Das deutsche Apple-Magazin ist da!

Einzelpreis DM 6,50

peeker

MAGAZIN FÜR APPLE-COMPUTER

1
SEPTEMBER 1984, 1. Jahrgang

65CO2-DISASSEMBLER
für Apple IIe und II Plus

ACCELERATOR IIe
Die neue Superkarte

DOUBLE LORES
Wie man die Grafik verdoppelt

TURBO PASCAL
Schritt für Schritt

PRODOS
80-Track-Formatierung

PREISAUSSCHREIBEN
Primzahlen Wettbewerb

Hüthig
PUBLIKATION

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: .....

c

EDITORIAL

Mit diesem Heft liegt Ihnen die erste Ausgabe von „Peeker" vor, die übrigens bereits die 45. Zeitschrift der Verlagsgruppe Dr. Alfred Hüthig ist. „Peeker" ist – wenn man von dem nicht allgemein erhältlichen Vereinsblatt des deutschen Apple-Clubs in Oberhausen absieht – das erste deutschsprachige Magazin, das sich ausschließlich mit Apple-Computern befaßt. PEEK ist ein Befehl aus der Programmiersprache BASIC, der die Untersuchung einer Speicherstelle gestattet. „Peeken" muß man immer dann, wenn *man bei einem Computer etwas genauer wissen will*. Und genau dies ist die Zielsetzung unserer neuen Zeitschrift „Peeker". Schön verpackte, aber oberflächliche Information hilft keinem Apple-Besitzer weiter, gleichviel ob er Selbstprogrammierer oder reiner Anwender ist. Nehmen wir als Beispiel den Image-Writer als einen der für Apple-Computer erhältlichen Matrix-Drucker. Die Handhabung der nicht genormten Steuerzeichen von Druckern war stets und ist immer noch ein leidiges Problem für alle Apple-Benutzer. Man könnte sich nun in einem Aufsatz über den Image-Writer auf oberflächliches Blabla in der Art „Der Image-Writer hat ein gefälliges beige Design und in seinem Herzen klopft ein kräftiger Prozessor" beschränken. Besitzt man jedoch erst einmal einen Image-Writer, dann stellt man plötzlich ganz andere Dinge als ein pochendes Herz oder ein formschönes Styling fest. So kommt man beispielsweise rasch zu der ernüchternden Erkenntnis, daß die mitgelieferte serielle Schnittstelle keinerlei druckerspezifische ROM-Software enthält und demzufolge viele nützliche Befehle fehlen. Jetzt beginnt die Phase des „Peekens", d.h. die Suche nach gezielten Detailinformationen, damit der Drucker das macht, was Sie von ihm erwarten. Und genau hier setzt unser „Peeker" ein. Jedes Heft enthält eine Fülle nützlicher Utilities und Programme, Tips und Tricks, Test- und Erfahrungsberichte sowie Aufsatzserien für Anfänger und Fortgeschrittene. Damit Sie die teilweise sehr umfangreichen, professionellen Programme nicht abtippen müssen, erscheint je nach Bedarf etwa alle 2-3 Monate eine Sammeldiskette, die die größeren Programme aus den vorangehenden Heften von „Peeker" zusammenfaßt.

Hüthig
PUBLIKATION

INHALT

9/84

Impressum

Peeker
Magazin für Apple-Computer
1. Jahrgang 1984
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:
Bernd Beutel, Tel. (0 62 21) 48 92 18
z. Zt. gilt Anzeigenpreisliste Nr. 1
Vertriebsleitung:
Ruth Biller, Tel. (0 62 21) 48 92 80
Produktionsleitung: Gunter Sokollek
Gestaltung: Rainer Schmitt

PEEKER

4

65C02-Feature

In diesem dreiteiligen Feature ist alles zusammengetragen, was Sie schon immer über den neuen 65C02-Prozessor wissen wollten. Im ersten Beitrag „Der neue 65C02-Prozessor" werden die neuen Befehle detailliert geschildert und an Hand von Beispielprogrammen erläutert. Im zweiten Aufsatz wird ein „65C02-Disassembler für Apple IIe und II Plus" vorgestellt, der auf dem IIc-Disassembler basiert und sogar auf Geräten lauffähig ist, die nur über den alten 6502-Prozessor verfügen. Schließlich werden im dritten Beitrag „Accelerator IIe – Die neue Superkarte" die Geheimnisse der brandneuen CMOS-Karte mit dem 4MHz-65C02C enthüllt. Insbesondere wird gezeigt, wie man sich das Pseudo-ROM für modifizierte Monitor- und Applesoft-ROMs zunutze machen kann.

30

Double Lores

Double Hires auf dem Apple-IIe mit 64K-Karte ist seit einiger Zeit bekannt. Daß aber auch doppelte Lores-Grafik möglich ist, wird hier erstmals geschildert. Damit eröffnen sich neue Perspektiven für Spielprogramme, für die Hires zu fein und normale Lores zu grob in der Grafikauflösung ist.

38

Hires-Grafik-Dump

Diese professionelle Utility gestattet den Hires-Bildschirmausdruck mit allen nur denkbaren Features wie horizontaler und vertikaler Verzerrung, Bildausschnittvergrößerung usw.

51

Turbo-Pascal

Für alle Pascal-Liebhaber, die auf das schnellere und komfortablere Turbo-Pascal umsteigen wollen, wird hier eine minutiöse Schritt-für-Schritt-Anleitung zur Systemkonfigurierung und zur Erstellung Ihres ersten „Turbo"-Programms gebracht.

54

ProDOS-Patches

Mehrere Patch-Programme zeigen, wie man erstens ProDOS auch bei geändertem Monitor-ROM sowie zweitens bei 80-Spur-Laufwerken zum Laufen bringt.

58

Primzahlen-Preisausschreiben

Wollen Sie 500,– DM gewinnen? Dann beteiligen Sie sich an unserem Primzahlen-Geschwindigkeitswettbewerb!

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

Erscheinungsweise:
1984 2 Hefte, ab 1985 12 Hefte jährlich.
Jahresabonnement DM 58,–, einschließlich MwSt, im Inland portofrei.
Einzelheft DM 6,50

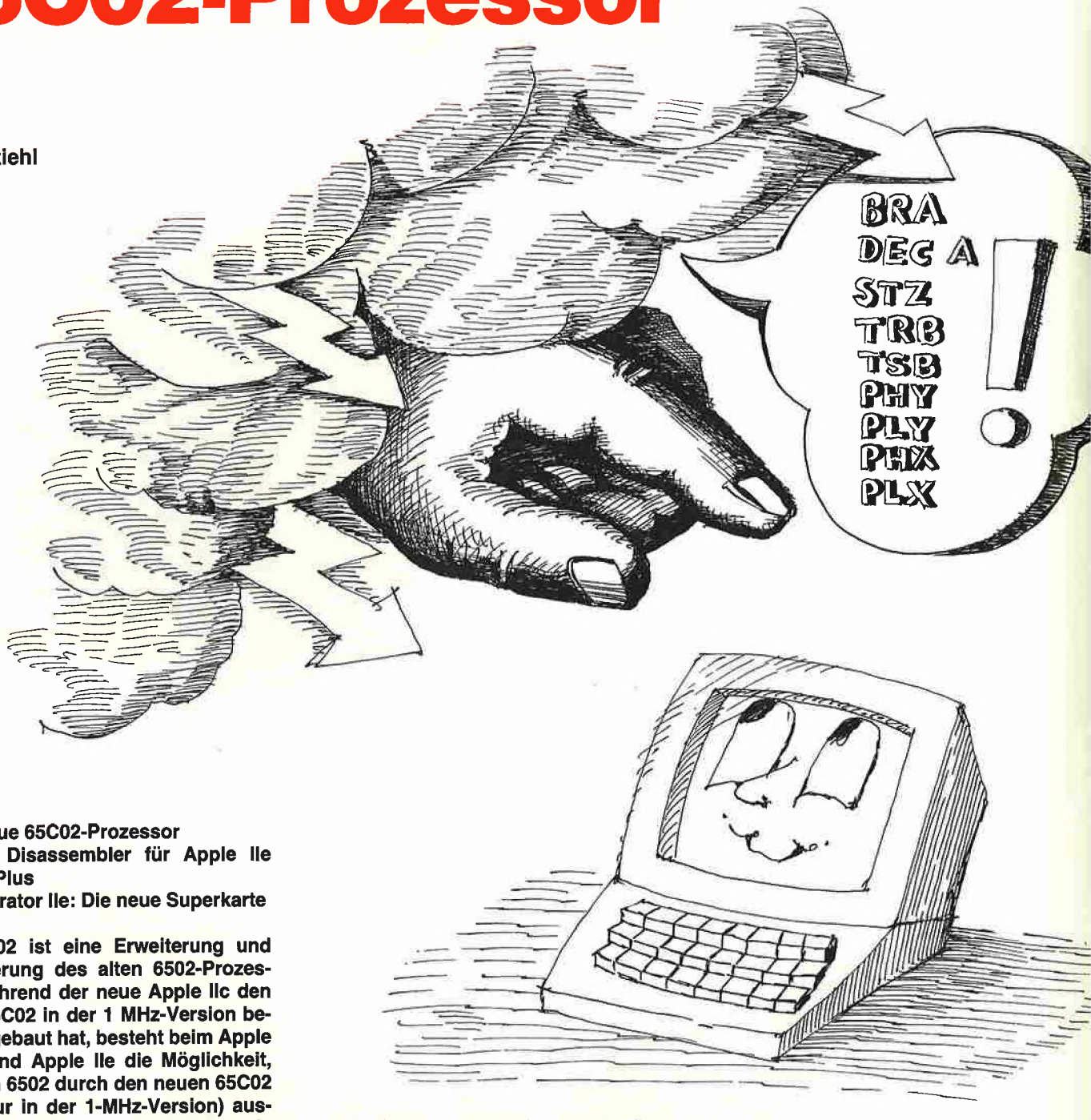
Zahlungen: an den Dr. Alfred Hüthig-Verlag GmbH, D-6900 Heidelberg 1: **Postscheckkonten:** BRD: Karlsruhe 48545-753; Österreich: Wien 7555888; Schweiz: Basel 40-24417; Niederlande: Den Haag 145728; Italien: Mailand 47718; Belgien: Brüssel 723026; Dänemark: Kopenhagen 34969; Norwegen: Oslo 99424; Schweden: Stockholm 5477765

Bankkonten: Landeszentralbank Heidelberg 67207341; BLZ 67200000; Deutsche Bank Heidelberg 0 2165 041; BLZ 67270003; Bezirkssparkasse Heidelberg 20451, BLZ 67250020.

