^{123}/x^{90} = x^{123-90}$

Man kann natürlich häufig vorkommende Sequenzen über den Kurztextspeicher laden (Ctrl-G und ? im Applewriter). Hier darf man nicht vergessen, auch die Ctrl-V-Sequenz mit einzutippen. Sie erscheint im Kurztextspeicher beim Eintippen als inverses V (Beim Europlus braucht man bei der Verwendung des Kurztextspeichers, zumindest beim englisch geschriebenen Applewriter, den ich hatte, das Ctrl-V nicht verwenden). *Horst Ibelgaufts, München*

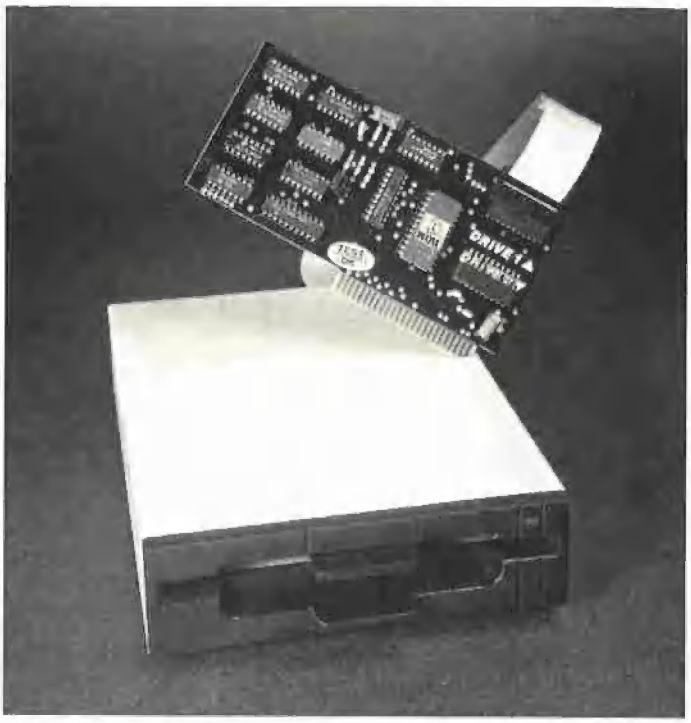
Exbasic Level II
Als Autor des Programms „EXBASIC LEVEL II" in den Versionen für Apple II+, IIe und Basis 108 möchte ich im beiderseitigen Interesse einer sachlich korrekten Berichterstattung folgende zwei Punkte zu ihrem Testbericht im „Peeker", Heft 1-2/1985 richtigstellen:
1. Der „PRINT AT"-Befehl arbeitet auch mit der 80-Zeichenkarte des Apple IIe uneingeschränkt, d.h. alle 1920 Positionen können adressiert werden (im Gegensatz zu „HTAB" oder Komma-Formatierung).
2. Exbasic Level II wandelt genauso wie der Applesoft-Interpreter bereits bei der Eingabe alle seine Befehle in Tokens um, die platzsparend in einem Byte abgelegt werden. Die Tokens sind auch im Handbuch auf der Seite 99 und im Apple-Addendum auf Seite 24 vollständig dokumentiert. *Matthias Kapffer, Wiesbaden*

Mehr Mac
Als erstes einmal ein Lob auf Ihre Zeitschrift! Sie ist gut gemacht und objektiv. Aber... In Ihrer Zeitschrift wird der Macintosh immer als „Schrott" dargestellt. Gut, der Macintosh hat große Fehler, aber was hilft es, wenn man immer nur sagt, daß das Betriebssystem schlecht ist, und daß der Macintosh nicht überzeugen kann etc. Wie wäre es z.B., wenn man neben aller Kritik auch die guten Seiten dieses Computers sieht und vor allem versucht die Fehler auszubügeln! Ich bin ein begeisterter Macintosh-User und auch mich nervt es,

wenn der Mac „Jahre" zum Kopieren langer Programme oder ganzer Disketten braucht; auch mich nervt es, wenn der Mac so lange rechnet, scrollt etc.
Mein Vorschlag:
1. Baldige Fortführung des (sehr guten und informativen) Teils „Microsoft Basic leicht geMACht".
2. Ich schließe mich der Meinung von Herrn Herbert Schmidt an (Leserbrief aus Heft 3/85) und würde es auch sehr begrüßen, wenn Sie, anschließend an Punkt 1, eine Reihe über das MAC-Pascal bringen würden.
3. Einführung einer „Tips- und Tricks-Ecke", wo Vorschläge zur Verbesserung des Macs (soft- und hardwaremäßig) eingebracht werden.
Insgesamt: Mehr Platz für den MAC in Ihrer Zeitschrift damit seine fatalen Fehler radikal ausgemerzt werden! *Sassan Tat, Düsseldorf*
(Anm.: Im Augenblick ist noch nicht entschieden, ob im Anschluß an die Basic-Serie mit einer Mac-Pascal- oder 68000-Assembler-Serie begonnen wird. Peeker-Leser werden gebeten, uns Ihre Präferenzen mitzuteilen. us)

Mehr Apple IIc

Als ich vor kurzem von einem Bekannten auf Ihre Zeitschrift aufmerksam gemacht wurde, habe ich mit Freude registriert, daß sie einen guten Mittelkurs zwischen allgemeinem Salonblatt und technischer Zeitschrift steuert und sich weitgehend dem großen Bereich der Apple-II-Rechner widmet. In diesem Zusammenhang darf ich erwähnen, daß ich eine Berichterstattung über Entwicklungen, die speziell den Apple IIc betreffen, sehr begrüßen würde. Meiner Entscheidung, einen Apple IIc zu kaufen, war eine Güterabwägung vorausgegangen: Soll ich das offene System des IIe oder das kompakte System des IIc vorziehen? Schließlich überwog, wenn auch nur knapp, die Überlegung der Tragbarkeit des Apple IIc. Aus meinem unmittelbaren Bekanntenkreis weiß ich auch von etlichen weiteren Fällen, wo einer einen Apple IIc wegen der Möglichkeit, ihn an zwei Stellen (z.B. am Arbeitsplatz und zu Hause) zu benutzen, gekauft hat. Meine Bitte an die Redaktion des „Peeker" wäre dann die, daß man, auch bei dem augenscheinlich geschlossenen System des IIc, nach Hardware-Entwicklungen (z.B. dem Anschluß einer Festplatte oder einer Accelerator-Karte) Ausschau hält, die die immer größer werdende Gruppe der Apple-IIc-Besitzer interessieren würden. *Dr. Raymond Hickey, Bonn*
(Anm.: Nach unseren Kenntnissen scheint man an einer Accelerator-Karte für den IIc noch nicht zu arbeiten. Andere und größere Laufwerke für den IIc gibt es jedoch bereits, desgleichen Adapter, die die üblichen IIe-Disk-Controller ersetzen. Softwaremäßig unterscheidet sich der IIc vom IIe nur in Feinheiten. So laufen beispielsweise fast alle Peeker-Programme auch auf dem IIc. Eine entsprechende Aufstellung der abweichenden Monitorroutinen wird in Kürze im Peeker veröffentlicht. us)



Slim-Line-Laufwerk von Chinon

In die Reihe der Slim-Line-Laufwerke gesellt sich nun auch ein Importgerät, das sich unter anderem durch einen recht günstigen Preis von ca. DM 500,- auszeichnet.
Das von der Firma D.O.S. in Schwäbisch-Hall importierte Chinon-Laufwerk, das an den Apple-Controller angeschlossen werden kann, besitzt einen direkt angetriebenen Disk-Teller in einem Alu-druckguß-Chassis, wobei im Gegensatz zur Disk II 40 Spuren bearbeitet werden können. Laut Herstellerangaben ist auch der Halbspurbetrieb möglich.
Die Zentrierung der Diskette erfolgt beim Schließen des Laufwerks durch einen anfänglich etwas gewöhnungsbedürftigen Schnappverschluß, der während des Tests jedoch keine nachteiligen Erscheinungen zeigte. Die

b. w.

Druckerumschaltung per Knopfdruck

Häufig werden an einem Computer ein Matrix- und ein Schönschreibdrucker bzw. eine Schreibmaschine betrieben. Beim Apple II läßt sich dieses Problem mit Hilfe zweier Interfacekarten lösen. Sollen zwei Computer denselben Drucker teilen, so ist dies nur durch Umstecken der Anschlüsse erreichbar. Abhilfe schafft hier ein „Printer Interface Selector", der den Anschluß von zwei Computern und zwei Druckern ermöglicht. Durch einfaches Umschalten können somit beide Computer beide Drucker benutzen.
Die Firma Radio Shack vertreibt in den USA ein solches Gerät, mit dessen Hilfe zwei Parallelschnittstellen geschaltet werden können und das folgende Kombinationen unterstützt:
- zwei Computer an einem Drucker
- ein Computer an zwei Druckern
- zwei Computer an zwei Druckern.

Das Gerät, das in den USA $119.95 kostet, wird mit einem für Deutschland nicht verwendbaren 120V-Netzteil ausgeliefert, das jedoch durch ein Netzteil mit 15V Ausgangsspannung ersetzt werden kann.
Eine deutsche Bezugsquelle ist derzeit nicht bekannt.

Laufgeräusche des Disk-Tellers und der Schreib/Lesekopf-Mechanik erwecken einen vertrauenswürdigen Eindruck.
Eine erhöhte Zugriffszeit gegenüber der Disk II konnte im Test nicht ermittelt werden, vielmehr war die benötigte Zeit zur Abspeicherung mehrerer Binär-Files um ca. 5% länger als beim Apple-Laufwerk (die benutzte Diskette wurde auf dem Chinon-Laufwerk formatiert).
Da kein Dauertest durchgeführt wurde, kann nichts über die langfristige Zuverlässigkeit dieses Laufwerks ausgesagt werden. Der insgesamt robuste Eindruck läßt jedoch darauf schließen, daß bei dem vorliegenden Gerät das Preis/Leistungs-Verhältnis ausgewogen ist.

The Last Disk-Utility

Ein umfangreicher Diskettenediteor

getestet von H. Grumser

Die rege Benutzung von Disketten als Massenspeicher erfordert immer wieder die Bearbeitung einzelner Sektoren, sei es, um die Daten von teilweise beschädigten Disketten zu retten oder Änderungen vorzunehmen, die das Betriebssystem nicht unterstützt. Zu diesem Zweck bietet die Firma FOCUS Computer GmbH eine umfangreiche Sammlung von Disketten-Hilfsprogrammen an, die sowohl auf Sektorebene als auch auf der Ebene der systemspezifischen Diskettenorganisation eine umfangreiche Bearbeitung gestatten. Dieses unter dem Namen T.L.D.U. zusammengefaßte Programm unterstützt die Betriebssysteme DOS 3.3, UCSD, CP/M sowie ProDOS.
Zur Bearbeitung einzelner Sektoren stellt die T.L.D.U. 10 (bzw. 16, s.u.) Editierpuffer zur Verfügung, die beliebig gelesen oder geschrieben werden können. Der hierfür eingebaute Editor ermöglicht das Überschreiben, Einsetzen und Löschen einzelner Bytes im ASCII- und Hex-Modus; die Anzeige erfolgt wahlweise im ASCII-, hexadezimalen oder gemischten Format, wobei auch die Möglichkeit der (nicht editierfähigen) Ausgabe als Disassembler-, Basic- oder Textlisting besteht.
Auf höherer Ebene können Directories, Dateien und bei DOS 3.3 die VTOC ohne die exakten Kenntnisse der Diskettenorganisation bearbeitet werden. Hier empfiehlt sich jedoch in jedem Fall die Lektüre der einschlägigen Literatur vor der Verwendung dieser Kommandos (s.u.). Als besonders hilfreich erweist sich in diesem Zusammenhang der automatische Abgleich der VTOC mit den TSL-Sektoren und die Freigabe noch belegter Sektoren, die bei der Kürzung von Dateien entstehen.
Eine weitere nützliche Einrichtung ist die Möglichkeit, Bytefolgen oder Strings auf der Diskette zu suchen (auch sektorüberlappend), um sich z.B. auf einer beschädigten Diskette zurechtzufinden (die Suche nach TSL-Sektoren existiert als spezieller Befehl).
Der eindrucksvollste Teil der T.L.D.U. ist ohne Zweifel die Implementierung einer Macro-Sprache, die unter anderem die Automatisierung häufig benötigter Befehlsfolgen gestattet. Mit Hilfe eines Editors können Macros definiert und im Kommando-Modus jederzeit ausgeführt werden. Die Sprache umfaßt, außer Variablen, Schleifen, der Möglichkeit der IF-THEN-ELSE-Programmierung und dem Aufruf von „Untermacros", auch eine Fülle nützlicher Anweisungen (die nur zum Teil denen des Kommando-Modus entsprechen), mit deren Hilfe die umfangreiche Bearbeitung von Disketten möglich wird, die ansonsten nur durch aufwendige System-Programme realisierbar wäre. Bei der Benutzung von Macros stehen 16 Puffer zur Verfügung. So können Macros z.B. zum
- Erstellen DOS-loser Datendisketten,
- Modifizieren des Catalogs,
- „UNDELETEn" gelöschter Dateien,
- Kopieren zwischen verschiedenen Betriebssystemen