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Der neue 65C02-Prozessor

Ulrich Stiehl



- **Der neue 65C02-Prozessor**
- **65C02 Disassembler für Apple IIe und II Plus**
- **Accelerator IIe: Die neue Superkarte**

Der 65C02 ist eine Erweiterung und Verbesserung des alten 6502-Prozessors. Während der neue Apple IIc den neuen 65C02 in der 1 MHz-Version bereits eingebaut hat, besteht beim Apple II Plus und Apple IIe die Möglichkeit, den alten 6502 durch den neuen 65C02 (indes nur in der 1-MHz-Version) auszuwechseln. Darüber hinaus kann die brandneue 65C02C-Karte namens „Accelerator IIe", die mit 4 Megahertz getaktet ist, in den Apple IIe oder auch den Apple II Plus eingesetzt werden, womit alle Programme bis zu 350% schneller laufen.

Die nachfolgenden technischen Details sowie das Diagramm basieren auf Unterlagen der Rockwell International GmbH, Fraunhoferstraße 11, 8033 München-Martinsried, sowie auf eigenen Untersuchungen anhand der Accelerator-Karte. Neben dem sog. R65C02 liefert Rockwell noch die Derivate R65C102 und R65C112. Den R65C02 gibt es (a) in Keramik- und Plastik-Versionen, (b) in 1, 2, 3 und 4 MHz-Versionen sowie (c) in kommerziellen und industriellen Temperatur-Versionen (im ersteren Fall 0 Grad bis 70 Grad Celsius).

Alle 65C02-Prozessoren werden als CMOS-CPUs geliefert, während die alten 6502-Prozessoren NMOS-CPUs waren (CMOS = Complementary Metal Oxide Semiconductor). Von Vorteil ist der erheblich geringere Stromverbrauch gegenüber der NMOS-Technologie. Von Nachteil ist dagegen die relativ hohe Empfindlichkeit gegenüber statischer Elektrizität. (Beispielsweise sollte man sich bei der Accelerator-Karte erst einmal „entladen", indem man etwa auf das Netzteil-Metallgehäuse des Apples greift, bevor man die Karte anfaßt.)

Verbesserungen

Neben den eigentlichen neuen Befehlen sind folgende technische Verbesserungen implementiert worden:

a) Ungültige Operationscodes

Ungültige Op-Codes, z.B. $02, werden jetzt allesamt als NOPs bzw. $EA behandelt. Beispiel für Apple II Plus/IIe:
CALL -151
0300: 02 4C 59 FF
0300G
– Beim Apple II Plus bzw. IIe mit normalem 6502-Prozessor würde der Apple nach 300G hängen".
– Beim Apple IIc oder beim Apple IIe mit Accelerator IIe würde $02 ignoriert und dann der nächste Befehl ab $0301 ausgeführt.
Für das Debuggen von Maschinenprogrammen ist die Behandlung von Nicht-Op-Codes als NOPs indessen eher nachteilig, da man hierbei u.U. nie feststellt, daß das Programm „Schrott-Stellen" enthält.

b) Bug beim indirekten JMP

Der Fehler beim indirekten Sprung, z.B. JMP ($10FF), wurde beseitigt. Dieser Fehler trat nur dann auf, wenn sich das Low Byte der indirekten Adresse in einer beliebigen Speicherstelle $??FF befand, also das Low Byte am Ende der letzten Page und das High Byte am Anfang der nächsten Page stand. Beispiel:

```
CALL -151
0300: 4C 59 FF   JMP $FF59 (Reset-Adresse)
0303: 6C FF 10   JMP ($10FF)
10FF: 00         Low Byte von $0300
1100: 03         High Byte von $0300
0303G
```

Nach 303G erfolgt beim neuen 65C02 ein korrekter indirekter Sprung zur Adresse $0300, die sich der Prozessor aus $10FF (Low Byte) und $1100 (High Byte) holt. Demgegenüber würde sich der alte 6502 irrtümlicherweise das Low Byte von Speicherstelle $10FF und das High Byte von Speicherstelle $1000 holen. Um dies zu beweisen, geben wir folgendes ein:
1000: 03 (High Byte!)
10FF: 00 (Low Byte!)
1100: 00 (High Byte wird ignoriert!)
0303G
Beim alten 6502 springt der Prozessor nunmehr nicht nach $0000, sondern nach $0300.

c) Dezimalmodus

Nach Hardware-Reset, d.h. nach Drücken der Ctrl-Reset-Tasten, schaltet der 65C02 nunmehr automatisch auf Binärmodus bzw. Nicht-Dezimalmodus (CLD) um. Da dies beim alten 6502 nicht der Fall war, mußte CLD durch die Software-Reset-Routine bei $FF59: CLD bzw. bei $FA62: CLD nachgeholt werden.

Bei ADC und SBC im Dezimalmodus sind die Flags N (Negative), V (Overflow) und Z (Zero) nunmehr gültig, wobei diese Befehle jedoch dann 1 Takt mehr benötigen. Beim alten 6502 waren bei Dezimaloperationen die N-, V- und Z-Flags unbestimmt bzw. ungültig.

d) Read-Modify-Write

Die direkten Speicheränderungsbefehle (Read-Modify-Write-Befehle) DEC, INC, ASL, LSR usw. werden beim 6502 mit 1 Lese- und 2 Schreibzyklen und bei 65C02 mit 2 Lese- und 1 Schreibzyklus ausgeführt. Inwieweit dadurch die Softswitches des Apple II, z.B. $C030 für Lautsprecher, andersartig beeinflußt werden, konnte bislang nicht festgestellt werden. Bei den Softswitches ist es bekanntlich nicht irrelevant, ob man auf diese mit LDA oder STA zugreift. Sinngemäßes gilt für DEC, INC usw.

Erweiterungen

Neue Befehle

BRA = Op-Code $80 = Branch relative always

Dieser Befehl bewirkt eine unbedingte, relative Verzweigung, während die alten bedingten Verzweigungsbefehle BCC, BCS usw. von dem jeweiligen Flag abhängen. Der BRA-Befehl ist insbesondere für relokative Programme sehr nützlich. Beim alten 6502 war man gezwungen, Scheinflags zu setzten, um eine unbedingte, relative Verzweigung zu erzwingen, z.B. CLV BVC LABEL.

DEC A = DEA = Op-Code $3A = Decrement A

INC A = INA = Op-Code $1A = Increment A

DEC A vermindert den Akkumulator um 1: A = A - 1 und entspricht insoweit den DEY- und DEX-Befehlen. Die Assembler-Syntax ist DEC A, DEC oder DEA. INC A erhöht den Akkumulator um 1: A = A + 1 und entspricht den INY- und INX-Befehlen. Die Assembler-Syntax ist entsprechend INC A, INC oder INA.
DEC A ersetzt die Folge SEC SBC #1 und INC A ersetzt CLC ADC #1, wodurch einige Prozessor-Takte eingespart werden können.

PHX = Op-Code $DA = Push X on Stack
PHY = Op-Code $5A = Push Y on Stack
PLX = Op-Code $FA = Pull X from Stack
PLY = Op-Code $7A = Pull Y from Stack

Bei alten 6502 konnte man nur den Akkumulator direkt auf den Stack retten. Wollte man auch noch die X- und Y-Register retten, so mußte man etwa wie folgt kodieren:

```
PHA  Vorher:   A, X und Y auf Stack
TXA            schieben
PHA
TYA
PHA
PLA  Nachher:  Y, X und A vom Stack
TAY            holen
PLA
TAX
PLA
```

Beim 65C02 geht es nunmehr viel eleganter:

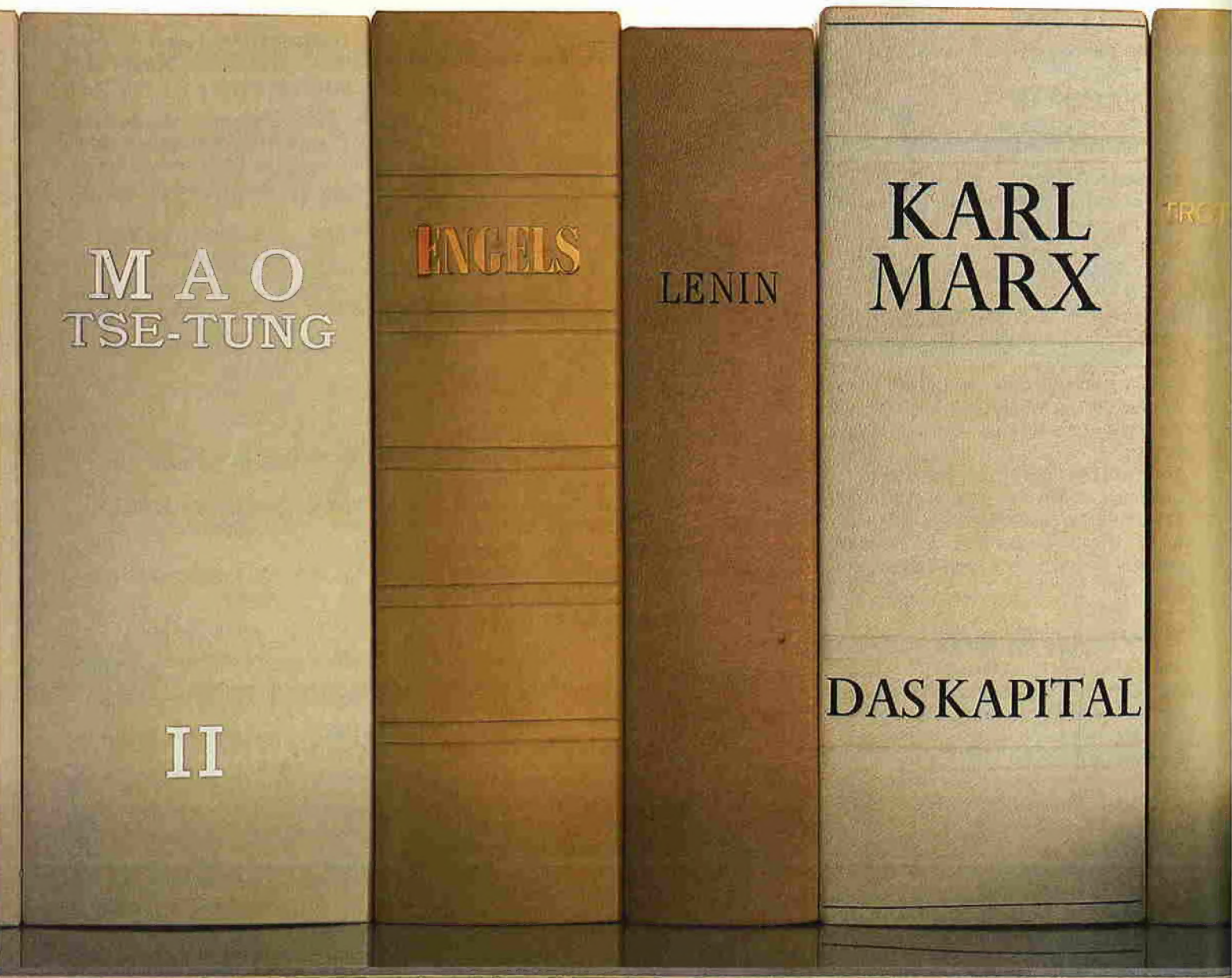
```
PHA  Vorher:   A, X und Y auf Stack
PHX            schieben
PHY
PLY  Nachher:  Y, X und A vom Stack
PLX            holen
PLA
```

Dadurch wird das Programm nicht nur kürzer, sondern auch schneller, da jeweils 2 Befehle eingespart werden.

STZ $1000 = STZ Absolute Adresse = Op-Code $9C
STZ $1000,X = STZ Absolute Adresse, X = Op-Code $9E
STZ $00 = STZ Zeropage = Op-Code $64
STZ $00,X = STZ Zeropage, X = Op-Code $74

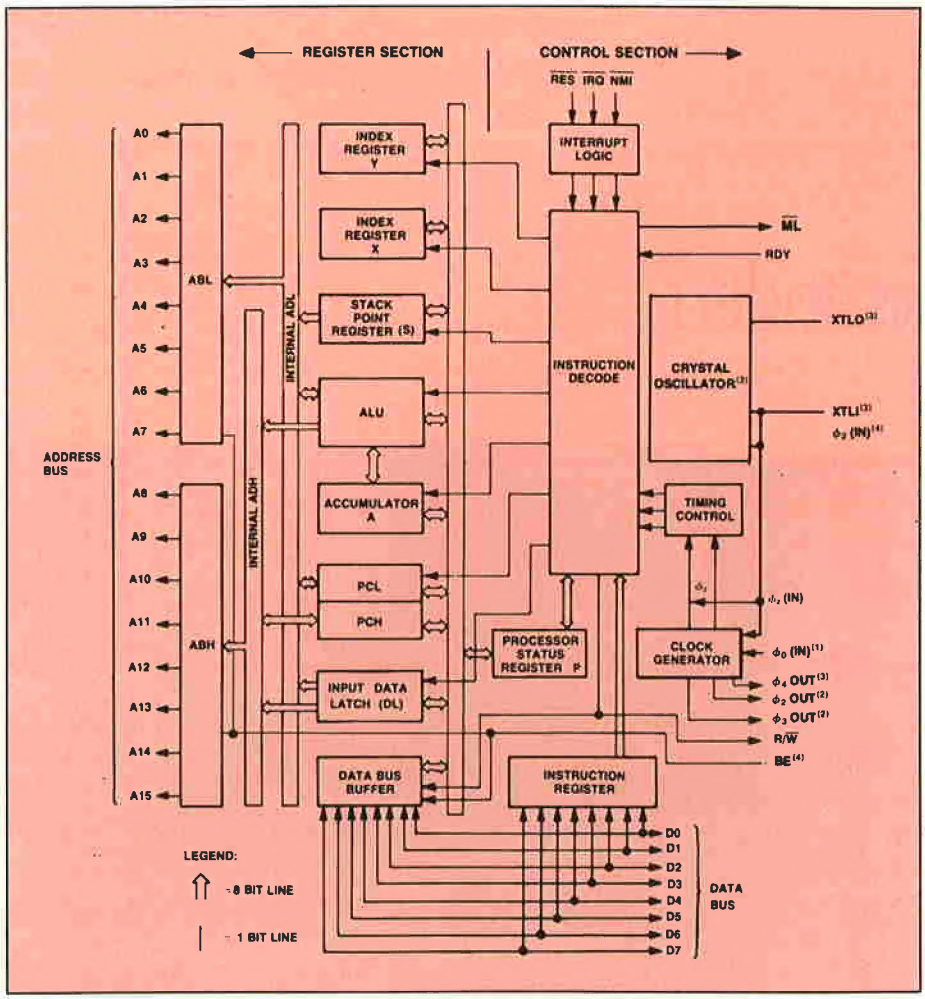
Diese Befehle setzen eine Speicherstelle auf Null bzw. Zero $00. Das „Z" in STZ steht nicht für Zeropage-Adressierung, sondern für Zero = Null = Byte $00. Die Register A, X und Y bleiben unverändert.

Es wurde Zeit, daß mal ein



Kapitalist die Welt verändert.

Apple hat den Macintosh erfunden.



R 65C02 Interne Architektur

Beim 6502 muß man demgegenüber ein Register „opfern", um eine Speicherstelle zu löschen, z.B. LDA #$00 STA ADRESSE.

TRB $FFFF = Test and reset memory bits with accumulator (Absolute Adresse) = Op-Code $1C
TRB $00 = Test and reset memory bits with accumulator (Zeropage-Adresse) = Op-Code $14

Dieser Befehl ist neu und entspricht einem AND-Befehl mit umgekehrten Vorzeichen, d.h. Undierung mit Einerkomplement des Bitmusters. Beim alten 6502 würde man etwa wie folgt simulieren:
LDA BITMUSTER
AND MEMORY
EOR #%11111111
STA MEMORY
TRB besagt, daß eine Speicherstelle M durch den Inhalt des Akkumulators A verändert wird, wobei A selbst unverändert bleibt, *und zwar blendet TRB bei M diejenigen Bits weg, die bei A gesetzt sind.*

Beispiele:

M 1111 1111 LDA #$00 TRB M, wobei M
A 0000 0000 vorher $FF
M 1111 1111 M nachher $FF

M 1111 1111 LDA #$FF TRB M, wobei M
A 1111 1111 vorher $FF
M 0000 0000 M nachher $00

M 0000 0000 LDA #$FF TRB M, wobei M
A 1111 1111 vorher $00
M 0000 0000 M nachher $00

M 0000 0000 LDA #$00 TRB M, wobei M
A 0000 0000 vorher $00
M 0000 0000 M nachher $00

M 1111 1111 LDA #$7E TRB M, wobei M
A 0111 1110 vorher $FF
M 1000 0001 M nachher $81

M 1010 1010 LDA #$F0 TRB M, wobei M
A 1111 0000 vorher $AA
M 0000 1010 M nachher $0A

TSB $FFFF = Test and set memory bits with accumulator (Absolute Adresse) = Op-Code $0C
TSB $00 = Test and set memory bits with accumulator (Zeropage-Adresse) = Op-Code $04

TSB entspricht dem ORA-Befehl, der jedoch direkt auf eine Speicherstelle angewandt wird. *TSB blendet bei M diejenigen Bits hinzu, die bei A zusätzlich gesetzt sind.* Beim alten 6502 würde man dies simulieren mit

LDA BITMUSTER
ORA MEMORY
STA MEMORY

Einige Beispiele:

M 1111 1111 LDA #$00 TSB M, wobei M
A 0000 0000 vorher $FF
M 1111 1111 M nachher $FF

M 1111 1111 LDA #$FF TSB M, wobei M
A 1111 1111 vorher $FF
M 1111 1111 M nachher $FF

M 0000 0000 LDA #$FF TSB M, wobei M
A 1111 1111 vorher $00
M 1111 1111 M nachher $FF

M 0000 0000 LDA #$00 TSB M, wobei M
A 0000 0000 vorher $00
M 0000 0000 M nachher $00

M 1111 1111 LDA #$7E TSB M, wobei M
A 0111 1110 vorher $FF
M 1111 1111 M nachher $FF

M 1010 1010 LDA #$F0 TSB M, wobei M
A 1111 0000 vorher $AA
M 1111 1010 M nachher $FA

Während die Befehle AND und ORA nur den Inhalt des Akkumulators beeinflussen, können mit TRB und TSB die Inhalte von Speicherstellen direkt verändert werden, wobei der Akkumulator unverändert bleibt. Die Befehlsfolge lautet stets:

LDA BITMUSTER
TSB MEMORY
oder
LDA BITMUSTER
TRB MEMORY

Nehmen wir an, bei einem Textverarbeitungs-Text, der sich im Speicher $2000-$3FFF befände, sollte bei jedem Buchstaben Bit 7 gelöscht werden. Zu diesem

Zweck müßte man beim alten 6502 den gesamten Bereich mit der Befehlsfolge

```
LOOP LDA M
     AND #%01111111
     STA M
```

...jetzt M inkrementieren und erneut zu Loop...
„bearbeiten". Hingegen würde man beim neuen 65C02 mit der Schleife

```
     LDA # %10000000
LOOP TRB M
```

... jetzt M inkrementieren und erneut zu Loop...
zu einer erheblich kürzeren Ausführungszeit kommen.

BBR, BBS, RMB, SMB

BBR (Branch on bit reset), BBS (Branch on bit set), RMB (Reset memory bit) und SMB (Set memory bit) sind laut Auskunft der Firma Rockwell nur beim R65C02 als Zeropage-Bitmanipulationsbefehle implementiert. Meine Accelerator-Karte ist mit einem NCR-65C02C bestückt, der diese Befehle nicht kennt. Dasselbe gilt für den im Apple IIc befindlichen 65C02. Auch der Apple-IIc-Disassembler verwendet die neuen Op-Codes nicht. Es wird empfohlen, auf die Benutzung dieser OP-Codes zu verzichten, um Kompatibilität mit allen 65C02-Varianten zu gewährleisten.