und zur Lösung vieler anderer Aufgaben eingesetzt werden.
Die Syntax der Sprache ist leicht verständlich, so daß bereits nach kurzer Einarbeitungszeit die ersten selbstgeschriebenen Programme die Sammlung der mitgelieferten Macros erweitern können. Abgesehen von einer etwas dürftig ausgefallenen Benutzerschnittstelle während der Abarbeitung von Macros läßt die Sprache keine Wünsche offen.
Der Aufruf der Befehle erfolgt durch Eingabe eines Buchstabens oder Ctrl-Zeichens, wobei bei manchen Kommandos ein zweites Zeichen und gegebenenfalls eine Name folgen muß. Trotz dieser baumartigen Einteilung ist der Zugang zu den einzelnen Kommandos zum Teil etwas unzulänglich, zumal das einzugebende Zeichen und der entsprechende Befehl nicht immer mnemonisch zueinanderpassen. Der sehr umfangreiche Befehlssatz bereitet in diesem Zusammenhang anfängliche Schwierigkeiten, was jedoch kein Problem einer schlechten Benutzerführung ist.
Zur Orientierung kann für die übergeordneten Kommandos ein Hilfsfenster eingeblendet werden (beim Apple II Plus nur mit 16K-Erweiterung), das eine kurze Zusammenfassung der Befehle enthält, das Handbuch jedoch nicht ersetzen kann.
Das über 100 Seiten umfassende DIN-A5-Handbuch (Ringbuch, fotokopierter Schönschreibdrucker-Printout) ist übersichtlich gegliedert. Die Befehle, speziell die Macro-Befehle, werden zum Teil sehr kurz behandelt, so daß auf das altbewährte Trial-and-error-Verfahren zurückgegriffen werden muß. Eine zügige Einarbeitung ist dennoch möglich. Das Handbuch vermittelt keine Kenntnisse in der Organisation der Disketten unter den verschiedenen Betriebssystemen, sondern verweist unter anderem auf das Buch „ProDOS für Aufsteiger" von Ulrich Stiehl. Das bisher leider fehlende Register, mit einer Übersicht aller Befehle, soll laut Angaben der Firma bald erscheinen.
Vor dem Starten der T.L.D.U. muß eine DOS-3.3-Diskette gebootet werden. Obwohl nur die RWTS benutzt wird, bleibt das DOS erhalten, kann jedoch wahlweise durch die Einrichtung weiterer Puffer überschrieben werden. Die Verarbeitung von Disketten mit höherer Spurenzahl (bis 160) wird unterstützt. Da alle Schreib/Lese-Befehle mit Hilfe der RWTS ausgeführt werden, beschränkt sich der Zugriff auf das Standardformat. Als Option lassen sich jedoch Header- und Trailerbytes ändern und die Checksum-Routinen lahmlegen.
Mit der T.L.D.U. erhält der Käufer ein umfangreiches Werkzeug zur Bearbeitung von Disketten, das trotz kleiner Unebenheiten kaum Wünsche übrigläßt. Für all jene, die Disketten öfters byteweise bearbeiten müssen, ist die Ausgabe von DM 99.- in jedem Fall empfehlenswert.

Tabelle der Interpreter- und Monitor-Adressen

Die nachfolgende doppelte Tabelle enthält alle Applesoft-Interpreter-Einsprungstellen ($D000-$F7FF) des Apple II Plus sowie alle Monitor-Einsprungstellen ($F800-$FFFF) des Apple II in numerischer, d.h. nach Adressen, und alphabetischer, d.h. nach Labels, sortierter Reihenfolge. Die Monitor-Labelnamen gehen noch auf Steve Wozniak zurück, während die Interpreter-Labelnamen auf der Nomenklatur der Zeitschrift „Call A.P.P.L.E." sowie auf dem Interpreter-Quellcode von Prof. Glen E. Bredon basieren. Peeker-Autoren wird empfohlen, die Labelnamen zu übernehmen, um eine einheitliche Terminologie zu gewährleisten. Vergleichstabellen für den Monitor von Apple II Plus, IIe und IIc sowie Ergänzungen zu dem nur beim Apple IIc geringfügig abweichenden Applesoft-Interpreter werden veröffentlicht, wenn die neuen ROMs für den IIe erschienen sind (wahrscheinlich im Sommer). Tabellen in der hier vorgestellten Art dienen nur als „Quick-Reference" für diejenigen, die bereits wissen, was sich hinter den Namen versteckt. Neben diesen nackten Adreß-Label-Tabellen werden wir deshalb eine Auswahl der wichtigsten Labels nach Sachgruppen zusammenfassen und anhand von Miniprogrammen erläutern.

U. Stiehl

Adressen-Labels $D000–$FFFF

$0000 = GOWARM
$0000 = LOC0
$0001 = LOC1
$0003 = GOSTROUT
$000A = USR
$000D = CHARAC
$000E = ENDCHR
$000F = NUMDIM
$000F = PNTR
$0010 = DIMFLG
$0011 = VALTYP
$0013 = DATAFLG
$0013 = GARFLG
$0014 = SUBFLG
$0015 = INPUTFLG
$0016 = CPRMASK
$0016 = SIGNFLG
$001A = SHAPEL
$001B = SHAPEH
$001C = HCOLOR1
$001D = COUNTH
$0020 = WNDLFT
$0021 = WNDWDTH
$0022 = WNDTOP
$0023 = WNDBTM
$0024 = CH
$0025 = CV
$0026 = GBASL
$0027 = GBASH
$0028 = BASL
$0029 = BASH
$002A = BAS2L
$002B = BAS2H
$002C = H2
$002C = LMNEM
$002C = RTNL
$002D = RMNEM
$002D = RTNH
$002D = V2
$002E = CHKSUM
$002E = FORMAT
$002E = MASK
$002F = LASTIN
$002F = LENGTH
$002F = SIGN
$0030 = COLOR
$0030 = HMASK
$0031 = MODE
$0032 = INVFLG
$0033 = PROMPT
$0034 = YSAV
$0035 = YSAV1
$0036 = CSWL
$0037 = CSWH
$0038 = KSWL
$0039 = KSWH
$003A = PCL
$003B = PCH
$003C = A1L
$003C = XQT
$003C = XQTNZ
$003D = A1H
$003E = A2L
$003F = A2H
$0040 = A3L
$0041 = A3H
$0042 = A4L
$0043 = A4H
$0044 = A5L
$0045 = A5H
$0045 = ACC
$0046 = XREG
$0047 = YREG
$0048 = STATUS
$0049 = SPNT
$004E = RNDL
$004F = RNDH
$0050 = ACL
$0050 = LINNUM
$0051 = ACH
$0052 = TEMPPT
$0052 = XTNDL
$0053 = LASTPT
$0053 = XTNDH
$0054 = AUXL
$0055 = AUXH
$0055 = TEMPST
$005E = INDEX
$0060 = DEST
$0062 = RESULT
$0067 = TXTTAB
$0069 = VARTAB
$006B = ARYTAB
$006D = STREND
$006F = FRETOP
$0071 = FRESPC
$0073 = MEMSIZ
$0075 = CURLIN
$0077 = OLDLIN
$0079 = OLDTEXT
$007B = DATLIN
$007D = DATPTR
$007F = INPTR
$0081 = VARNAM
$0083 = VARPNT
$0085 = FORPNT
$0087 = LASTOP
$0087 = TXPSV
$0089 = CPRTYP
$008A = FNCNAM
$008A = TEMP3
$008C = DSCPTR
$008F = DSCLEN
$0090 = JMPADRS
$0091 = LENGTH
$0092 = EXTRASV
$0093 = TEMP1
$0094 = ARYPNT
$0094 = HIGHDS
$0095 = PICK
$0096 = HIGHTR
$0098 = TEMP2
$0099 = INDX
$0099 = TMPEXP
$009A = EXPON
$009B = DPFLG
$009B = LOWTR
$009C = EXPSGN
$009D = DSCTMP
$009D = FAC
$00A0 = VPNT
$00A2 = FACSGN
$00A3 = SERLEN
$00A4 = FPGEN
$00A5 = ARG
$00AA = ARGSGN
$00AB = SGNCPR
$00AB = STRNG1
$00AC = EXTRAFAC
$00AD = SERPNT
$00AD = STRNG2
$00AF = PRGEND
$00B1 = CHRGET
$00B7 = CHRGOT
$00B8 = TXTPTR
$00C9 = RNDSEED
$00D0 = DXL
$00D1 = DXH
$00D2 = DY
$00D3 = QDRNT
$00D4 = EL
$00D5 = EH
$00D6 = LOCK
$00D8 = ERRFLG
$00DA = ERRLIN
$00DC = ERRPOS
$00DE = ERRNUM
$00DF = ERRSTK
$00E0 = X0L
$00E1 = X0H
$00E2 = Y0
$00E4 = HCOLORZ
$00E5 = HNDX
$00E6 = HPAG
$00E7 = SCALEZ
$00E8 = SHAPEPNT
$00EA = COLCOUNT
$00F0 = FIRST
$00F1 = SPEEDZ
$00F2 = TRCFLG
$00F3 = ORMASK
$00F4 = TXTPSV
$00F6 = CURLSV
$00F8 = REMSTK
$00F9 = ROTZ
$0100 = STACK
$0200 = IN
$03F5 = AMPER
$03F8 = USRADR
$03FB = NMI
$03FE = IRQLOC
$C000 = IOADR
$C000 = KBD
$C000 = KEY
$C010 = KBDSTRB
$C020 = TAPEOUT
$C030 = SPKR
$C050 = TXTCLR
$C051 = TXTSET
$C052 = MIXCLR
$C053 = MIXSET
$C054 = LOWSCR
$C055 = HISCR
$C056 = LORES
$C057 = HIRES
$C060 = TAPEIN
$C064 = PADDL0
$C070 = PTRIG
$D000 = CMDTABL
$D080 = UNFNC
$D0B2 = MATHTBL
$D0C7 = MINUS
$D0CA = EQUAL
$D0CD = PLUS
$D0D0 = TOKTABL
$D260 = ERRMSG
$D260 = NXwoFOR
$D270 = SYNTXERR
$D276 = RTNwoGSB
$D28A = OofDATA
$D295 = ILLQUAN
$D2A5 = OVFLOW
$D2AD = OofMEM
$D2BA = UNDSTAT
$D2CB = BADSUBS
$D2D8 = REdimARR
$D2E5 = DIVbyZRO
$D2F5 = ILLDIR
$D303 = TYPEMISS
$D310 = STRtoLNG
$D31F = FORMtoCX
$D332 = CANTCON
$D340 = UNDFUNC
$D350 = ERRIN
$D358 = INMSG
$D35D = BREAKIN
$D365 = GTFORPNT
$D36A = FNDFOR
$D37F = SAMEFOR?
$D38B = NXFOR
$D392 = RET1
$D393 = BLTU
$D39A = BLTU2
$D3B7 = SETEND
$D3C3 = MVBYT
$D3C7 = NXBYT
$D3CE = NXPAG
$D3D6 = CHKMEM
$D3E3 = REASON
$D3ED = RS1
$D3F1 = RS2
$D3FC = RS3
$D40F = RET2
$D410 = MEMERR
$D412 = ERROR
$D419 = DOERRMSG
$D41F = ERLUP
$D431 = PRNTIN?
$D43C = RESTART
$D45C = NXLIN
$D49F = NL1
$D4A7 = MVDWN
$D4B5 = NEWLN?
$D4D1 = MVPRG
$D4EA = INSRTLIN
$D4F2 = LINKSET
$D4FE = NXLINK
$D50F = PUTLINK
$D511 = FINDEOL
$D52C = INLIN
$D52E = INLIN2
$D539 = GDBUFS
$D541 = STRIP
$D54C = NOI
$D553 = INCHR
$D559 = GETIN
$D56C = PARSE
$D56D = NXCHR
$D578 = SE
$D588 = TOK?
$D590 = ISTOK?
$D5A2 = NY
$D5A7 = NX
$D5A8 = LIN
$D5CB = PUTTOK
$D5CD = PUTIN
$D5E0 = SSF
$D5E2 = REM?
$D5E9 = SHFTIN
$D5F2 = SHIN
$D5F9 = SKIPTOK
$D5FD = SK2
$D604 = PLU?
$D610 = DONE
$D61A = FNDLIN
$D61E = FL1
$D635 = FL2
$D63E = GETLINK
$D647 = NOSUCH
$D648 = RET3
$D649 = NEW
$D64B = SCRTCH
$D665 = SETPTRS
$D66A = CLEAR
$D66C = CLEARC
$D683 = STKINI
$D696 = RET4
$D697 = STXTPT
$D6A5 = LIST
$D6B1 = STRTRNG
$D6C4 = ENDRNG
$D6CC = MAINLST
$D6DA = NXLST
$D6F5 = LSTD?
$D6F7 = LST1LIN
$D6FE = LISTLOOP
$D702 = SENDCHR
$D712 = NCR
$D724 = LISTED
$D72C = GETCHR
$D731 = GC
$D734 = TOKEN?
$D746 = SKPTK
$D749 = TOKLP
$D750 = PRTOK
$D755 = TOKLUP
$D75F = TOKDONE
$D766 = FOR
$D777 = FOR2
$D7AF = STEP
$D7C3 = ONESTEP
$D7D2 = NEWSTT
$D7E5 = DIRCT
$D805 = TRACE?
$D81D = TR1
$D826 = GOEND
$D828 = GOCMD
$D82A = GOCMD2
$D83F = NOTOK
$D842 = COLON?
$D846 = JSY
$D849 = RESTORE
$D853 = SETDA
$D857 = RET5
$D858 = ISCNTC
$D860 = GK
$D863 = ERFLG?
$D86C = CTRC?
$D86E = STOP
$D870 = END
$D871 = END2
$D888 = END3
$D88A = END4
$D893 = GOSTART
$D896 = CONT
$D8A1 = CON
$D8AF = RET6
$D8B0 = SAVE
$D8C9 = LOAD
$D8ED = JLNK
$D8F0 = VARTIO
$D901 = PROGIO
$D912 = RUN
$D91B = RUNLINE
$D921 = GOSUB
$D935 = GOLINE
$D93E = GOTO
$D955 = GO1
$D959 = GO2
$D96A = RET7
$D96B = POP
$D97C = UNDERR
$D981 = GSYNER
$D984 = RETURN
$D995 = DATA
$D998 = ADDON
$D9A2 = RET8
$D9A3 = DATAN
$D9A6 = REMN
$D9AE = RM1
$D9B6 = RM2
$D9C5 = PULL3
$D9C9 = IF
$D9D8 = TRUE?
$D9DC = REM
$D9E1 = IFTRUE
$D9E9 = JGOCMD
$D9EC = ONGOTO
$D9F4 = GOTO?
$D9F8 = ONCNT
$DA00 = NXNUM
$DA0B = RET9
$DA0C = LINGET
$DA12 = ASCHEX
$DA40 = NXDIG
$DA46 = LET
$DA63 = LET2
$DA77 = LETREAL
$DA7A = LETSTR
$DA7B = PUTSTR
$DA8C = DESC?
$DA9A = COPSTR
$DAA1 = NEWDESC
$DAB7 = COPY
$DACF = PRSTRING
$DAD5 = PRINT
$DAD7 = PRINT2
$DAFB = CRDO
$DB00 = NEGATE
$DB02 = RET10
$DB03 = TAB
$DB0E = NXCLM
$DB16 = TABWHERE
$DB21 = SPC?
$DB2B = TABIT
$DB2C = NXSPC
$DB2F = NEXTCHR
$DB35 = DOSPC
$DB3A = STROUT
$DB3D = STRPRT
$DB44 = NXCHAR
$DB57 = OUTSP
$DB5A = OUTQUES
$DB5C = OUTDO
$DB64 = SEND
$DB71 = INPUTERR
$DB7B = READERR
$DB7F = ERLIN
$DB86 = INPERR
$DB87 = RESPERR
$DB90 = DOREENT
$DBA0 = GET
$DBB2 = INPUT
$DBC4 = QOUT
$DBC7 = DIR?
$DBDC = NXIN
$DBE2 = READ
$DBE9 = ZF
$DBEB = MAININP
$DBF1 = NXINP
$DC1F = SNDQ?
$DC27 = STXP
$DC2B = INSTART
$DC3F = PUTCHR
$DC4B = PENCHR
$DC4C = PECHR
$DC57 = SKP
$DC63 = NUMIN
$DC69 = DATIN
$DC72 = WNX
$DC7E = SWPNT
$DC99 = INPFIN
$DCA0 = FINDATA
$DCB9 = NXS
$DCC6 = INPDONE
$DCD1 = NTD
$DCDE = RET11
$DCDF = EXIG
$DCEF = REENT
$DCF9 = NEXT
$DCFF = VARNXT
$DD02 = SKPV
$DD0D = GERR
$DD0F = GOTFOR
$DD52 = GONEWST
$DD55 = ENDFOR
$DD67 = FRMNUM
$DD6A = CHKNUM
$DD6C = CHKSTR
$DD6D = CHKVAL
$DD73 = RET12
$DD74 = CV2
$DD76 = MISMTCH
$DD78 = JERROR
$DD7B = FRMEVL
$DD86 = FEVLOOP
$DD95 = FRMEVL2
$DD98 = CPROP
$DDB4 = CHKTYP
$DDC5 = ARITH
$DDCD = PREFTEST
$DDD6 = NXOP
$DDD7 = SAVOP
$DDE4 = COMPARE
$DDEE = ND
$DDF6 = PREFNC
$DDFD = PSHMAD
$DE0D = SNTXERR
$DE10 = PSHF
$DE15 = PSHFACX
$DE20 = PUSHFAC
$DE35 = NOTMATH
$DE38 = GOEX
$DE3A = DOMTH
$DE41 = DMTH
$DE43 = DOMATH
$DE5D = EXIT
$DE60 = GETVAL
$DE64 = SKIP
$DE69 = NUMBER
$DE6C = VAR?
$DE81 = STRTXT
$DE8A = ST1
$DE90 = NOT?
$DE98 = EQUOP
$DE9F = NOTZ
$DEA4 = FN?
$DEAB = SGN?
$DEB2 = PARCHK
$DEB8 = CHKCLS
$DEBB = CHKOPN
$DEBE = CHKCOM
$DEC0 = SYNCHR
$DEC9 = SYNERR