Neue Adressierungsarten

Indirekte nicht-indizierte Adressierung

Anstelle der häufig benutzten Adressierungsform
LDA (INDIREKT), Y
usw., die bislang nur mit Hilfe des Y-Registers implementiert werden konnte, besteht nunmehr die zusätzliche Möglichkeit, auf Y völlig zu verzichten. Im einzelnen gilt dies für folgende Befehle:

```
LDA (INDIREKT) = Op-Code $B2; neben
                 LDA (INDIREKT), Y
STA (INDIREKT) = Op-Code $92; neben
                 STA (INDIREKT), Y
ORA (INDIREKT) = Op-Code $12; neben
                 ORA (INDIREKT), Y
EOR (INDIREKT) = Op-Code $52; neben
                 EOR (INDIREKT), Y
AND (INDIREKT) = Op-Code $32; neben
                 AND (INDIREKT), Y
ADC (INDIREKT) = Op-Code $72; neben
                 ADC (INDIREKT), Y
SBC (INDIREKT) = Op-Code $F2; neben
                 SBC (INDIREKT), Y
CMP (INDIREKT) = Op-Code $D2; neben
                 CMP (INDIREKT), Y
```

Dies ist eine nützliche Erweiterung, da man nunmehr bei Array-Manipulationen nicht mehr permanent das Y-Register retten bzw. für die indizierte Adressierungsform freihalten muß.

BIT-Adressierung

Der BIT-Befehl wurde um drei neue Adressierungsarten erweitert. Insgesamt gibt es jetzt folgende BIT-Instruktionen:

BIT $00 Zeropage-Adressierung (alt) - Op-Code $2C
BIT $1000 Absolute Adressierung (alt) - Op-Code $24
BIT $00,X Indizierte Zeropage-Adressierung (neu) - Op-Code $34
BIT $1000,X Indizierte absolute Adressierung (neu) - Op-Code $3C
BIT #$00 Unmittelbare Adressierung (neu) - Op-Code $89

Die unmittelbare Adressierungsform scheint mir nutzlos zu sein, da hiermit *nicht* Bit 7 (N-Flag) und Bit 6 (V-Flag) getestet werden können, was für alle anderen BIT-Adressierungsarten zutrifft.

Indizierter, indirekter JMP

JMP ($1000,X) - Op-Code $7C - ist eine wertvolle Erweiterung des indirekten Sprungs für den Fall, daß eine Sprungadresse einer Tabelle entnommen werden soll. Nehmen wir an, ab Speicherstelle $1000 steht folgende fingierte Sprungtabelle:
1000: 00 20 $2000 0. Sprung
1002: 00 30 $3000 1. Sprung
1004: 00 40 $4000 2. Sprung usw.
Nehmen wir fernerhin an, der zweite Sprung sei auszuführen. Dann verfahren wir wie folgt:

```
LDA #2          ;2. Sprung
ASL             ;2 * 2 = 4 = $1004
TAX
JMP ($1000,X) ;entspricht JMP $3000
```

D.h. $1000 + X = $1004 enthält das Low Byte und $1004 + 1 = $1005 enthält das High Byte der effektiven Sprungadresse.

Das nachfolgende **„65C02-Demo einiger neuer Op-Codes"** veranschaulicht die Anwendung der neuen Befehle und Adressierungsarten. Dieses Demo zeigt übrigens folgende Hexzahlen und ASCII-Zeichen an:
030201
01
BCA
0A AF

Zur Assemblierung wurde der zu den von Firma Apple herausgegebenen „ProDOS Assembler Tools" gehörige 6502-Assembler verwendet, der ebenso wie der „Bugbyter Debugger" die neuen Befehle verarbeitet. Der Assembler-Quellcode muß zu diesem Zweck mit dem Pseudo-Op-Code „X6502" eingeleitet werden.

Zusätzlich zu dem Source-Code des 65C02-Demos wird das disassemblierte Listing in der Form abgedruckt, wie es mit dem „65C02-Disassembler", der im nachfolgenden Aufsatz geschildert wird, erzeugt werden kann.

Schlußbemerkung

a) Da der Apple II Plus und der Apple IIe im Gegensatz zum Apple IIc „von Hause aus" den 65C02 nicht enthalten, sollte man in kommerziellen Programmen, die für *alle* Gerätetypen gedacht sind, auf die neuen Op-Codes verzichten.

b) Bei eigenen Programmen lohnt es sich u.U., den alten Apple II Plus bzw. Apple IIe mit dem neuen 65C02 zu bestücken und die neuen Op-Codes zu verwenden, da Programme dann etwas kompakter und zugleich etwas schneller werden. Der Vorteil ist jedoch nur in Spezialfällen überzeugend. Grundsätzlich kann man auch ohne die neuen Op-Codes auskommen.

c) Wichtiger scheint mit der Umstand, daß der 65C02 weniger Wärme entwickelt. Wer sich z.B. noch an die „brütende Hitze" der alten Videx-Karte im Apple II Plus zurückerinnern kann, der wird von der „wohligen Kühle" der Accelerator-IIe-Karte im Apple IIe angenehm überrascht sein. Ventilatoren werden dann überflüssig.

d) Der 65C02-Prozessor wird erst dann richtig munter, wenn er mit mehr als 1 MHz getaktet ist, d.h. mit etwa einem 3,5-MHz-65C02 *und* der Verwendung der neuen Op-Codes laufen entsprechende Programme bis zu 400% schneller als vorher, womit ein 65C02C durchaus an die Leistungsfähigkeit von 16-Bit-Prozessoren heranreichen kann. Deshalb ist es bedauerlich und unverständlich, weshalb die Firma Apple den Apple IIe und Apple IIc nicht mit den schnelleren Prozessor-Varianten bestückt hat.

```
0300:          0300    1           ORG   $300
0300:                  2 *
0300:                  3 * 65C02C-Demo einiger neuer Op-Codes
0300:                  4 * ==================================
0300:                  5 *
0300:          0300    6           X6502               ;65C02C-Pseudo-Op
0300:                  7 *
0300:          00CE    8 IND       EQU   $CE
0300:          FDED    9 COUT      EQU   $FDED         ;PRINT
0300:          FDDA   10 PRBYTE    EQU   $FDDA         ;HEXOUT
0300:                 11 *
0300:                 12 *
0300:                 13 * Unbedingte, relative Verzweigung
0300:                 14 *
0300:80 01     0303   15           BRA   STETS         ;Verzeigung erfolgt stets
0302:00               16           BRK                 ;BRK wird übersprungen!
0303:                 17 *
0303:                 18 * Push/Pull X/Y on/from Stack
0303:                 19 *
0303:A2 01            20 STETS     LDX   #1
0305:A0 02            21           LDY   #2
0307:A9 03            22           LDA   #3
0309:DA               23           PHX                 ;Push X auf Stack
030A:5A               24           PHY                 ;Push Y auf Stack
030B:48               25           PHA
030C:68               26           PLA
030D:20 DA FD         27           JSR   PRBYTE
0310:7A               28           PLY                 ;Pull Y vom Stack
0311:98               29           TYA
0312:20 DA FD         30           JSR   PRBYTE
0315:FA               31           PLX                 ;Pull X vom Stack
0316:8A               32           TXA
0317:20 DA FD         33           JSR   PRBYTE
031A:                 34 *
031A:A9 8D            35           LDA   #$8D          ;Return
031C:20 ED FD         36           JSR   COUT
031F:                 37 *
031F:                 38 * $1000, $10FF, $00FE und $00FF auf 1 setzen
031F:                 39 *
031F:A9 01            40           LDA   #1
0321:A2 FF            41           LDX   #$FF
0323:8D 00 10         42           STA   $1000
0326:9D 00 10         43           STA   $1000,X
0329:85 FE            44           STA   $00FE
032B:95 00            45           STA   $0000,X
032D:                 46 *
032D:                 47 * $1000, $10FF, $00FE und $00FF auf 0 setzen
032D:                 48 * STZ = Store Zero entspricht LDA #$00 STA Adresse
032D:                 49 * A-Register bleibt jedoch unverändert!
032D:                 50 *
032D:9C 00 10         51           STZ   $1000
0330:9E 00 10         52           STZ   $1000,X
0333:64 FE            53           STZ   $00FE
0335:74 00            54           STZ   $0000,X
0337:20 DA FD         55           JSR   PRBYTE        ;A noch 1!
033A:A9 8D            56           LDA   #$8D          ;Return
033C:20 ED FD         57           JSR   COUT
033F:                 58 *
033F:                 59 * INC A (= INA) und DEC A (= DEA)
033F:                 60 * Increment/Decrement von A-Register
033F:                 61 *
033F:A9 C2            62           LDA   #'B'+$80
0341:20 ED FD         63           JSR   COUT          ;'B'
0344:1A               64           INC   A             ;A=A+1
0345:20 ED FD         65           JSR   COUT          ;'C'
0348:3A               66           DEC   A             ;A=A-1
0349:3A               67           DEC   A             ;A=A-1
034A:20 ED FD         68           JSR   COUT          ;'A'
034D:A9 8D            69           LDA   #$8D          ;Return
034F:20 ED FD         70           JSR   COUT
0352:                 71 *
0352:                 72 * Test Memory Bits with Accumulator
0352:                 73 *
0352:A9 F0            74           LDA   #%11110000
0354:1C 85 03         75           TRB   BYTE1         ;Reset Bits
0357:A9 0F            76           LDA   #%00001111
0359:0C 86 03         77           TSB   BYTE2         ;Set Bits
035C:AD 85 03         78           LDA   BYTE1
035F:20 DA FD         79           JSR   PRBYTE
0362:A9 A0            80           LDA   #$A0
0364:20 ED FD         81           JSR   COUT
0367:AD 86 03         82           LDA   BYTE2
036A:20 DA FD         83           JSR   PRBYTE
036D:60               84           RTS                 ;Ende Demo
036E:                 85 *
036E:                 86 * Indirekt Adressierung ohne Y
036E:                 87 *
036E:B2 CE            88           LDA   (IND)
0370:92 CE            89           STA   (IND)
0372:                 90 *
0372:12 CE            91           ORA   (IND)
0374:52 CE            92           EOR   (IND)
0376:32 CE            93           AND   (IND)
0378:                 94 *
0378:72 CE            95           ADC   (IND)
037A:F2 CE            96           SBC   (IND)
037C:                 97 *
037C:D2 CE            98           CMP   (IND)
037E:                 99 *
037E:                100 * Neue BIT-Adressierungsarten
037E:                101 *
037E:89 A0           102           BIT   #$A0
0380:34 A0           103           BIT   $A0,X
0382:3C 00 A0        104           BIT   $A000,X
0385:                105 *
0385:AA              106 BYTE1     DFB   %10101010
0386:AA              107 BYTE2     DFB   %10101010
```