Adressen-Labels $D000–$FFFF

$DECE = MIN
$DED0 = EQUL
$DED5 = VARL
$DEE5 = VR1
$DEF6 = VR2
$DEF9 = SCREEN
$DF0C = UNARY
$DF3A = NOTinstr
$DF3F = GOROUT
$DF4F = OR
$DF55 = AND
$DF5D = FALSE
$DF60 = TRUE
$DF65 = POSOP
$DF7D = STRCMP
$DFA5 = SFS
$DFAA = CMPLOOP
$DFB0 = NUMCMP
$DFB5 = DOCMP
$DFC1 = CMPDONE
$DFCA = JF
$DFCD = PDL
$DFD6 = NXDIM
$DFD9 = DIM
$DFE3 = PTRGET
$DFE8 = PTRGET2
$DFEA = PTRGET3
$DFF4 = BADNAM
$DFF7 = NAMOK
$E000 = BASIC
$E003 = BASIC2
$E007 = MORNAM
$E011 = GTLT
$E012 = BYPASS
$E01C = STRNG?
$E026 = INTVAR?
$E036 = NIN
$E03D = SCDCH
$E046 = JARY
$E049 = BSB
$E04F = VSEARCH
$E059 = NXVAR
$E05B = NV1
$E065 = NV2
$E073 = NXPTR
$E07D = ISLETC
$E086 = RTN1
$E087 = NOTFND
$E09A = TWOBRK
$E09C = NEWVAR
$E0B2 = NWV
$E0DE = SETVPNT
$E0E8 = SVP
$E0ED = GETARY
$E0EF = GETARY2
$E0F9 = GD
$E0FE = NEGNUM
$E100 = MAKINT
$E108 = MKINT
$E10C = AYINT
$E119 = MI1
$E11B = MI2
$E11E = ARRAY
$E12C = NXTDIM
$E169 = FNDARY
$E16D = ARYLOOP
$E179 = ARYNAM?
$E188 = NXARY
$E196 = SUBERR
$E199 = IQERR
$E19B = JER
$E19E = ARYFOUND
$E1AA = CHKDIM
$E1B8 = NOTFOUND
$E1C1 = MAKARY
$E1D5 = NINT
$E1DE = RAR
$E1E7 = SAVDIM
$E1F7 = DFLTDIM
$E21A = ZARY
$E229 = ZLUP
$E22E = NXPG
$E24B = FNDELEM
$E253 = DIMLUP
$E269 = GSE
$E26C = GME
$E26F = DIMOK

$E270 = DIM0K2
$E281 = NXDM
$E292 = NINTA
$E298 = RARY
$E2AC = RTN2
$E2AD = MULT
$E2B6 = MU1
$E2C0 = MU2
$E2D9 = MU3
$E2DE = FRE
$E2E5 = FRE2
$E2F2 = GIVAYF
$E2FF = POS
$E301 = SNGFLT
$E306 = ERRDIR
$E30E = UNDFNC
$E313 = DEF
$E341 = FNC?
$E354 = FUNCT
$E378 = SAVOLD
$E3A9 = GETOLD
$E3AF = FNCDATA
$E3C5 = STR
$E3D5 = STRINI
$E3DD = STRSPA
$E3E7 = STRLIT
$E3ED = STRLT2
$E3F7 = FEND
$E404 = QUO?
$E408 = ZEND
$E409 = NZ
$E415 = FE1
$E41F = FE2
$E42A = PUTNEW
$E432 = JERR
$E435 = PUTEMP
$E452 = GETSPA
$E454 = GETSPC
$E45F = CY
$E469 = GOTSPA
$E474 = FULL
$E484 = GARBAG
$E488 = FNDVAR
$E4A0 = TVAR
$E4A9 = SVARS
$E4B5 = SVAR
$E4BD = SVARGO
$E4C2 = ARYVAR
$E4CA = ARYVA2
$E4CE = ARYVA3
$E4D9 = ARYVGO
$E50C = ARYSTR
$E514 = GOGO
$E519 = DVARS
$E523 = DVAR
$E538 = DV1
$E542 = DV2
$E552 = DVARTS
$E55D = VDONE
$E562 = GRBPAS
$E597 = CAT
$E5B7 = NTL
$E5D4 = MOVINS
$E5E2 = MOVSTR
$E5E6 = MOVESTR
$E5EA = MVS2
$E5F3 = MVS3
$E5FD = FRESTR
$E600 = FREFAC
$E604 = FRETMP
$E630 = NB
$E635 = FRETMS
$E645 = RTN3
$E646 = CHRSTR
$E65A = LEFTSTR
$E660 = INS1
$E666 = INS2
$E667 = INS3
$E668 = INS4
$E67F = BPD
$E686 = RIGHTSTR
$E691 = MIDSTR
$E6A2 = INS5
$E6B9 = INSTRNG
$E6D6 = LEN
$E6DC = GETSTR
$E6E5 = ASC

$E6F2 = GOIQ
$E6F5 = GTBYTC
$E6F8 = GETBYT
$E6FB = CONINT
$E707 = VAL
$E70F = VL2
$E727 = VL3
$E73D = POINT
$E746 = GTNUM
$E74C = COMBYTE
$E752 = GETADR
$E764 = PEEK
$E77B = POKE
$E784 = WAIT
$E793 = WT2
$E797 = WT3
$E79F = RTN4
$E7A0 = FADDH
$E7A7 = FSUB
$E7AA = FSUBT
$E7B9 = ADØ
$E7BE = FADD
$E7C1 = FADDT
$E7C6 = AD1
$E7CE = AD2
$E7EA = AD3
$E7EE = AD4
$E7FA = AD5
$E806 = SUBMAN
$E829 = SGNIF
$E82E = SIGNIF
$E832 = FLOOP
$E84E = ZEROFAC
$E850 = AtoFAC
$E852 = AtoFACS
$E855 = ADMAN
$E874 = FR1
$E880 = FR2
$E88D = FR3
$E88F = FROUND
$E89D = RTN5
$E89E = NEGFAC
$E8A4 = NEG2
$E8C6 = PLUSEPS
$E8D4 = RTN6
$E8D5 = OVERFLOW
$E8DA = SHFTRES
$E8DC = NXSFT
$E8F0 = SHIFT
$E8FD = SH1
$E903 = SH2
$E907 = SHFTR
$E911 = SH3
$E913 = ONE
$E918 = LOGSER
$E92D = SQRhalf
$E932 = SQRtwo
$E937 = HALFneg
$E93C = LOGtwo
$E941 = LOG
$E948 = GIQ
$E94B = LG2
$E97F = FMULT
$E982 = FMULTT
$E987 = FMU
$E9B0 = FM1
$E9B5 = FM2
$E9B8 = FM3
$E9D4 = FM4
$E9E2 = RTN7
$E9E3 = CONUPK
$EA0E = ADEXP
$EA10 = ADEX2
$EA1B = ADEX3
$EA26 = ADEX4
$EA2B = OUTOFRNG
$EA31 = ZERO
$EA36 = JOV
$EA39 = MUL10
$EA4F = RTN8
$EA50 = NUM10
$EA55 = DIV10
$EA5E = DIV
$EA66 = FDIV
$EA69 = FDIVT
$EA80 = FD1
$EA96 = FD2