Disassembler-Listing zu „65C02-Demo einiger neuer Op-Codes“

```
0300-  80 01        BRA   $0303
0302-  00           BRK
0303-  A2 01        LDX   #$01
0305-  A0 02        LDY   #$02
0307-  A9 03        LDA   #$03
0309-  DA           PHX
030A-  5A           PHY
030B-  48           PHA
030C-  68           PLA
030D-  20 DA FD     JSR   $FDDA
0310-  7A           PLY
0311-  98           TYA
0312-  20 DA FD     JSR   $FDDA
0315-  FA           PLX
0316-  8A           TXA
0317-  20 DA FD     JSR   $FDDA
031A-  A9 8D        LDA   #$8D
031C-  20 ED FD     JSR   $FDED
031F-  A9 01        LDA   #$01
0321-  A2 FF        LDX   #$FF
0323-  8D 00 10     STA   $1000
0326-  9D 00 10     STA   $1000,X
0329-  85 FE        STA   $FE
032B-  95 00        STA   $00,X
032D-  9C 00 10     STZ   $1000
0330-  9E 00 10     STZ   $1000,X
0333-  64 FE        STZ   $FE
0335-  74 00        STZ   $00,X
0337-  20 DA FD     JSR   $FDDA
033A-  A9 8D        LDA   #$8D
033C-  20 ED FD     JSR   $FDED
033F-  A9 C2        LDA   #$C2
0341-  20 ED FD     JSR   $FDED
0344-  1A           INC
0345-  20 ED FD     JSR   $FDED
0348-  3A           DEC
0349-  3A           DEC
034A-  20 ED FD     JSR   $FDED
034D-  A9 8D        LDA   #$8D
034F-  20 ED FD     JSR   $FDED
0352-  A9 F0        LDA   #$F0
0354-  1C 85 03     TRB   $0385
0357-  A9 0F        LDA   #$0F
0359-  0C 86 03     TSB   $0386
035C-  AD 85 03     LDA   $0385
035F-  20 DA FD     JSR   $FDDA
0362-  A9 A0        LDA   #$A0
0364-  20 ED FD     JSR   $FDED
0367-  AD 86 03     LDA   $0386
036A-  20 DA FD     JSR   $FDDA
036D-  60           RTS
036E-  B2 CE        LDA   ($CE)
0370-  92 CE        STA   ($CE)
0372-  12 CE        ORA   ($CE)
0374-  52 CE        EOR   ($CE)
0376-  32 CE        AND   ($CE)
0378-  72 CE        ADC   ($CE)
037A-  F2 CE        SBC   ($CE)
037C-  D2 CE        CMP   ($CE)
037E-  89 A0        BIT   #$A0
0380-  34 A0        BIT   $A0,X
0382-  3C 00 A0     BIT   $A000,X
0385-  AA           TAX
0386-  AA           TAX
```

Hüthig-FACHBUCH-TIP

Apple Assembler
Tips und Tricks

von Ulrich Stiehl
1984, 226 S., 3 Abb., kart.,
DM 34,-
ISBN 3-7785-1047-9
Hüthig Verlag, Heidelberg

„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.

Aus dem Inhalt:
6520-Repetitorium – Monitor-ROM – Speicherverwaltung – Applesoft-ROM – Text- und Grafikspeicher

BESTELLCOUPON

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

Buchtitel

Name

Straße

Ort

Unterschrift

65C02-Disassembler

für Apple IIe und II Plus

Dieser hier vorgestellte 65C02-Disassembler basiert auf dem im Jahre 1977 von Steve Wozniak geschriebenen 6502-Disassembler, der später für den Apple IIc gepatcht und von uns nunmehr für den Apple II Plus und IIe modifiziert wurde. Da unsere Variante die neuen Op-Codes selbst nicht benutzt, kann sie auch als "Cross"-Disassembler auf denjenigen Apple-Modellen eingesetzt werden, die nicht mit dem 65C02-Prozessor bestückt sind.

Wenn man sich in seinen Apple IIe oder Apple II Plus einen 65C02-Prozessor eingebaut hat oder wenn man eine Accelerator-IIe-Karte benutzt, wird einem erst richtig bewußt, wie wertvoll der im Monitor-ROM enthaltene Disassembler ist, denn dieser funktioniert jetzt nicht mehr. Genauer gesagt, werden die neuen Op-Codes des 65C02 als „???" gelistet. Der Apple IIc enthält zwar in dem umgeschriebenen Monitor-ROM einen für den 65C02 gepatchten Disassembler, aber dieser läßt sich nicht ohne weiteres in die Language Card des Apple IIe oder II Plus kopieren, da er teilweise Routinen überschreiben würde, die dringend benötigt werden, z.B. bei $FAB8 die PWRUP-Routine (Power-Up, Disk-Booten). Grundsätzlich gilt: Der ROM-Bereich $D000-$FFFF (Applesoft-Interpreter und Monitor) des Apple II und II Plus läßt sich in die Language Card des Apple IIe kopieren. Dagegen läßt sich weder der ROM-Bereich des Apple IIe noch der des IIc in die LC des Apple II oder II Plus schieben. Ferner kann auch der ROM-Bereich des Apple IIe nicht in die LC des Apple IIc verlegt werden und umgekehrt. Die Gründe hierfür liegen in dem INTCXROM-Bereich $C100-$CFFF, der beim Apple II Plus fehlt und beim Apple IIe und IIc völlig unterschiedliche Einsprungadressen hat.

Deshalb habe ich den im Apple IIc enthaltenen 65C02-Disassembler dergestalt modifiziert, daß er uneingeschränkt auf dem Apple IIe und II Plus lauffähig ist. Allerdings mußten aus Platzgründen die Lores-Grafik-Routinen teils überschrieben werden, die deshalb nicht benutzt werden dürfen, während der modifizierte Monitor in der Language Card aktiv ist. Notfalls muß man den Bereich $F82A-$F878 aus dem Lores-Bereich heraus in einen anderen Bereich verlegen.

Es wurde darauf geachtet, daß der Disassembler die neuen Op-Codes selbst nicht benutzt, so daß man damit sogar 65C02-Maschinenprogramme disassemblieren kann, wenn man *nicht* über einen 65C02-Prozessor verfügt.

Das Programm „65C02-DISASSEMBLER" gliedert sich in folgende Teile:

$6000-$604D enthält eine Move-Routine, die den eigentlichen Disassembler, der zunächst im Speicher ab $604E liegt, in die Language Card verschiebt.

$F82A-$F878 enthält eine Erweiterung des alten 6502-Disassemblers. Aus Platzgründen mußte die Program-Counter-Adjust-Routine nach $F86A verschoben werden, doch steht an der ursprünglichen Stelle von PCADJ = $F953 ein Sprung nach $F86A, so daß die Kompatibilität mit den anderen Monitor-Routinen gewahrt bleibt.

$F879-$FA3F enthält den Hauptteil des Apple IIc-Disassemblers mit einigen Patches für den Apple IIe und II Plus. Man beachte, daß es zwei Mnenomics-Tabellen gibt, und zwar zum einen die erweiterte alte Mnenomics-Tabelle ab $F9F9 sowie zum anderen eine neue Mnemonics-Tabelle ab $F853. Der 65C02-Disassembler berücksichtigt alle Mnemonics, die in den Programmzeilen 514-585 gelistet sind. Dazu gehören nicht die Befehle BBR, BBS, RMB und SMB, über die der Rockwell R65C02 zusätzlich verfügt; diese werden als „???" gelistet.

Für diejenigen, die den Disassembler - z.B. wegen der nicht berücksichtigen Op-Codes - ändern wollen, ist nachfolgend der Source-Code gelistet. Da man dieses Programm infolge der mehrfachen ORGs in den Zeilen 1 und 120 als Assembler-Hexdump nur mit Mühe eingeben könnte, ist außerdem noch ein „normaler" Hexdump hinzugefügt.