$EAA3 = FD3
$EAA6 = FD4
$EAB4 = FD5
$EAD1 = FD6
$EAD5 = FD7
$EAE1 = DIVZ
$EAE6 = RES>FAC
$EAF9 = MOVFM
$EB1E = MOV2F
$EB21 = MOV1F
$EB23 = MOVML
$EB27 = SETFOR
$EB2B = MOVMF
$EB53 = MOVFA
$EB55 = MFA
$EB59 = MFA2
$EB63 = MOVAF
$EB66 = MAF
$EB68 = MAF2
$EB71 = RTN9
$EB72 = RNDB
$EB7A = ROUND
$EB82 = SIGN
$EB86 = SIGN1
$EB88 = SIGN2
$EB8F = RTN10
$EB90 = SGN
$EB93 = FLOAT
$EB9B = FLO1
$EBA0 = FLO2
$EBAF = ABS
$EBB2 = FCOMP
$EBB4 = FCOMP2
$EBE9 = FC1
$EBEF = FC2
$EBF2 = QINT
$EC06 = QI1
$EC11 = RTN11
$EC12 = QI2
$EC23 = INT
$EC40 = ZFAC
$EC49 = RTN12
$EC4A = FIN
$EC4E = FIN2
$EC5D = FIN3
$EC61 = EVAL
$EC64 = NXDIGIT
$EC66 = CHKDP
$EC85 = SETSGN
$EC87 = DPDIG
$EC8A = COGEX
$EC8C = SGNCHK
$EC98 = SETDP
$EC9E = ADJEXP
$ECA0 = AEX
$ECA9 = DPLEFT
$ECB2 = DPRIGHT
$ECB9 = EVDONE
$ECBE = EVD
$ECC1 = INSRTDIG
$ECC8 = NDP
$ECD5 = ADDACC
$ECE8 = GETEXP
$ECF7 = MVDG
$ED05 = STEX
$ED0A = HMmiTNTH
$ED0F = BILmiONE
$ED14 = BILLION
$ED19 = INPRT
$ED24 = LINPRT
$ED2E = PRNTFAC
$ED31 = PRSTR
$ED34 = FOUT
$ED36 = FACSTRNG
$ED41 = SFSG
$ED4F = NOTZE
$ED57 = MB
$ED60 = STE
$ED62 = CMPBM1
$ED6D = CMPHM
$ED78 = JM10
$ED7F = JD10
$ED86 = ROUN
$ED89 = INTPART
$ED9E = DPLOC
$ED9F = DPL
$EDAA = PUTDP

$EDBB = SVY
$EDBD = MAKSTR
$EDC1 = MSLUP
$EDE5 = PARITY?
$EDE7 = COUNTED
$EDEE = MAKDIGIT
$EE09 = SAVY
$EE19 = MVBACK
$EE26 = NEEDEX?
$EE36 = PUTEX
$EE42 = WHATX
$EE57 = WNDUP
$EE5A = MARKEND
$EE5F = PNTSTK
$EE64 = HALF
$EE69 = DECTBL
$EE8D = SQR
$EE8D = TEND
$EE97 = FPWRT
$EEA0 = PW1
$EEBA = PW2
$EED0 = NEGOP
$EEDA = RTN13
$EEDB = LOGe
$EEE0 = EXPSER
$EF09 = EXP
$EF19 = X1
$EF24 = OOR
$EF27 = X2
$EF37 = X3
$EF5C = ODDSER
$EF72 = SERIES
$EF76 = SERMAIN
$EF85 = SS
$EF89 = SERLOOP
$EF96 = NXTERM
$EFA5 = RTN14
$EFA6 = RNDADJ1
$EFAA = RNDADJ2
$EFAE = RND
$EFCC = RD1
$EFE7 = RD2
$EFEA = COS
$EFF1 = SIN
$F023 = SI1
$F026 = SI2
$F033 = SI3
$F03A = TAN
$F062 = TAN2
$F066 = PIhalf
$F06B = PIdoub
$F070 = QUARTER
$F075 = SINSER
$F09E = ATN
$F0A6 = ATN1
$F0B4 = ATN2
$F0C7 = ATN3
$F0CD = RTN15
$F0CE = ATNSER
$F10B = ZPSTUFF
$F122 = RTN16
$F128 = COLDST
$F152 = MVZP
$F181 = FNDMEMHI
$F195 = MEMFOUND
$F1D5 = CALL
$F1DE = INNU
$F1E5 = PRNU
$F1EC = PLOTFNS
$F206 = GOERR
$F209 = LINCOOR
$F218 = AT?
$F225 = PLOT
$F232 = HLIN
$F241 = VLIN
$F24F = COLOR
$F256 = VTAB
$F262 = SPEED
$F26D = TRACE
$F26F = NOTRACE
$F273 = NORMAL
$F277 = INVERSE
$F279 = NRM
$F27B = SI
$F280 = FLASH
$F286 = HIMEM
$F296 = JMM

$F299 = SETHI
$F2A6 = LOMEM
$F2CB = ONERR
$F2E9 = HANDLERR
$F318 = RESUME
$F32E = JSYN
$F331 = DEL
$F365 = MOVDWN
$F367 = MOVIT
$F388 = NDX
$F390 = GR
$F399 = TEXT
$F39F = STORE
$F3AF = SLI
$F3BC = RECALL
$F3D2 = MOK
$F3D8 = HGR2
$F3E2 = HGR
$F3EA = SETHPG
$F3F2 = HCLR
$F3F6 = BKGND
$F3FE = BKGND1
$F411 = HPOSN
$F441 = HPOSN1
$F442 = HPOSN2
$F457 = HPLOT0
$F465 = LFTRT
$F467 = LEFT
$F46E = LR1
$F471 = LEFT1
$F476 = LEFT2
$F478 = NEWNDX
$F47C = CSHIFT
$F47E = CSHFT2
$F489 = RTS1
$F48A = RIGHT
$F49C = LRUDX1
$F49D = LRUDX2
$F4B3 = LRUD1
$F4B4 = LRUD2
$F4C4 = LRUD3
$F4C8 = LRUD4
$F4CD = EQ3
$F4D3 = UPDOWN
$F4D5 = UP
$F4EB = UP1
$F4F6 = UP5
$F4FB = UP2
$F4FD = UP3
$F4FF = UP4
$F501 = UPDWN1
$F505 = DOWN
$F508 = EQ4
$F524 = DOWN3
$F52A = DOWN1
$F52C = DOWN2
$F530 = HLINRL
$F53A = HGLIN
$F550 = HLIN2
$F568 = HLIN3
$F57C = MOVEX
$F581 = MOVEX2
$F58B = HCOUNT
$F59E = HLIN4
$F5B2 = MSKTBL
$F5B9 = EQ1C
$F5BA = COSTBL
$F5CB = HFIND
$F5F0 = HFIND1
$F5FE = HFIND2
$F600 = RTS2
$F601 = DRAW0
$F605 = DRAW1
$F626 = DRAW2
$F630 = DRAW3
$F63D = DRAW4
$F648 = DRAW5
$F65D = XDRAW0
$F661 = XDRAW1
$F682 = XDRAW2
$F68C = XDRAW3
$F699 = XDRAW4
$F6A4 = XDRAW5
$F6B4 = NI
$F6B9 = HFNS
$F6CD = HFNS1
$F6E6 = GGERR

Adressen-Labels $D000–$FFFF

```
$F6E9 = HCOLOR
$F6F5 = RTS3
$F6F6 = COLORTBL
$F6FE = HPLOT
$F7Ø8 = HP2
$F7ØF = HP3
$F721 = ROT
$F727 = SCALE
$F72D = DRWPNT
$F741 = DP1
$F747 = DP2
$F766 = DP3
$F769 = DRAW
$F76F = XDRAW
$F775 = SHLOAD
$F796 = SL1
$F7AØ = SL2
$F7A3 = SL3
$F7BC = TAPEPNT
$F7D9 = GETARYPT
$F7E7 = HTAB
$F7EC = HTAB1
$F7FA = HTAB2
$F8ØØ = MONPLOT
$F8ØØ = PLOT
$F8ØC = RTMASK
$F8ØE = PLOT1
$F819 = HLINE
$F81C = HLINE1
$F826 = VLINEZ
$F828 = VLINE
$F831 = RTS1
$F832 = CLRSCR
$F836 = CLRTOP
$F838 = CLRSC2
$F83C = CLRSC3
$F847 = GBASCALC
$F856 = GBCALC
$F85F = NXTCOL
$F864 = SETCOL
$F871 = SCR
$F871 = SCRN
$F879 = SCRN2
$F87F = RTMSKZ
$F882 = INSDS1
$F88E = INSDS2
$F89B = IEVEN
$F8A5 = ERR
$F8A9 = GETFMT
$F8BE = MNNDX1
$F8C2 = MNNDX2
$F8C9 = MNNDX3
$F8DØ = INSTDSP
$F8D4 = PRNTOP
$F8DB = PRNTBL
$F8F5 = PRMN1
$F8F9 = PRMN2
$F91Ø = PRADR1
$F914 = PRADR2
$F926 = PRADR3
$F92A = PRADR4
$F93Ø = PRADR5
$F938 = RELADR
$F94Ø = PRNTYX
$F941 = PRNTAX
$F944 = PRNTX
$F948 = PRBLNK
$F94A = PRBL2
$F94C = PRBL3
$F953 = PCADJ
$F954 = PCADJ2
$F956 = PCADJ3
$F95C = PCADJ4
$F961 = RTS2
$F962 = FMT1
$F9A6 = FMT2
$F9B4 = CHAR1
$F9BA = CHAR2
$F9CØ = MNEML
$FAØØ = MNEMR
$FA43 = STEP
$FA4E = XQINIT
$FA78 = XQ1
$FA7A = XQ2
$FA86 = IRQ
$FA92 = BREAK
$FA9C = XBRK
$FAA5 = XRTI
$FAA9 = XRTS
$FAAD = PCINC2
$FAAF = PCINC3
$FAB9 = XJSR
$FAC4 = XJMP
$FAC5 = XJMPAT
$FACD = NEWPCL
$FAD1 = RTNJMP
$FAD7 = REGDSP
$FADA = RGDSP1
$FAE4 = RDSP1
$FAFD = BRANCH
$FBØB = NBRNCH
$FB11 = INITBL
$FB19 = RTBL
$FB1E = PREAD
$FB25 = PREAD2
$FB2E = RTS2D
$FB2F = INIT
$FB39 = SETTXT
$FB4Ø = SETGR
$FB4B = SETWND
$FB5B = TABV
$FB6Ø = MULPM
$FB63 = MUL
$FB65 = MUL2
$FB6D = MUL3
$FB76 = MUL4
$FB78 = MUL5
$FB81 = DIVPM
$FB84 = DIV
$FB86 = DIV2
$FBAØ = DIV3
$FBA4 = MD1
$FBAF = MD2
$FBB4 = MD3
$FBCØ = MDRTS
$FBC1 = BASCALC
$FBDØ = BSCLC2
$FBD9 = BELL1
$FBE4 = BELL2
$FBEF = RTS2B
$FBFØ = STOADV
$FBF4 = ADVANCE
$FBFC = RTS3
$FBFD = VIDOUT
$FC1Ø = BS
$FC1A = UP
$FC22 = VTAB
$FC24 = VTABZ
$FC2B = RTS4
$FC2C = ESC1
$FC42 = CLREOP
$FC46 = CLEOP1
$FC58 = HOME
$FC62 = CR
$FC66 = LF
$FC7Ø = SCROLL
$FC76 = SCRL1
$FC8C = SCRL2
$FC95 = SCRL3
$FC9C = CLREOL
$FC9E = CLEOLZ
$FCAØ = CLEOL2
$FCA8 = MONWAIT
$FCA8 = WAIT
$FCA9 = WAIT2
$FCAA = WAIT3
$FCB4 = NXTA4
$FCBA = NXTA1
$FCC8 = RTS4B
$FCC9 = HEADR
$FCD6 = WRBIT
$FCDB = ZERDLY
$FCE2 = ONEDLY
$FCE5 = WRTAPE
$FCEC = RDBYTE
$FCEE = RDBYT2
$FCFA = RD2BIT
$FCFD = RDBIT
$FDØC = RDKEY
$FD1B = KEYIN
$FD21 = KEYIN2
$FD2F = ESC
$FD35 = RDCHAR
$FD3D = NOTCR
$FD5F = NOTCR1
$FD62 = CANCEL
$FD67 = GETLNZ
$FD6A = GETLN
$FD71 = BCKSPC
$FD75 = NXTCHAR
$FD7E = CAPTST
$FD84 = ADDINP
$FD8E = CROUT
$FD92 = PRA1
$FD96 = PRYX2
$FDA3 = XAM8
$FDAD = MOD8CHK
$FDB3 = XAM
$FDB6 = DATAOUT
$FDC5 = RTS4C
$FDC6 = XAMPM
$FDD1 = ADD
$FDDA = PRBYTE
$FDE3 = PRHEX
$FDE5 = PRHEXZ
$FDED = COUT
$FDFØ = COUT1
$FDF6 = COUTZ
$FEØØ = BL1
$FE04 = BLANK
$FEØB = STOR
$FE17 = RTS5
$FE18 = SETMODE
$FE1D = SETMDZ
$FE2Ø = LT
$FE22 = LT2
$FE2C = MOVE
$FE36 = VFY
$FE58 = VFYOK
$FE5E = LIST
$FE63 = LIST2
$FE75 = A1PC
$FE78 = A1PCLP
$FE7F = A1PCRTS
$FE8Ø = SETINV
$FE84 = SETNORM
$FE86 = SETIFLG
$FE89 = SETKBD
$FE8B = INPORT
$FE8D = INPRT
$FE93 = SETVID
$FE95 = OUTPORT
$FE97 = OUTPRT
$FE9B = IOPRT
$FEA7 = IOPRT1
$FEA9 = IOPRT2
$FEBØ = XBASIC
$FEB3 = BASCONT
$FEB6 = GO
$FEBF = REGZ
$FEC2 = TRACE
$FEC4 = STEPZ
$FECA = USR
$FECD = WRITE
$FED4 = WR1
$FEED = WRBYTE
$FEEF = WRBYT2
$FEF6 = CRMON
$FEFD = MONREAD
$FEFD = READ
$FFØ2 = MONREAD2
$FF0A = RD2
$FF16 = RD3
$FF2D = PRERR
$FF3A = BELL
$FF3F = RESTORE
$FF44 = RESTR1
$FF4A = SAVE
$FF4C = SAV1
$FF59 = RESET
$FF65 = MON
$FF69 = MONZ
$FF73 = NXTITM
$FF7A = CHRSRCH
$FF8A = DIG
$FF9Ø = NXTBIT
$FF98 = NXTBAS
$FFA2 = NXTBS2
$FFA7 = GETNUM
$FFAD = NXTCHR
$FFBE = TOSUB
$FFC7 = ZMODE
$FFCC = CHRTBL
$FFE3 = SUBTBL
```

Labels-Adressen $D000–$FFFF

```
A1H      = $ØØ3D
A1L      = $ØØ3C
A1PC     = $FE75
A1PCLP   = $FE78
A1PCRTS  = $FE7F
A2H      = $ØØ3F
A2L      = $ØØ3E
A3H      = $ØØ41
A3L      = $ØØ4Ø
A4H      = $ØØ43
A4L      = $ØØ42
A5H      = $ØØ45
A5L      = $ØØ44
ABS      = $EBAF
ACC      = $ØØ45
ACH      = $ØØ51
ACL      = $ØØ5Ø
ADØ      = $E7B9
AD1      = $E7C6
AD2      = $E7CE
AD3      = $E7EA
AD4      = $E7EE
AD5      = $E7FA
ADD      = $FDD1
ADDACC   = $ECD5
ADDINP   = $FD84
ADDON    = $D998
ADEX2    = $EA1Ø
ADEX3    = $EA1B
ADEX4    = $EA26
ADEXP    = $EAØE
ADJEXP   = $EC9E
ADMAN    = $E855
ADVANCE  = $FBF4
AEX      = $ECAØ
AMPER    = $Ø3F5
AND      = $DF55
ARG      = $ØØA5
ARGSGN   = $ØØAA
ARITH    = $DDC5
ARRAY    = $E11E
ARYFOUND = $E19E
ARYLOOP  = $E16D
ARYNAM?  = $E179
ARYPNT   = $ØØ94
ARYSTR   = $E5ØC
ARYTAB   = $ØØ6B
ARYVA2   = $E4CA
ARYVA3   = $E4CE
ARYVAR   = $E4C2
ARYVGO   = $E4D9
ASC      = $E6E5
ASCHEX   = $DA12
AT?      = $F218
ATN      = $FØ9E
ATN1     = $FØA6
ATN2     = $FØB4
ATN3     = $FØC7
ATNSER   = $FØCE
AUXH     = $ØØ55
AUXL     = $ØØ54
AYINT    = $E1ØC
AtoFAC   = $E85Ø
AtoFACS  = $E852
BADNAM   = $DFF4
BADSUBS  = $D2CB
BAS2H    = $ØØ2B
BAS2L    = $ØØ2A
BASCALC  = $FBC1
BASCONT  = $FEB3
BASH     = $ØØ29
BASIC    = $EØØØ
BASIC2   = $EØØ3
BASL     = $ØØ28
BCKSPC   = $FD71
BELL     = $FF3A
BELL1    = $FBD9
BELL2    = $FBE4
BILLION  = $ED14
BILmiONE = $ED9F
BKGND    = $F3F6
BKGND1   = $F3FE
BL1      = $FEØØ
BLANK    = $FE04
BLTU     = $D393
BLTU2    = $D39A
BPD      = $E67F
BRANCH   = $FAFD
BREAK    = $FA92
BREAKIN  = $D35D
BS       = $FC1Ø
BSB      = $EØ49
BSCLC2   = $FBDØ
BYPASS   = $EØ12
CALL     = $F1D5
CANCEL   = $FD62
CANTCON  = $D332
CAPTST   = $FD7E
CAT      = $E597
CH       = $ØØ24
CHAR1    = $F9B4
CHAR2    = $F9BA
CHARAC   = $ØØØD
CHKCLS   = $DEB8
CHKCOM   = $DEBE
CHKDIM   = $E1AA
CHKDP    = $EC66
CHKMEM   = $D3D6
CHKNUM   = $DD6A
CHKOPN   = $DEBB
CHKSTR   = $DD6C
CHKSUM   = $ØØ2E
CHKTYP   = $DDB4
CHKVAL   = $DD6D
CHRGET   = $ØØB1
CHRGOT   = $ØØB7
CHRSRCH  = $FF7A
CHRSTR   = $E646
CHRTBL   = $FFCC
CLEAR    = $D66A
CLEARC   = $D66C
CLEOL2   = $FCAØ
CLEOLZ   = $FC9E
CLEOP1   = $FC46
CLREOL   = $FC9C
CLREOP   = $FC42
CLRSC2   = $F838
CLRSC3   = $F83C
CLRSCR   = $F832
CLRTOP   = $F836
CMDTABL  = $DØØØ
CMPBM1   = $ED62
CMPDONE  = $DFC1
CMPHM    = $ED6D
CMPLOOP  = $DFAA
COLCOUNT = $ØØEA
COLDST   = $F128
COLON?   = $D842
COLOR    = $ØØ3Ø
COLOR    = $F24F
COLORTBL = $F6F6
COMBYTE  = $E74C
COMPARE  = $DDE4
CON      = $D8A1
CONINT   = $E6FB
CONT     = $D896
CONUPK   = $E9E3
COPSTR   = $DA9A
COPY     = $DAB7
COS      = $EFEA
COSTBL   = $F5BA
COUNTED  = $EDE7
COUNTH   = $ØØ1D
COUT     = $FDED
COUT1    = $FDFØ
COUTZ    = $FDF6
CPRMASK  = $ØØ16
CPROP    = $DD98
CPRTYP   = $ØØ89
CR       = $FC62
CRDO     = $DAFB
CRMON    = $FEF6
CROUT    = $FD8E
CSHFT2   = $F47E
CSHIFT   = $F47C
CSWH     = $ØØ37
CSWL     = $ØØ36
CTRC?    = $D86C
CURLIN   = $ØØ75
CURLSV   = $ØØF6
CV       = $ØØ25
CV2      = $DD74
CY       = $E45F
DATA     = $D995
DATAFLG  = $ØØ13
DATAN    = $D9A3
DATAOUT  = $FDB6
DATIN    = $DC69
DATLIN   = $ØØ7B
DATPTR   = $ØØ7D
DECTBL   = $EE69
DEF      = $E313
DEL      = $F331
DESC?    = $DA8C
DEST     = $ØØ6Ø
DFLTDIM  = $E1F7
DIG      = $FF8A
DIM      = $DFD9
DIMFLG   = $ØØ1Ø
DIMLUP   = $E253
DIMOK    = $E26F
DIMOK2   = $E27Ø
DIR?     = $DBC7
DIRCT    = $D7E5
DIV      = $EA5E
DIV      = $FB84
DIV1Ø    = $EA55
DIV2     = $FB86
DIV3     = $FBAØ
DIVPM    = $FB81
DIVZ     = $EAE1
DIVbyZRO = $D2E5
DMTH     = $DE41
DOCMP    = $DFB5
DOERRMSG = $D419
DOMATH   = $DE43
DOMTH    = $DE3A
DONE     = $D61Ø
DOREENT  = $DB9Ø
DOSPC    = $DB35
DOWN     = $F5Ø5
DOWN1    = $F52A
DOWN2    = $F52C
DOWN3    = $F524
DP1      = $F741
DP2      = $F747
DP3      = $F766
DPDIG    = $EC87
DPFLG    = $ØØ9B
DPL      = $ED9F
DPLEFT   = $ECA9
DPLOC    = $ED9E
DPRIGHT  = $ECB2
DRAW     = $F769
DRAWØ    = $F6Ø1
DRAW1    = $F6Ø5
DRAW2    = $F626
DRAW3    = $F63Ø
DRAW4    = $F63D
DRAW5    = $F648
DRWPNT   = $F72D
DSCLEN   = $ØØ8F
DSCPTR   = $ØØ8C
DSCTMP   = $ØØ9D
DV1      = $E538
DV2      = $E542
DVAR     = $E523
DVARS    = $E519
DVARTS   = $E552
DXH      = $ØØD1
DXL      = $ØØDØ
DY       = $ØØD2
EH       = $ØØD5
EL       = $ØØD4
END      = $D87Ø
END2     = $D871
END3     = $D888
END4     = $D88A
ENDCHR   = $ØØØE
ENDFOR   = $DD55
ENDRNG   = $D6C4
```