us

```
                        1           ORG  $6000
                        2   *
                        3   * 65C02-DISASSEMBLER
                        4   * ==================
                        5   *
                        6   ******************
                        7   *                *
                        8   *  Move-Routine  *
                        9   *                *
                       10   ******************
                       11   *
                       12   * APSOFT = Interpreter + Monitor
                       13   *          $D000-$FFFF
                       14   * REST   = Sonstiger Monitor
                       15   *          $FA40-$FFFF
                       16   *
                       17   APSOFT   EQU  $D000
                       18   REST     EQU  $FA40
                       19   IND1     EQU  $FC        ;-$FD
                       20   IND2     EQU  $FE        ;-$FF
                       21   *
                       22   * Language Card schreibfähig
                       23   * machen: Read ROM, Write LC
                       24   *
6000: AD 81 C0         25   MOVE1    LDA  $C081      ;RDROM
6003: AD 81 C0         26            LDA  $C081      ;WRRAM
                       27   *
                       28   * ROM ab $D000-$FFFF in LC
                       29   * kopieren: Applesoft + F8-ROM
                       30   *
6006: A0 00            31            LDY  #0
6008: 84 FC            32            STY  IND1
600A: A9 D0            33            LDA  #>APSOFT
600C: 85 FD            34            STA  IND1+1
600E: B1 FC            35   MOVE2    LDA  (IND1),Y   ;LOADROM
6010: 91 FC            36            STA  (IND1),Y   ;STORELC
6012: C8               37            INY
6013: D0 F9            38            BNE  MOVE2
6015: E6 FD            39            INC  IND1+1
6017: D0 F5            40            BNE  MOVE2      ;-$FFFF
                       41   *
                       42   * $F82A-$FA3F mit neuem
                       43   * Disassembler überschreiben
                       44   *
                       45   * IND1: $604E
                       46   *
6019: A9 4E            47            LDA  #<MOVE6+1
601B: 85 FC            48            STA  IND1
601D: A9 60            49            LDA  #>MOVE6
601F: 85 FD            50            STA  IND1+1
                       51   *
                       52   * IND2: $F82A
                       53   *
6021: A9 2A            54            LDA  #<NEU1
6023: 85 FE            55            STA  IND2
6025: A9 F8            56            LDA  #>NEU1
6027: 85 FF            57            STA  IND2+1
                       58   *
6029: A0 00            59            LDY  #0
602B: B1 FC            60   MOVE3    LDA  (IND1),Y
602D: 91 FE            61            STA  (IND2),Y
602F: E6 FC            62            INC  IND1
6031: D0 02            63            BNE  MOVE4
6033: E6 FD            64            INC  IND1+1
6035: E6 FE            65   MOVE4    INC  IND2
6037: D0 02            66            BNE  MOVE5
6039: E6 FF            67            INC  IND2+1
603B: A5 FF            68   MOVE5    LDA  IND2+1
603D: C9 FA            69            CMP  #>REST
603F: 90 EA            70            BCC  MOVE3      ;<$FA00
6041: A5 FE            71            LDA  IND2
6043: C9 40            72            CMP  #<REST
6045: 90 E4            73            BCC  MOVE3      ;<$FA40
                       74   *
                       75   * LC lese- UND schreibfähig
                       76   * machen: Bank 2 R/W enable
                       77   *
6047: AD 83 C0         78            LDA  $C083      ;RDLC
604A: AD 83 C0         79            LDA  $C083      ;WRLC
604D: 60               80   MOVE6    RTS             ;Exit
                       81   *
```

```
                       82   ****************************
                       83   *                          *
                       84   *  Apple IIc Disassembler  *
                       85   *                          *
                       86   *  für 65C02-Processor,    *
                       87   *                          *
                       88   *  umgeschrieben für IIe,  *
                       89   *                          *
                       90   *  II Plus von U.Stiehl    *
                       91   *                          *
                       92   ****************************
                       93   *
                       94   * Verwendbar auf Apple II Plus
                       95   * mit Language Card sowie auf
                       96   * Apple IIe. Disassemblierung
                       97   * ist sowohl möglich mit als
                       98   * auch ohne 65C02-Processor,
                       99   * da der Disassembler selbst
                      100   * die neuen Op-Codes NICHT
                      101   * benutzt!
                      102   *
                      103   * Achtung: Lores-Grafik-Routinen
                      104   * ab $F800 werden teils über-
                      105   * schrieben und dürfen nicht
                      106   * aufgerufen werden!
                      107   *
                      108   *-----------------------------
                      109   *
                      110   * Externe Routinen
                      111   *
                      112   PRYX2    EQU  $FD96
                      113   PRBYTE   EQU  $FDDA
                      114   COUT     EQU  $FDED
                      115   *
                      116   *-----------------------------
                      117   *
                      118   * Disassembler-Beginn: $F82A
                      119   *
                      120            ORG  $F82A
                      121   *
F82A: 98              122   NEU1     TYA
F82B: A2 16           123            LDX  #$16
F82D: DD 3C F8        124   NEU2     CMP  NEUTAB1,X
F830: F0 04           125            BEQ  NEU3
F832: CA              126            DEX
F833: 10 F8           127            BPL  NEU2
F835: 60              128            RTS
                      129   *
F836: BD 53 F8        130   NEU3     LDA  NEUTAB2,X
F839: A0 00           131            LDY  #$00
F83B: 60              132            RTS
                      133   *
                      134   *-----------------------------
                      135   *
F83C: 12              136   NEUTAB1  HEX  12         ;$00
F83D: 14              137            HEX  14
F83E: 1A              138            HEX  1A
F83F: 1C              139            HEX  1C
F840: 32              140            HEX  32
F841: 34              141            HEX  34
F842: 3A              142            HEX  3A
F843: 3C              143            HEX  3C
F844: 52              144            HEX  52
F845: 5A              145            HEX  5A
F846: 64              146            HEX  64
F847: 72              147            HEX  72
F848: 74              148            HEX  74
F849: 7A              149            HEX  7A
F84A: 7C              150            HEX  7C
F84B: 89              151            HEX  89
F84C: 92              152            HEX  92         ;10
F84D: 9C              153            HEX  9C
F84E: 9E              154            HEX  9E
F84F: B2              155            HEX  B2
F850: D2              156            HEX  D2
F851: F2              157            HEX  F2
F852: FC              158            HEX  FC
                      159   *
                      160   *-----------------------------
                      161   *
F853: 38              162   NEUTAB2  HEX  38         ;ORA
F854: 43              163            HEX  43         ;TRB
F855: 37              164            HEX  37         ;INC
F856: 43              165            HEX  43         ;TRB
F857: 39              166            HEX  39         ;AND
F858: 21              167            HEX  21         ;BIT
F859: 36              168            HEX  36         ;DEC
F85A: 21              169            HEX  21         ;BIT
```

```
F85B: 3A        170           HEX  3A          ;EOR
F85C: 40        171           HEX  40          ;PHY
F85D: 42        172           HEX  42          ;STZ
F85E: 3B        173           HEX  3B          ;ADC
F85F: 42        174           HEX  42          ;STZ
F860: 41        175           HEX  41          ;PLY
F861: 22        176           HEX  22          ;JMP
F862: 21        177           HEX  21          ;BIT
F863: 3C        178           HEX  3C          ;STA
F864: 42        179           HEX  42          ;STZ
F865: 42        180           HEX  42          ;STZ
F866: 3D        181           HEX  3D          ;LDA
F867: 3E        182           HEX  3E          ;CMP
F868: 3F        183           HEX  3F          ;SBC
F869: 44        184           HEX  44          ;???
                185 *
                186 *-------------------------------
                187 *
                188 * PC-Adjust aus Platzgründen
                189 * an diese Stelle verschoben.
                190 * Ursprünglich ab $F953,
                191 * wo jetzt ein JMP steht.
                192 *
F86A: 38        193 PCADJ     SEC              ;$F953
F86B: A5 2F     194 PCADJ2    LDA  $2F
F86D: A4 3B     195 PCADJ3    LDY  $3B
F86F: AA        196           TAX
F870: 10 01     197           BPL  PCADJ4
F872: 88        198           DEY
F873: 65 3A     199 PCADJ4    ADC  $3A
F875: 90 01     200           BCC  RTS2
F877: C8        201           INY
F878: 60        202 RTS2      RTS
                203 *
                204 *-------------------------------
                205 *
                206 * Apple IIc ROM-Kopie ab $F879
                207 * mit diversen Patches, da sich
                208 * die Opcodes TRB, PHY, PLY usw.
                209 * im IIc-ROM bei $FAB8-$FABC
                210 * und $FAF8-$FAFC befinden,
                211 * wo beim Apple II Plus und
                212 * IIe unentbehrliche Monitor-
                213 * Routinen stehen.
                214 *
F879: 90 04     215 SCRN2     BCC  RTMSKZ
F87B: 4A        216           LSR
F87C: 4A        217           LSR
F87D: 4A        218           LSR
F87E: 4A        219           LSR
F87F: 29 0F     220 RTMSKZ    AND  #$0F
F881: 60        221           RTS
F882: A6 3A     222 INSDS1    LDX  $3A
F884: A4 3B     223           LDY  $3B
F886: 20 96 FD  224           JSR  PRYX2
F889: 20 48 F9  225           JSR  PRBLNK
F88C: A1 3A     226           LDA  ($3A,X)
F88E: A8        227           TAY
F88F: 4A        228           LSR
F890: 90 05     229           BCC  INSDS1A
F892: 6A        230           ROR
F893: B0 0C     231           BCS  IEVEN
F895: 29 87     232           AND  #$87
F897: 4A        233 INSDS1A   LSR              ;neu
F898: AA        234           TAX
F899: BD 56 F9  235           LDA  FMT1,X
F89C: 20 79 F8  236           JSR  SCRN2
F89F: D0 04     237           BNE  GETFMT
F8A1: A0 FC     238 IEVEN     LDY  #$FC        ;$F8A1!
F8A3: A9 00     239           LDA  #$00
F8A5: AA        240 GETFMT    TAX              ;$F8A5!
F8A6: BD 9A F9  241           LDA  FMT2,X
F8A9: 85 2E     242           STA  $2E