Labels-Adressen $D000–$FFFF

```
EQ1C     = $F5B9
EQ3      = $F4CD
EQ4      = $F5Ø8
EQUAL    = $DØCA
EQUL     = $DEDØ
EQUOP    = $DE98
ERFLG?   = $D863
ERLIN    = $DB7F
ERLUP    = $D41F
ERR      = $F8A5
ERRDIR   = $E3Ø6
ERRFLG   = $ØØD8
ERRIN    = $D35Ø
ERRLIN   = $ØØDA
ERRMSG   = $D26Ø
ERRNUM   = $ØØDE
ERROR    = $D412
ERRPOS   = $ØØDC
ERRSTK   = $ØØDF
ESC      = $FD2F
ESC1     = $FC2C
EVAL     = $EC61
EVD      = $ECBE
EVDONE   = $ECB9
EXIG     = $DCDF
EXIT     = $DE5D
EXP      = $EF9Ø
EXPON    = $ØØ9A
EXPSER   = $EEEØ
EXPSGN   = $ØØ9C
EXTRAFAC = $ØØAC
EXTRASV  = $ØØ92
FAC      = $ØØ9D
FACSGN   = $ØØA2
FACSTRNG = $ED36
FADD     = $E7BE
FADDH    = $E7AØ
FADDT    = $E7C1
FALSE    = $DF5D
FC1      = $EBE9
FC2      = $EBEF
FCOMP    = $EBB2
FCOMP2   = $EBB4
FD1      = $EA8Ø
FD2      = $EA96
FD3      = $EAA3
FD4      = $EAA6
FD5      = $EAB4
FD6      = $EAD1
FD7      = $EAD5
FDIV     = $EA66
FDIVT    = $EA69
FE1      = $E415
FE2      = $E41F
FEND     = $E3F7
FEVLOOP  = $DD86
FIN      = $EC4A
FIN2     = $EC4E
FIN3     = $EC5D
FINDATA  = $DCAØ
FINDEOL  = $D511
FIRST    = $ØØFØ
FL1      = $D61E
FL2      = $D635
FLASH    = $F28Ø
FLO1     = $EB9B
FLO2     = $EBAØ
FLOAT    = $EB93
FLOOP    = $E832
FM1      = $E9BØ
FM2      = $E9B5
FM3      = $E9B8
FM4      = $E9D4
FMT1     = $F962
FMT2     = $F9A6
FMU      = $E987
FMULT    = $E97F
FMULTT   = $E982
FN?      = $DEA4
FNC?     = $E341
FNCDATA  = $E3AF
FNCNAM   = $ØØ8A
FNDARY   = $E169
FNDELEM  = $E24B
FNDFOR   = $D36A
FNDLIN   = $D61A
FNDMEMHI = $F181
FNDVAR   = $E488
FOR      = $D766
FOR2     = $D777
FORMAT   = $ØØ2E
FORMtoCX = $D31F
FORPNT   = $ØØ85
FOUT     = $ED34
FPGEN    = $ØØA4
FPWRT    = $EE97
FR1      = $E874
FR2      = $E88Ø
FR3      = $E88D
FRE      = $E2DE
FRE2     = $E2E5
FREFAC   = $E6ØØ
FRESPC   = $ØØ71
FRESTR   = $E5FD
FRETMP   = $E6Ø4
FRETMS   = $E635
FRETOP   = $ØØ6F
FRMEVL   = $DD7B
FRMEVL2  = $DD95
FRMNUM   = $DD67
FROUND   = $E88F
FSUB     = $E7A7
FSUBT    = $E7AA
FULL     = $E474
FUNCT    = $E354
GARBAG   = $E484
GARFLG   = $ØØ13
GBASCALC = $F847
GBASH    = $ØØ27
GBASL    = $ØØ26
GBCALC   = $F856
GC       = $D731
GD       = $EØF9
GDBUFS   = $D539
GERR     = $DDØD
GET      = $DBAØ
GETADR   = $E752
GETARY   = $EØED
GETARY2  = $EØEF
GETARYPT = $F7D9
GETBYT   = $E6F8
GETCHR   = $D72C
GETEXP   = $ECE8
GETFMT   = $F8A9
GETIN    = $D559
GETLINK  = $D63E
GETLN    = $FD6A
GETLNZ   = $FD67
GETNUM   = $FFA7
GETOLD   = $E3A9
GETSPA   = $E452
GETSPC   = $E454
GETSTR   = $E6DC
GETVAL   = $DE6Ø
GGERR    = $F6E6
GIQ      = $E948
GIVAYF   = $E2F2
GK       = $D86Ø
GME      = $E26C
GO       = $FEB6
GO1      = $D955
GO2      = $D959
GOCMD    = $D828
GOCMD2   = $D82A
GOEND    = $D826
GOERR    = $F2Ø6
GOEX     = $DE38
GOGEX    = $EC8A
GOGO     = $E514
GOIQ     = $E6F2
GOLINE   = $D935
GONEWST  = $DD52
GOROUT   = $DF3F
GOSTART  = $D893
GOSTROUT = $ØØØ3
GOSUB    = $D921
GOTFOR   = $DDØF
GOTO     = $D93E
GOTO?    = $D9F4
GOTSPA   = $E469
GOWARM   = $ØØØØ
GR       = $F39Ø
GRBPAS   = $E562
GSE      = $E269
GSYNER   = $D981
GTBYTC   = $E6F5
GTFORPNT = $D365
GTLT     = $E911
GTNUM    = $E746
H2       = $ØØ2C
HALF     = $EE64
HALFneg  = $E937
HANDLERR = $F2E9
HCLR     = $F3F2
HCOLOR   = $F6E9
HCOLOR1  = $ØØ1C
HCOLORZ  = $ØØE4
HCOUNT   = $F58B
HEADR    = $FCC9
HFIND    = $F5CB
HFIND1   = $F5FØ
HFIND2   = $F5FE
HFNS     = $F6B9
HFNS1    = $F6CD
HGLIN    = $F53A
HGR      = $F3E2
HGR2     = $F3D8
HIGHDS   = $ØØ94
HIGHTR   = $ØØ96
HIMEM    = $F286
HIRES    = $C057
HISCR    = $C055
HLIN     = $F232
HLIN2    = $F55Ø
HLIN3    = $F568
HLIN4    = $F59E
HLINE    = $F819
HLINE1   = $F81C
HLINRL   = $F53Ø
HMASK    = $ØØ3Ø
HMmiTNTH = $EDØA
HNDX     = $ØØE5
HOME     = $FC58
HP2      = $F7Ø8
HP3      = $F7ØF
HPAG     = $ØØE6
HPLOT    = $F6FE
HPLOTØ   = $F457
HPOSN    = $F411
HPOSN1   = $F441
HPOSN2   = $F442
HTAB     = $F7E7
HTAB1    = $F7EC
HTAB2    = $F7FA
IEVEN    = $F89B
IF       = $D9C9
IFTRUE   = $D9E1
ILLDIR   = $D2F5
ILLQUAN  = $D295
IN       = $Ø2ØØ
INCHR    = $D553
INDEX    = $ØØ5E
INDX     = $ØØ99
INIT     = $FB2F
INITBL   = $FB11
INLIN    = $D52C
INLIN2   = $D52E
INMSG    = $D358
INNU     = $F1DE
INPDONE  = $DCC6
INPERR   = $DB86
INPFIN   = $DC99
INPORT   = $FE8B
INPRT    = $ED19
INPRT    = $FE8D
INPTR    = $ØØ7F
INPUT    = $DBB2
INPUTERR = $DB71
INPUTFLG = $ØØ15
INS1     = $E66Ø
INS2     = $E666
INS3     = $E667
INS4     = $E668
INS5     = $E6A2
INSDS1   = $F882
INSDS2   = $F88E
INSRTDIG = $ECC1
INSRTLIN = $D4EA
INSTART  = $DC2B
INSTDSP  = $F8DØ
INSTRNG  = $E6B9
INT      = $EC23
INTPART  = $ED89
INTVAR?  = $EØ26
INVERSE  = $F277
INVFLG   = $ØØ32
IOADR    = $CØØØ
IOPRT    = $FE9B
IOPRT1   = $FEA7
IOPRT2   = $FEA9
IQERR    = $E199
IRQ      = $FA86
IRQLOC   = $Ø3FE
ISCNTC   = $D858
ISLETC   = $EØ7D
ISTOK?   = $D59Ø
JARY     = $EØ46
JD1Ø     = $ED7F
JER      = $E19B
JERR     = $E432
JERROR   = $DD78
JF       = $DFCA
JGOCMD   = $D9E9
JLNK     = $D8ED
JM1Ø     = $ED78
JMM      = $F296
JMPADRS  = $ØØ9Ø
JOV      = $EA36
JSY      = $D846
JSYN     = $F32E
KBD      = $CØØØ
KBDSTRB  = $C01Ø
KEY      = $CØØØ
KEYIN    = $FD1B
KEYIN2   = $FD21
KSWH     = $ØØ39
KSWL     = $ØØ38
LASTIN   = $ØØ2F
LASTOP   = $ØØ87
LASTPT   = $ØØ53
LEFT     = $F467
LEFT1    = $F471
LEFT2    = $F476
LEFTSTR  = $E65A
LEN      = $E6D6
LENGTH   = $ØØ2F
LENGTH   = $ØØ91
LET      = $DA46
LET2     = $DA63
LETREAL  = $DA77
LETSTR   = $DA7A
LF       = $FC66
LFTRT    = $F465
LG2      = $E94B
LIN      = $D5A8
LINCOOR  = $F2Ø9
LINGET   = $DAØC
LINKSET  = $D4F2
LINNUM   = $ØØ5Ø
LINPRT   = $ED24
LIST     = $D6A5
LIST     = $FE5E
LIST2    = $FE63
LISTED   = $D724
LISTLOOP = $D6FE
LMNEM    = $ØØ2C
LOAD     = $D8C9
LOCØ     = $ØØØØ
LOC1     = $ØØØ1
LOCK     = $ØØD6
LOG      = $E941
LOGSER   = $E918
LOGe     = $EEDB
LOGtwo   = $E93C
LOMEM    = $F2A6
LORES    = $CØ56
LOWSCR   = $CØ54
LOWTR    = $ØØ9B
LR1      = $F46E
LRUD1    = $F4B3
LRUD2    = $F4B4
LRUD3    = $F4C4
LRUD4    = $F4C8
LRUDX1   = $F49C
LRUDX2   = $F49D
LST1LIN  = $D6F7
LSTD?    = $D6F5
LT       = $FE2Ø
LT2      = $FE22
MAF      = $EB66
MAF2     = $EB68
MAININP  = $DBEB
MAINLST  = $D6CC
MAKARY   = $E1C1
MAKDIGIT = $EDEE
MAKINT   = $E1Ø2
MAKSTR   = $EDBD
MARKEND  = $EE5A
MASK     = $ØØ2E
MATHTBL  = $DØB2
MB       = $ED57
MD1      = $FBA4
MD2      = $FBAF
MD3      = $FBB4
MDRTS    = $FBCØ
MEMERR   = $D41Ø
MEMFOUND = $F195
MEMSIZ   = $ØØ73
MFA      = $EB55
MFA2     = $EB59
MI1      = $E119
MI2      = $E11B
MIDSTR   = $E691
MIN      = $DECE
MINUS    = $DØC7
MISMTCH  = $DD76
MIXCLR   = $CØ52
MIXSET   = $CØ53
MKINT    = $E1Ø8
MNEML    = $F9CØ
MNEMR    = $FAØØ
MNNDX1   = $F8BE
MNNDX2   = $F8C2
MNNDX3   = $F8C9
MOD8CHK  = $FDAD
MODE     = $ØØ31
MOK      = $F3D2
MON      = $FF65
MONPLOT  = $F8ØØ
MONREAD  = $FEFD
MONREAD2 = $FFØ2
MONWAIT  = $FCA8
MONZ     = $FF69
MORNAM   = $EØØ7
MOV1F    = $EB21
MOV2F    = $EB1E
MOVAF    = $EB63
MOVDWN   = $F365
MOVE     = $FE2C
MOVESTR  = $E5E6
MOVEX    = $F57C
MOVEX2   = $F581
MOVFA    = $EB53
MOVFM    = $EAF9
MOVINS   = $E5D4
MOVIT    = $F367
MOVMF    = $EB2B
MOVML    = $EB23
MOVSTR   = $E5E2
MSKTBL   = $F5B2
MSLUP    = $EDC1
MU1      = $E2B6
MU2      = $E2CØ
MU3      = $E2D9
MUL      = $FB63
MUL1Ø    = $EA39
MUL2     = $FB65
MUL3     = $FB6D
MUL4     = $FB76
MUL5     = $FB78
MULPM    = $FB6Ø
MULT     = $E2AD
MVBACK   = $EE19
MVBYT    = $D3C3
MVDG     = $ECF7
MVDWN    = $D4A7
MVPRG    = $D4D1
MVS2     = $E5EA
MVS3     = $E5F3
MVZP     = $F152
NAMOK    = $DFF7
NB       = $E63Ø
NBRNCH   = $FBØB
NCR      = $D712
ND       = $DDEE
NDP      = $ECC8
NDX      = $F388
NEEDEX?  = $EE26
NEG2     = $E8A4
NEGATE   = $DBØØ
NEGFAC   = $E89E
NEGNUM   = $EØFE
NEGOP    = $EED0
NEW      = $D649
NEWDESC  = $DAA1
NEWLN?   = $D4B5
NEWNDX   = $F478
NEWPCL   = $FACD
NEWSTT   = $D7D2
NEWVAR   = $E09C
NEXT     = $DCF9
NEXTCHR  = $DB2F
NI       = $F6B4
NIN      = $E036
NINT     = $E1D5
NINTA    = $E292
NL1      = $D49F
NMI      = $Ø3FB
NOI      = $D54C
NORMAL   = $F273
NOSUCH   = $D647
NOT?     = $DE9Ø
NOTCR    = $FD3D
NOTCR1   = $FD5F
NOTFND   = $EØ87
NOTFOUND = $E1B8
NOTMATH  = $DE35
NOTOK    = $D83F
NOTRACE  = $F26F
NOTZ     = $DE9F
NOTZE    = $ED4F
NOTinstr = $DF3A
NRM      = $F279
NTD      = $DCD1
NTL      = $E5B7
NUM1Ø    = $EA5Ø
NUMBER   = $DE69
NUMCMP   = $DFBØ
NUMDIM   = $ØØØF
NUMIN    = $DC63
NV1      = $E05B
NV2      = $E065
NWV      = $EØB2
NX       = $D5A7
NXARY    = $E188
NXBYT    = $D3C7
NXCHAR   = $DB44
NXCHR    = $D56D
NXCLM    = $DBØE
NXDIG    = $DA4Ø
NXDIGIT  = $EC64
NXDIM    = $DFD6
NXDM     = $E281
NXFOR    = $D38B
NXIN     = $DBDC
NXINP    = $DBF1
NXLIN    = $D45C
NXLINK   = $D4FE
NXLST    = $D6DA
NXNUM    = $DAØØ
NXOP     = $DDD6
NXPAG    = $D3CE
NXPG     = $E22E
NXPTR    = $EØ73
NXS      = $DCB9
NXSFT    = $E8DC
NXSPC    = $DB2C
NXTA1    = $FCBA
NXTA4    = $FCB4
NXTBAS   = $FF98
NXTBIT   = $FF9Ø
NXTBS2   = $FFA2
NXTCHAR  = $FD75
NXTCHR   = $FFAD
NXTCOL   = $F85F
NXTDIM   = $E12C
NXTERM   = $EF96
NXTITM   = $FF73
NXVAR    = $EØ59
NXwoFOR  = $D26Ø
NY       = $D5A2
NZ       = $E4Ø9
```

Labels-Adressen $D000–$FFFF

```
ODDSER  = $EF5C
OLDLIN  = $0077
OLDTEXT = $0079
ONCNT   = $D9F8
ONE     = $E913
ONEDLY  = $FCE2
ONERR   = $F2CB
ONESTEP = $D7C3
ONGOTO  = $D9EC
OOR     = $EF24
OR      = $DF4F
ORMASK  = $00F3
OUTDO   = $DB5C
OUTOFRNG = $EA2B
OUTPORT = $FE95
OUTPRT  = $FE97
OUTQUES = $DB5A
OUTSP   = $DB57
OVERFLOW = $E8D5
OVFLOW  = $D2A5
OofDATA = $D28A
OofMEM  = $D2AD
PADDLØ  = $C064
PARCHK  = $DEB2
PARITY? = $EDE5
PARSE   = $D56C
PCADJ   = $F953
PCADJ2  = $F954
PCADJ3  = $F956
PCADJ4  = $F95C
PCH     = $003B
PCINC2  = $FAAD
PCINC3  = $FAAF
PCL     = $003A
PDL     = $DFCD
PECHR   = $DC4C
PEEK    = $E764
PENCHR  = $DC4B
PICK    = $0095
PIdoub  = $F06B
PIhalf  = $F066
PLOT    = $F225
PLOT    = $F80D
PLOT1   = $F80E
PLOTFNS = $F1EC
PLU?    = $D604
PLUS    = $D0CD
PLUSEPS = $E8C6
PNTR    = $000F
PNTSTK  = $EE5F
POINT   = $E73D
POKE    = $E77B
POP     = $D96B
POS     = $E2FF
POSOP   = $DF65
PRA1    = $FD92
PRADR1  = $F91Ø
PRADR2  = $F914
PRADR3  = $F926
PRADR4  = $F92A
PRADR5  = $F93Ø
PRBL2   = $F94A
PRBL3   = $F94C
PRBLNK  = $F948
PRBYTE  = $FDDA
PREAD   = $FB1E
PREAD2  = $FB25
PREFNC  = $DDF6
PREFTEST = $DDCD
PRERR   = $FF2D
PRGEND  = $00AF
PRHEX   = $FDE3
PRHEXZ  = $FDE5
PRINT   = $DAD5
PRINT2  = $DAD7
PRMN1   = $F8F5
PRMN2   = $F8F9
PRNTAX  = $F941
PRNTBL  = $F8DB
PRNTFAC = $ED2E
PRNTIN? = $D431
PRNTOP  = $F8D4
PRNTX   = $F944
PRNTYX  = $F94Ø
PRNU    = $F1E5
PROGIO  = $D901
PROMPT  = $0033
PRSTR   = $ED31
PRSTRING = $DACF
PRTOK   = $D75Ø
PRYX2   = $FD96
PSHF    = $DE1Ø
PSHFACX = $DE15
PSHMAD  = $DDFD
PTRGET  = $DFE3
PTRGET2 = $DFE8
PTRGET3 = $DFEA
PTRIG   = $C07Ø
PULL3   = $D9C5
PUSHFAC = $DE2Ø
PUTCHR  = $DC3F
PUTDP   = $EDAA
PUTEMP  = $E435
PUTEX   = $EE36
PUTIN   = $D5CD
PUTLINK = $D50F
PUTNEW  = $E42A
PUTSTR  = $DA7B
PUTTOK  = $D5CB
PW1     = $EEAØ
PW2     = $EEBA
QDRNT   = $00D3
QI1     = $EC06
QI2     = $EC12
QINT    = $EBF2
QOUT    = $DBC4
QUARTER = $F07Ø
QUO?    = $E404
RAR     = $E1DE
RARY    = $E298
RD1     = $EFCC
RD2     = $EFE7
RD2     = $FF0A
RD2BIT  = $FCFA
RD3     = $FF16
RDBIT   = $FCFD
RDBYT2  = $FCEE
RDBYTE  = $FCEC
RDCHAR  = $FD35
RDKEY   = $FD0C
RDSP1   = $FAE4
READ    = $DBE2
READ    = $FEFD
READERR = $DB7B
REASON  = $D3E3
RECALL  = $F3BC
REENT   = $DCEF
REGDSP  = $FAD7
REGZ    = $FEBF
RELADR  = $F938
REM     = $D9DC
REM?    = $D5E2
REMN    = $D9A6
REMSTK  = $00F8
RES>FAC = $EAE6
RESET   = $FF59
RESPERR = $DB87
RESTART = $D43C
RESTORE = $D849
RESTORE = $FF3F
RESTR1  = $FF44
RESULT  = $0062
RESUME  = $F318
RET1    = $D392
RET1Ø   = $DB02
RET11   = $DCDE
RET12   = $DD73
RET2    = $D40F
RET3    = $D648
RET4    = $D696
RET5    = $D857
RET6    = $D8AF
RET7    = $D96A
RET8    = $D9A2
RET9    = $DA0B
RETURN  = $D984
REdimARR = $D2D8
RGDSP1  = $FADA
RIGHT   = $F48A
RIGHTSTR = $E686
RM1     = $D9AE
RM2     = $D9B6
RMNEM   = $002D
RND     = $EFAE
RNDADJ1 = $EFA6
RNDADJ2 = $EFAA
RNDB    = $EB72
RNDH    = $004F
RNDL    = $004E
RNDSEED = $00C9
ROT     = $F721
ROTZ    = $00F9
ROUN    = $ED86
ROUND   = $EB7A
RS1     = $D3ED
RS2     = $D3F1
RS3     = $D3FC
RTBL    = $FB19
RTMASK  = $F80C
RTMSKZ  = $F87F
RTN1    = $E086
RTN1Ø   = $EB8F
RTN11   = $EC11
RTN12   = $EC49
RTN13   = $EEDA
RTN14   = $EFA5
RTN15   = $F0CD
RTN16   = $F122
RTN2    = $E2AC
RTN3    = $E645
RTN4    = $E79F
RTN5    = $E89D
RTN6    = $E8D4
RTN7    = $E9E2
RTN8    = $EA4F
RTN9    = $EB71
RTNH    = $002D
RTNJMP  = $FAD1
RTNL    = $002C
RTNwoGSB = $D276
RTS1    = $F489
RTS1    = $F831
RTS2    = $F600
RTS2    = $F961
RTS2B   = $FBEF
RTS2D   = $FB2E
RTS3    = $F6F5
RTS3    = $FBFC
RTS4    = $FC2B
RTS4B   = $FCC8
RTS4C   = $FDC5
RTS5    = $FE17
RUN     = $D912
RUNLINE = $D91B
SAMEFOR? = $D37F
SAV1    = $FF4C
SAVDIM  = $E1E7
SAVE    = $D8B0
SAVE    = $FF4A
SAVOLD  = $E378
SAVOP   = $DDD7
SAVY    = $EE09
SCALE   = $F727
SCALEZ  = $00E7
SCDCH   = $E03D
SCR     = $F871
SCREEN  = $DEF9
SCRL1   = $FC76
SCRL2   = $FC8C
SCRL3   = $FC95
SCRN    = $F871
SCRN2   = $F879
SCROLL  = $FC70
SCRTCH  = $D64B
SE      = $D578
SEND    = $DB64
SENDCHR = $D702
SERIES  = $EF72
SERLEN  = $00A3
SERLOOP = $EF89
SERMAIN = $EF76
SERPNT  = $00AD
SETCOL  = $F864
SETDA   = $D853
SETDP   = $EC98
SETEND  = $D3B7
SETFOR  = $EB27
SETGR   = $FB40
SETHI   = $F299
SETHPG  = $F3EA
SETIFLG = $FE86
SETINV  = $FE80
SETKBD  = $FE89
SETMDZ  = $FE1D
SETMODE = $FE18
SETNORM = $FE84
SETPTRS = $D665
SETSGN  = $EC85
SETTXT  = $FB39
SETVID  = $FE93
SETVPNT = $E0DE
SETWND  = $FB4B
SFS     = $DFA5
SFSG    = $ED41
SGN     = $EB90
SGN?    = $DEAB
SGNCHK  = $EC8C
SGNCPR  = $00AB
SGNIF   = $E829
SH1     = $E8FD
SH2     = $E903
SH3     = $E911
SHAPEH  = $001B
SHAPEL  = $001A
SHAPEPNT = $00E8
SHFTIN  = $D5E9
SHFTR   = $E907
SHFTRES = $E8DA
SHIFT   = $E8F0
SHIN    = $D5F2
SHLOAD  = $F775
SI      = $F27B
SI1     = $F023
SI2     = $F026
SI3     = $F033
SIGN    = $002F
SIGN    = $EB82
SIGN1   = $EB86
SIGN2   = $EB88
SIGNFLG = $0016
SIGNIF  = $E82E
SIN     = $EFF1
SINSER  = $F075
SK2     = $D5FD
SKIP    = $DE64
SKIPTOK = $D5F9
SKP     = $DC57
SKPTK   = $D746
SKPV    = $DD02
SL1     = $F796
SL2     = $F7A0
SL3     = $F7A3
SLI     = $F3AF
SNDQ?   = $DC1F
SNGFLT  = $E301
SNTXERR = $DE0D
SPC?    = $DB21
SPEED   = $F262
SPEEDZ  = $00F1
SPKR    = $C030
SPNT    = $0049
SQR     = $EE8D
SQRhalf = $E92D
SQRtwo  = $E932
SS      = $EF85
SSF     = $D5E0
ST1     = $DE8A
STACK   = $0100
STATUS  = $0048
STE     = $ED60
STEP    = $D7AF
STEP    = $FA43
STEPZ   = $FEC4
STEX    = $ED05
STKINI  = $D683
STOADV  = $FBF0
STOP    = $D86E
STOR    = $FE0B
STORE   = $F39F
STR     = $E3C5
STRCMP  = $DF7D
STREND  = $006D
STRINI  = $E3D5
STRIP   = $D541
STRLIT  = $E3E7
STRLT2  = $E3ED
STRNG1  = $00AB
STRNG2  = $00AD
STRNG?  = $E01C
STROUT  = $DB3A
STRPRT  = $DB3D
STRSPA  = $E3DD
STRTRNG = $D6B1
STRTXT  = $DE81
STRtoLNG = $D310
STXP    = $DC27
STXTPT  = $D697
SUBERR  = $E196
SUBFLG  = $0014
SUBMAN  = $E806
SUBTBL  = $FFE3
SVAR    = $E4B5
SVARGO  = $E4BD
SVARS   = $E4A9
SVP     = $E0E8
SVY     = $EDBB
SWPNT   = $DC7E
SYNCHR  = $DEC0
SYNERR  = $DEC9
SYNTXERR = $D270
TAB     = $DB03
TABIT   = $DB2B
TABV    = $FB5B
TABWHERE = $DB16
TAN     = $F03A
TAN2    = $F062
TAPEIN  = $C060
TAPEOUT = $C020
TAPEPNT = $F7BC
TEMP1   = $0093
TEMP2   = $0098
TEMP3   = $008A
TEMPPT  = $0052
TEMPST  = $0055
TEND    = $EE8D
TEXT    = $F399
TMPEXP  = $0099
TOK?    = $D588
TOKDONE = $D75F
TOKEN?  = $D734
TOKLP   = $D749
TOKLUP  = $D755
TOKTABL = $D0D0
TOSUB   = $FFBE
TR1     = $D81D
TRACE   = $F26D
TRACE   = $FEC2
TRACE?  = $D805
TRCFLG  = $00F2
TRUE    = $DF60
TRUE?   = $D9D8
TVAR    = $E4A0
TWOBRK  = $E09A
TXPSV   = $0087
TXTCLR  = $C050
TXTPSV  = $00F4
TXTPTR  = $00B8
TXTSET  = $C051
TXTTAB  = $0067
TYPEMISS = $D303
UNARY   = $DF0C
UNDERR  = $D97C
UNDFNC  = $E30E
UNDFUNC = $D340
UNDSTAT = $D2BA
UNFNC   = $D080
UP      = $F4D5
UP      = $FC1A
UP1     = $F4EB
UP2     = $F4FB
UP3     = $F4FD
UP4     = $F4FF
UP5     = $F4F6
UPDOWN  = $F4D3
UPDWN1  = $F501
USR     = $000A
USR     = $FECA
USRADR  = $03F8
V2      = $002D
VAL     = $E707
VALTYP  = $0011
VAR?    = $DE6C
VARL    = $DED5
VARNAM  = $0081
VARNXT  = $DCFF
VARPNT  = $0083
VARTAB  = $0069
VARTIO  = $D8F0
VDONE   = $E55D
VFY     = $FE36
VFYOK   = $FE58
VIDOUT  = $FBFD
VL2     = $E70F
VL3     = $E727
VLIN    = $F241
VLINE   = $F828
VLINEZ  = $F826
VPNT    = $00A0
VR1     = $DEE5
VR2     = $DEF6
VSEARCH = $E04F
VTAB    = $F256
VTAB    = $FC22
VTABZ   = $FC24
WAIT    = $E784
WAIT    = $FCA8
WAIT2   = $FCA9
WAIT3   = $FCAA
WHATX   = $EE42
WNDBTM  = $0023
WNDLFT  = $0020
WNDTOP  = $0022
WNDUP   = $EE57
WNDWDTH = $0021
WNX     = $DC72
WR1     = $FED4
WRBIT   = $FCD6
WRBYT2  = $FEEF
WRBYTE  = $FEED
WRITE   = $FECD
WRTAPE  = $FCE5
WT2     = $E793
WT3     = $E797
XØH     = $00E1
XØL     = $00E0
X1      = $EF19
X2      = $EF27
X3      = $EF37
XAM     = $FDB3
XAM8    = $FDA3
XAMPM   = $FDC6
XBASIC  = $FEB0
XBRK    = $FA9C
XDRAW   = $F76F
XDRAWØ  = $F65D
XDRAW1  = $F661
XDRAW2  = $F682
XDRAW3  = $F68C
XDRAW4  = $F699
XDRAW5  = $F6A4
XJMP    = $FAC4
XJMPAT  = $FAC5
XJSR    = $FAB9
XQ1     = $FA78
XQ2     = $FA7A
XQINIT  = $FA4E
XQT     = $003C
XQTNZ   = $003C
XREG    = $0046
XRTI    = $FAA5
XRTS    = $FAA9
XTNDH   = $0053
XTNDL   = $0052
YØ      = $00E2
YREG    = $0047
YSAV    = $0034
YSAV1   = $0035
ZARY    = $E21A
ZEND    = $E408
ZERDLY  = $FCDB
ZERO    = $EA31
ZEROFAC = $E84E
ZF      = $DBE9
ZFAC    = $EC40
ZLUP    = $E229
ZMODE   = $FFC7
ZPSTUFF = $F10B
```