F8AB: 29 03     243           AND  #$03
F8AD: 85 2F     244           STA  $2F
F8AF: 20 2A F8  245           JSR  NEU1        ;!!!
F8B2: F0 18     246           BEQ  RTSNEU
F8B4: 29 8F     247           AND  #$8F
F8B6: AA        248           TAX
F8B7: 98        249           TYA
F8B8: A0 03     250           LDY  #$03
F8BA: E0 8A     251           CPX  #$8A
F8BC: F0 0B     252           BEQ  MNNDX3
F8BE: 4A        253 MNNDX1    LSR
F8BF: 90 08     254           BCC  MNNDX3
F8C1: 4A        255           LSR
F8C2: 4A        256 MNNDX2    LSR
F8C3: 09 20     257           ORA  #$20
```

```
F8C5: 88        258           DEY
F8C6: D0 FA     259           BNE  MNNDX2
F8C8: C8        260           INY
F8C9: 88        261 MNNDX3    DEY
F8CA: D0 F2     262           BNE  MNNDX1
F8CC: 60        263 RTSNEU    RTS              ;neu
                264 *
F8CD: FF FF FF  265           HEX  FFFFFF      ;Füller
                266 *
                267 * Wird von $FE64 angesprungen
                268 *
F8D0: 20 82 F8  269 INSTDSP   JSR  INSDS1      ;extern
F8D3: 48        270           PHA
F8D4: B1 3A     271 PRNTOP    LDA  ($3A),Y
F8D6: 20 DA FD  272           JSR  PRBYTE
F8D9: A2 01     273           LDX  #$01
F8DB: 20 4A F9  274 PRNTBL    JSR  PRBL2
F8DE: C4 2F     275           CPY  $2F
F8E0: C8        276           INY
F8E1: 90 F1     277           BCC  PRNTOP
F8E3: A2 03     278           LDX  #$03
F8E5: C0 04     279           CPY  #$04
F8E7: 90 F2     280           BCC  PRNTBL
F8E9: 68        281           PLA
F8EA: A8        282           TAY
F8EB: B9 B4 F9  283           LDA  MNEML,Y
F8EE: 85 2C     284           STA  $2C
F8F0: B9 F9 F9  285           LDA  MNEMR,Y
F8F3: 85 2D     286           STA  $2D
F8F5: A9 00     287 PRMN1     LDA  #$00
F8F7: A0 05     288           LDY  #$05
F8F9: 06 2D     289 PRNM2     ASL  $2D
F8FB: 26 2C     290           ROL  $2C
F8FD: 2A        291           ROL
F8FE: 88        292           DEY
F8FF: D0 F8     293           BNE  PRNM2
F901: 69 BF     294           ADC  #$BF
F903: 20 ED FD  295           JSR  COUT
F906: CA        296           DEX
F907: D0 EC     297           BNE  PRMN1
F909: 20 48 F9  298           JSR  PRBLNK
F90C: A4 2F     299           LDY  $2F
F90E: A2 06     300           LDX  #$06
F910: E0 03     301 PRADR1    CPX  #$03
F912: F0 1C     302           BEQ  PRADR5
F914: 06 2E     303 PRADR2    ASL  $2E
F916: 90 0E     304           BCC  PRADR3
F918: BD AD F9  305           LDA  CHAR2,X
F91B: 20 ED FD  306           JSR  COUT
F91E: BD A7 F9  307           LDA  CHAR1,X
F921: F0 03     308           BEQ  PRADR3
F923: 20 ED FD  309           JSR  COUT
F926: CA        310 PRADR3    DEX
F927: D0 E7     311           BNE  PRADR1
F929: 60        312           RTS
F92A: 88        313 PRADR4    DEY
F92B: 30 E7     314           BMI  PRADR2
F92D: 20 DA FD  315           JSR  PRBYTE
F930: A5 2E     316 PRADR5    LDA  $2E
F932: C9 E8     317           CMP  #$E8
F934: B1 3A     318           LDA  ($3A),Y
F936: 90 F2     319           BCC  PRADR4
F938: 20 6D F8  320 RELADR    JSR  PCADJ3
F93B: AA        321           TAX
F93C: E8        322           INX
F93D: D0 01     323           BNE  PRNTYX
F93F: C8        324           INY
F940: 98        325 PRNTYX    TYA
F941: 20 DA FD  326           JSR  PRBYTE
F944: 8A        327           TXA
F945: 4C DA FD  328           JMP  PRBYTE
F948: A2 03     329 PRBLNK    LDX  #$03
F94A: A9 A0     330 PRBL2     LDA  #$A0
F94C: 20 ED FD  331           JSR  COUT
F94F: CA        332           DEX
F950: D0 F8     333           BNE  PRBL2
F952: 60        334           RTS
                335 *
                336 * Aus Platzgründen JMP nach
                337 * Program Counter Adjust
                338 * Wird von $FE67 angesprungen.
                339 *
F953: 4C 6A F8  340 PCADJJMP  JMP  PCADJ       ;extern
                341 *
F956: 0F        342 FMT1      HEX  0F          ;neu
F957: 22        343           HEX  22
F958: FF        344           HEX  FF
F959: 33        345           HEX  33
```

```
F95A: CB        346             HEX  CB
F95B: 62        347             HEX  62
F95C: FF        348             HEX  FF
F95D: 73        349             HEX  73
F95E: 03        350             HEX  03
F95F: 22        351             HEX  22
F960: FF        352             HEX  FF
F961: 33        353             HEX  33
F962: CB        354             HEX  CB
F963: 66        355             HEX  66
F964: FF        356             HEX  FF
F965: 77        357             HEX  77
F966: 0F        358             HEX  0F
F967: 20        359             HEX  20
F968: FF        360             HEX  FF
F969: 33        361             HEX  33
F96A: CB        362             HEX  CB
F96B: 60        363             HEX  60
F96C: FF        364             HEX  FF
F96D: 70        365             HEX  70
F96E: 0F        366             HEX  0F
F96F: 22        367             HEX  22
F970: FF        368             HEX  FF
F971: 39        369             HEX  39
F972: CB        370             HEX  CB
F973: 66        371             HEX  66
F974: FF        372             HEX  FF
F975: 7D        373             HEX  7D
F976: 0B        374             HEX  0B
F977: 22        375             HEX  22
F978: FF        376             HEX  FF
F979: 33        377             HEX  33
F97A: CB        378             HEX  CB
F97B: A6        379             HEX  A6
F97C: FF        380             HEX  FF
F97D: 73        381             HEX  73
F97E: 11        382             HEX  11
F97F: 22        383             HEX  22
F980: FF        384             HEX  FF
F981: 33        385             HEX  33
F982: CB        386             HEX  CB
F983: A6        387             HEX  A6
F984: FF        388             HEX  FF
F985: 87        389             HEX  87
F986: 01        390             HEX  01
F987: 22        391             HEX  22
F988: FF        392             HEX  FF
F989: 33        393             HEX  33
F98A: CB        394             HEX  CB
F98B: 60        395             HEX  60
F98C: FF        396             HEX  FF
F98D: 70        397             HEX  70
F98E: 01        398             HEX  01
F98F: 22        399             HEX  22
F990: FF        400             HEX  FF
F991: 33        401             HEX  33
F992: CB        402             HEX  CB
F993: 60        403             HEX  60
F994: FF        404             HEX  FF
F995: 70        405             HEX  70
F996: 24        406             HEX  24
F997: 31        407             HEX  31
F998: 65        408             HEX  65
F999: 78        409             HEX  78
F99A: 00        410 FMT2        HEX  00
F99B: 21        411             HEX  21
F99C: 81        412             HEX  81
F99D: 82        413             HEX  82
F99E: 59        414             HEX  59
F99F: 4D        415             HEX  4D
F9A0: 91        416             HEX  91
F9A1: 92        417             HEX  92
F9A2: 86        418             HEX  86
F9A3: 4A        419             HEX  4A
F9A4: 85        420             HEX  85
F9A5: 9D        421             HEX  9D
F9A6: 49        422             HEX  49
F9A7: 5A        423 CHAR1       HEX  5A         ;neu
F9A8: D9        424             HEX  D9
F9A9: 00        425             HEX  00
F9AA: D8        426             HEX  D8
F9AB: A4        427             HEX  A4
F9AC: A4        428             HEX  A4
F9AD: 00        429 CHAR2       HEX  00         ;neu
F9AE: AC        430             HEX  AC
F9AF: A9        431             HEX  A9
F9B0: AC        432             HEX  AC
F9B1: A3        433             HEX  A3
```

```
F9B2: A8        434             HEX  A8
F9B3: A4        435             HEX  A4
                436 *
                437 *-------------------------------
                438 *
F9B4: 1C        439 MNEML       HEX  1C         ;$00
F9B5: 8A        440             HEX  8A
F9B6: 1C        441             HEX  1C
F9B7: 23        442             HEX  23
F9B8: 5D        443             HEX  5D
F9B9: 8B        444             HEX  8B
F9BA: 1B        445             HEX  1B
F9BB: A1        446             HEX  A1
F9BC: 9D        447             HEX  9D
F9BD: 8A        448             HEX  8A
F9BE: 1D        449             HEX  1D
F9BF: 23        450             HEX  23
F9C0: 9D        451             HEX  9D
F9C1: 8B        452             HEX  8B
F9C2: 1D        453             HEX  1D
F9C3: A1        454             HEX  A1
F9C4: 1C        455             HEX  1C         ;$10
F9C5: 29        456             HEX  29
F9C6: 19        457             HEX  19
F9C7: AE        458             HEX  AE
F9C8: 69        459             HEX  69
F9C9: A8        460             HEX  A8
F9CA: 19        461             HEX  19
F9CB: 23        462             HEX  23
F9CC: 24        463             HEX  24
F9CD: 53        464             HEX  53
F9CE: 1B        465             HEX  1B
F9CF: 23        466             HEX  23
F9D0: 24        467             HEX  24
F9D1: 53        468             HEX  53
F9D2: 19        469             HEX  19
F9D3: A1        470             HEX  A1
F9D4: AD        471             HEX  AD         ;$20
F9D5: 1A        472             HEX  1A
F9D6: 5B        473             HEX  5B
F9D7: 5B        474             HEX  5B
F9D8: A5        475             HEX  A5
F9D9: 69        476             HEX  69
F9DA: 24        477             HEX  24
F9DB: 24        478             HEX  24
F9DC: AE        479             HEX  AE
F9DD: AE        480             HEX  AE
F9DE: A8        481             HEX  A8
F9DF: AD        482             HEX  AD
F9E0: 29        483             HEX  29
F9E1: 8A        484             HEX  8A
F9E2: 7C        485             HEX  7C
F9E3: 8B        486             HEX  8B
F9E4: 15        487             HEX  15         ;$30
F9E5: 9C        488             HEX  9C
F9E6: 6D        489             HEX  6D
F9E7: 9C        490             HEX  9C
F9E8: A5        491             HEX  A5
F9E9: 69        492             HEX  69
F9EA: 29        493             HEX  29
F9EB: 53        494             HEX  53
F9EC: 84        495             HEX  84
F9ED: 13        496             HEX  13
F9EE: 34        497             HEX  34
F9EF: 11        498             HEX  11
F9F0: A5        499             HEX  A5
F9F1: 69        500             HEX  69
F9F2: 23        501             HEX  23
F9F3: A0        502             HEX  A0         ;$3F
                503 *
                504 * Ergänzung der Tabelle
                505 *
F9F4: 8A        506             HEX  8A
F9F5: 8B        507             HEX  8B
F9F6: A5        508             HEX  A5
F9F7: AC        509             HEX  AC
F9F8: 00        510             HEX  00         ;$44
                511 *
                512 *-------------------------------
                513 *
F9F9: D8        514 MNEMR       HEX  D8         ;BRK
F9FA: 62        515             HEX  62         ;PHP
F9FB: 5A        516             HEX  5A         ;BPL
F9FC: 48        517             HEX  48         ;CLC
F9FD: 26        518             HEX  26         ;JSR
F9FE: 62        519             HEX  62         ;PLP
F9FF: 94        520             HEX  94         ;BMI
```