peeker Börse

Verkauf Software

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

Apple II-Programmgenerator für Eingabemasken 59,–. DATEIVERWALTUNG m. Editor, Grafik, Listenpaket 65,–. Info –,80 bei: FAHRENBERG, Rittershausstr. 4, 5300 Bonn 1

Multi Tools, 20 leistungsstarke Assemblerutilities für II +, e,c 60,–, **programmierbare Laufschrift** für Werbung u.ä. 30,–. C. F. Mahr, Waldackerweg 71, 7300 Esslingen

SIDEWAYS für Apple und IBM druckt Kalkulationstabellen und Texte um 90 Grad gedreht. **SIDEWAYS** kostet ab Lager nur DM 250 plus MwSt. bei LUCIUS Computer-Programme, Th.-Körner-Str. 5C, 4220 Dinslaken, 0 21 34/5 27 82

160-Track-Besitzer aufgepaßt: 16-Sekt-Copyprg / einfache Bedien / in Masch / jede Trackzahl möglich! Rüter, Rahdener Str. 65, 4955 Hille

Ankauf Software

Suche für **Apple-II** Fortran-77 und Fortran-80 mit Anleitungen Tel.: 0 52 51/9 16 24

Verkauf Hardware

TEAC 55F 600,– 55B 550,– * SSDD 10 St 42,– 96 tpi DS 78,– Lochverst. europ. Markenw. * Box Rglas. Schl. für 40–42,– für 80–49,– * Monitore 18Mhz 12″ entsp. gr. ab 298,–. **VICO**, Selchower Str. 31, 1000 Bln 44

Verk. Apple IIe, 128K, 80 Z, 2 Disk., Monitor, Drucker 8510 A, A-Writer, Visic., Datenbank usw., NP 11500, VP 5500, Tel. 02 09/27 10 70 ab 18 h.

Verkaufe APPLE IIc + Monitor + Imagewriter – Tel. 0 94 09/933

Apple IIc mit 5 orig. Disk + 2 LW 3 Mon. alt zvk. DM 3300,– VHS Tel. 0 72 53/37 14.

APPLE-Tastaturen werden durch BETA-Box zu frei programmierbar. Tastatur Info: Altenburg, Kampstr. 20, 5014 Kerpen-3, Preis ab 215,–

APPLE + Zubehör und Drucker Tel. 0421/358488

X-Y Plotter DIN A3
6 Farb X-Y Plotter mit Centronic Schnittstelle günstig zu verkaufen Tel.: 0 63 03/43 28.

Biete **Epson** MX80 + Interf. 780,– Tel.: 0 70 71/6 31 36

Compatibler im IBM-Look mit 6502 + 280 CPU, 256K Ram Disk, 2 160K TEAK-Laufwerke, abgesetzte Tastatur, VOICE-Generator, Musik- und Printerinterface zu verkaufen – Tel.: 0 63 03/43 28

MACINTOSH 128KB neuwertig 5999 DM IMAGEWRITER original verp. 1499 DM zu verkaufen. Tel. 0 61 51/2 71 47

BASIS 108 Komplettsyst. CP/M 3. 2 F122 Laufw. Preis DM 4100,–. PAL Karten DM 120,– Tel.: 0 27 47/29 99

Disketten 5 1/4″, 1 D, zu 3,70 DM, EDV-Zubehör. G. Einbeck, Noriker Str. 19, Block 2, Nürnberg, Tel. 09 11/46 06 81, Mo–Fr 7 bis 12 Uhr.

Verschiedenes

APPLE USERS
Verkaufe deutsche Bedienungsanleitung für den Imagewriter-Drucker. Tel.: 02 31/33 60 23 nach 18 Uhr

** **APPLE-Bücher** natürlich bei ** DV-Fachbuchhandlung M. Igelhaut, Hardenbergstr. 34, 8510 Fürth, Tel. 09 11/72 14 43, GRATIS-INFO: AP

Einkaufsführer

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

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

Erscheinungs- und Anzeigenschlußtermine für peeker

Ausgabe	Erscheinungstermin	Anzeigenschluß
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 5/85

Sie haben einen Apple...

wir haben die Software...

und die Hardware...

wir haben die Bücher...

und die Zeitschriften*...

*Fordern Sie unseren Gratiskatalog an!

ALLES FÜR DEN APPLE II, II+, IIE

pandasoft Dr.-Ing. Eden

UHLANDSTR. 195 · D-1000 BERLIN 12

TEL.:(030) 310 423 · TELEX: 18 58 59

Autorisierter apple Fachhändler MICROSOFT Distributor

Ich besitze einen Apple. Bitte schicken Sie mir Ihren kostenlosen Katalog.

Name: _____

Adresse: _____

Kenner kaufen, was Sie kennen:

MULTILIFE

Die klassische Markendiskette. Die Alternative zu den Namenlosen!

Auch bei den Disketten machen sich "NoNames" breit. Was verbirgt sich in den weißen Kartons? Sie können es nicht wissen. Selbst wenn Sie einmal mit der Qualität zufrieden waren, was steckt beim nächsten Kauf in der namenlosen Packung?

MULTILIFE Wendedisketten
Mit 2 Schreibschutzkerben und 2 Indexlöchern. Geprüft für zweiseitige Benutzung durch Wenden bei einseitigen Laufwerken (z.B. APPLE, COMMODORE 1541).

10 Stück im roten Karton

49.-

Vertrauen Sie deshalb einer Markendiskette! Denn an der Marke erkennen Sie die gleichbleibende Qualität: **MULTILIFE**, die klassische Markendiskette. Jede **MULTILIFE**-Diskette wird während der Produktion über 100 Mal geprüft!
Deshalb ist jede MULTILIFE 100% Error-free!

Alle **MULTILIFE**-Disketten mit Mittellochverstärkung. Für noch längere Lebensdauer auch bei häufigem Diskettenwechsel!

MULTILIFE-Disketten gibt es in 5 verschiedenen Ausführungen:

Art.-Nr.	Menge		Preis
49010	10 Stück	im braunen Karton 1S/1D	29.-
49012	10 Stück	im schwarzen Karton 2S/1D	39.-
49013	10 Stück	im roten Karton 2S/1D Wendediskette beidseitig benutzbar	49.-
49014	10 Stück	in Nachfüllpackung 2S/2D	59.-
49016	10 Stück	in Plastikarchivbox 2S/2D	69.-

MULTILIFE-DISKETTEN-ZUBEHÖR

40er-Diskettenbox	35.-	60er-Diskettenbox	45.-
80er Diskettenbox	55.-	Reinigungsset für Laufwerke	39.-

WICHTIG! Herstellerbedingte Lieferzeiten. Aufgrund erhöhter Nachfrage sind nicht immer alle Teile sofort lieferbar!

ACHTUNG: Großabnehmer! Sonderkonditionen!
Bei 50 Stück Jahresabnahme 5% Nachlaß
Bei 100 Stück Jahresabnahme 10% Nachlaß
Gemischte Abrufe von je 10 Stück über 1 Jahr möglich. Einfach nebenstehenden Coupon in der Filiale abgeben oder zur Versandzentrale nach Aachen schicken!

Hiermit bestelle ich:

☐ 50 Stück MULTILIFE-Disketten (5% Nachlaß)
☐ 100 Stück MULTILIFE-Disketten (10% Nachlaß)
zum Abruf in einem Jahr beginnend mit dem heutigen Datum (Mindestabruf 10 Stück).
Als ersten Abruf bitte ich um Lieferung von:

____ x 10 Stück _____ (Art.-Nr.) à _____ = _____ DM
____ x 10 Stück _____ (Art.-Nr.) à _____ = _____ DM
____ x 10 Stück _____ (Art.-Nr.) à _____ = _____ DM
Zwischensumme _____ DM
abzüglich ____ % Nachlaß – _____ DM
Endsumme _____ DM

Meine Anschrift:

Name/Vorname

Straße (PLZ) Ort

Datum/Unterschrift

PEEK

kompetent + preiswert

VOBIS
Deutschlands umsatzgrößter Microcomputer-Spezialist

VERSANDZENTRALE: Viktoriastraße 74
5100 AACHEN · Tel. 0241/50081 · Tx 832389 vobisd

FILIALEN:
HAMBURG Krohnskamp 15 · 040/2 79 46 76
HANNOVER Berliner Allee 47 · 0511/81 65 71
DÜSSELDORF Heideweg 107 · 0211/63 33 88
DORTMUND Hamburger Str. 110 · 0231/57 30 72
KÖLN Mathiasstr. 24-26 · 0221/24 86 42
AACHEN Viktoriastr. 74 · 0241/54 31 00
AACHEN Pontstraße 60
FRANKFURT Frankenallee 207/209 · 069/73 40 49
STUTTGART Marienstr. 11-13 · 0711/60 63 36
MÜNCHEN Aberlestr. 3 · 089/77 21 10