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1021-5

Der für den Apple II bestimmte Maskengenerator „Input 2.0" basiert auf den früheren Programmen „Input 1.0" und „Input 80 1.0" (von denen noch Restbestände lieferbar sind) und ist sowohl unter DOS 3.3 wie auch unter dem neuen ProDOS lauffähig. Der Maskengenerator setzt einen Apple II Plus mit Language Card oder einen Apple IIe voraus. Im 40 Z/Z-Modus funktioniert er auf beiden Gerätetypen, im 80 Z/Z-Modus dagegen nur auf dem Apple IIe mit 80-Zeichen-Karte. (Die alte Videx-Karte für den Apple II wird nicht unterstützt!)

„Input 2.0" liegt wahlweise in der Bank 1 oder Bank 2 der Language Card und wird durch einen kurzen Driver in den unteren 48K aufgerufen. „Input 2.0" läßt sich problemlos in nicht-compilierte und compilierte Applesoft- sowie in Assemblerprogramme einbinden. Die Übergabe der Feldinhalte an das Anwenderprogramm erfolgt durch ein einfaches Verfahren, das auch bei Compilern funktioniert.

Für jedes Feld der Bildschirmmaske lassen sich u. a. definieren: Feldlänge (bis zu 255 Zeichen) – Vtab – Htab – Datentyp (insgesamt 8 Typen) – Scrollflag (starre oder dynamische Maske) – Ctrlflag – Füllflag – Löschflag – Bildschirmflag (40- oder 80 Zeichendarstellung). Innerhalb eines Eingabefeldes besteht jeder denkbare Redigierkomfort (Insert, Delete, Rubout, Restore usw.).

Bei der neuen Version des Maskengenerators können jetzt auch Ctrl-Zeichen beim Datentyp String eingegeben werden. Ferner sind – das gilt nur für IIe – die Apfeltasten als schnelle Cursortasten definiert. Schließlich wurden Features implementiert, die den Einsatz von „Input 2.0" als zeilenorientiertes Textverarbeitungsprogramm ermöglichen. Die „Input 2.0"-Diskette enthält zahlreiche Demos zur Veranschaulichung der Anwendung.

Gerätevoraussetzung: Apple IIe oder IIc; ferner Apple II+ im 40-Zeichenmodus

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

```
FA00: 88        521          HEX  88          ;SEC
FA01: 54        522          HEX  54          ;RTI
FA02: 44        523          HEX  44          ;PHA
FA03: C8        524          HEX  C8          ;BVC
FA04: 54        525          HEX  54          ;CLI
FA05: 68        526          HEX  68          ;RTS
FA06: 44        527          HEX  44          ;PLA
FA07: E8        528          HEX  E8          ;BVS
FA08: 94        529          HEX  94          ;SEI
FA09: C4        530          HEX  C4          ;BRA
FA0A: B4        531          HEX  B4          ;DEY
FA0B: 08        532          HEX  08          ;BCC
FA0C: 84        533          HEX  84          ;TYA
FA0D: 74        534          HEX  74          ;LDY
FA0E: B4        535          HEX  B4          ;TAY
FA0F: 28        536          HEX  28          ;BCS
FA10: 6E        537          HEX  6E          ;CLV
FA11: 74        538          HEX  74          ;CPY
FA12: F4        539          HEX  F4          ;INY
FA13: CC        540          HEX  CC          ;BNE
FA14: 4A        541          HEX  4A          ;CLD
FA15: 72        542          HEX  72          ;CPX
FA16: F2        543          HEX  F2          ;INX
FA17: A4        544          HEX  A4          ;BEQ
FA18: 8A        545          HEX  8A          ;SED
FA19: 06        546          HEX  06          ;TSB
FA1A: AA        547          HEX  AA          ;BIT
FA1B: A2        548          HEX  A2          ;JMP
FA1C: A2        549          HEX  A2          ;JMP
FA1D: 74        550          HEX  74          ;STY
FA1E: 74        551          HEX  74          ;LDY
FA1F: 74        552          HEX  74          ;CPY
FA20: 72        553          HEX  72          ;CPX
FA21: 44        554          HEX  44          ;TXA
FA22: 68        555          HEX  68          ;TXS
FA23: B2        556          HEX  B2          ;TAX
FA24: 32        557          HEX  32          ;TSX
FA25: B2        558          HEX  B2          ;DEX
FA26: 72        559          HEX  72          ;PHX
FA27: 22        560          HEX  22          ;NOP
FA28: 72        561          HEX  72          ;PLX
FA29: 1A        562          HEX  1A          ;ASL
FA2A: 1A        563          HEX  1A          ;ROL
FA2B: 26        564          HEX  26          ;LSR
FA2C: 26        565          HEX  26          ;ROR
FA2D: 72        566          HEX  72          ;STX
FA2E: 72        567          HEX  72          ;LDX
FA2F: 88        568          HEX  88          ;DEC
FA30: C8        569          HEX  C8          ;INC
FA31: C4        570          HEX  C4          ;ORA
FA32: CA        571          HEX  CA          ;AND
FA33: 26        572          HEX  26          ;EOR
FA34: 48        573          HEX  48          ;ADC
FA35: 44        574          HEX  44          ;STA
FA36: 44        575          HEX  44          ;LDA
FA37: A2        576          HEX  A2          ;CMP
FA38: C8        577          HEX  C8          ;SBC
                578  *
                579  * Ergänzung der Tabelle
                580  *
FA39: 74        581          HEX  74          ;PHY
FA3A: 74        582          HEX  74          ;PLY
FA3B: 76        583          HEX  76          ;STZ
FA3C: C6        584          HEX  C6          ;TRB
FA3D: 00        585          HEX  00          ;???
                586  *
                587  * Füller bis $FA3F
                588  *
FA3E: EA        589          NOP
FA3F: EA        590  ENDE     NOP              ;$FA3F

--End assembly--

612 bytes

Errors: 0
```

65C02-Disassembler

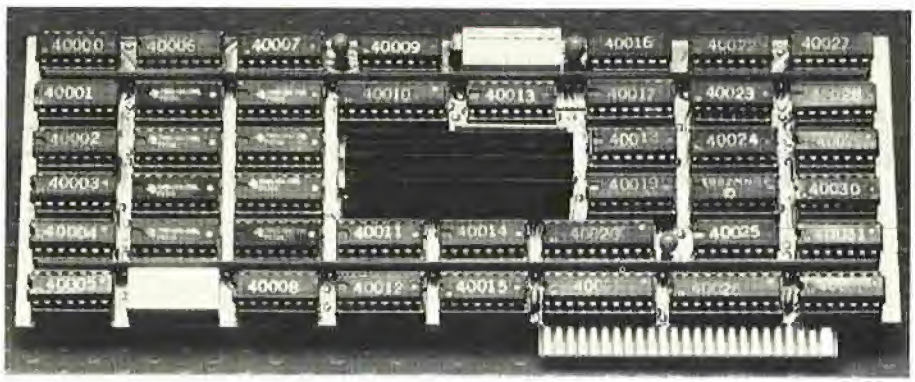
```
$6000: AD 81 C0 AD 81 C0 A0 00
$6008: 84 FC A9 D0 85 FD B1 FC
$6010: 91 FC C8 D0 F9 E6 FD D0
$6018: F5 A9 4E 85 FC A9 60 85
$6020: FD A9 2A 85 FE A9 F8 85
$6028: FF A0 00 B1 FC 91 FE E6
$6030: FC D0 02 E6 FD E6 FE D0
$6038: 02 E6 FF A5 FF C9 FA 90
$6040: EA A5 FE C9 40 90 E4 AD
$6048: 83 C0 AD 83 C0 60 98 A2
$6050: 16 DD 3C F8 F0 04 CA 10
$6058: F8 60 BD 53 F8 A0 00 60
$6060: 12 14 1A 1C 32 34 3A 3C
$6068: 52 5A 64 72 74 7A 7C 89
$6070: 92 9C 9E B2 D2 F2 FC 38
$6078: 43 37 43 39 21 36 21 3A
$6080: 40 42 3B 42 41 22 21 3C
$6088: 42 42 3D 3E 3F 44 38 A5
$6090: 2F A4 3B AA 10 01 88 65
$6098: 3A 90 01 C8 60 90 04 4A
$60A0: 4A 4A 4A 29 0F 60 A6 3A
$60A8: A4 3B 20 96 FD 20 48 F9
$60B0: A1 3A A8 4A 90 05 6A B0
$60B8: 0C 29 87 4A AA BD 56 F9
$60C0: 20 79 F8 D0 04 A0 FC A9
$60C8: 00 AA BD 9A F9 85 2E 29
$60D0: 03 85 2F 20 2A F8 F0 18
$60D8: 29 8F AA 98 A0 03 E0 8A
$60E0: F0 0B 4A 90 08 4A 4A 09
$60E8: 20 88 D0 FA C8 88 D0 F2
$60F0: 60 FF FF FF 20 82 F8 48
$60F8: B1 3A 20 DA FD A2 01 20
$6100: 4A F9 C4 2F C8 90 F1 A2
$6108: 03 C0 04 90 F2 68 A8 B9
$6110: B4 F9 85 2C B9 F9 F9 85
$6118: 2D A9 00 A0 05 06 2D 26
$6120: 2C 2A 88 D0 F8 69 BF 20
$6128: ED FD CA D0 EC 20 48 F9
$6130: A4 2F A2 06 E0 03 F0 1C
$6138: 06 2E 90 0E BD AD F9 20
$6140: ED FD BD A7 F9 F0 03 20
$6148: ED FD CA D0 E7 60 88 30
$6150: E7 20 DA FD A5 2E C9 E8
$6158: B1 3A 90 F2 20 6D F8 AA
$6160: E8 D0 01 C8 98 20 DA FD
$6168: 8A 4C DA FD A2 03 A9 A0
$6170: 20 ED FD CA D0 F8 60 4C
$6178: 6A F8 0F 22 FF 33 CB 62
$6180: FF 73 03 22 FF 33 CB 66
$6188: FF 77 0F 20 FF 33 CB 60
$6190: FF 70 0F 22 FF 39 CB 66
$6198: FF 7D 0B 22 FF 33 CB A6
$61A0: FF 73 11 22 FF 33 CB A6
$61A8: FF 87 01 22 FF 33 CB 60
$61B0: FF 70 01 22 FF 33 CB 60
$61B8: FF 70 24 31 65 78 00 21
$61C0: 81 82 59 4D 91 92 86 4A
$61C8: 85 9D 49 5A D9 00 D8 A4
$61D0: A4 00 AC A9 AC A3 A8 A4
$61D8: 1C 8A 1C 23 5D 8B 1B A1
$61E0: 9D 8A 1D 23 9D 8B 1D A1
$61E8: 1C 29 19 AE 69 A8 19 23
$61F0: 24 53 1B 23 24 53 19 A1
$61F8: AD 1A 5B 5B A5 69 24 24
$6200: AE AE A8 AD 29 8A 7C 8B
$6208: 15 9C 6D 9C A5 69 29 53
$6210: 84 13 34 11 A5 69 23 A0
$6218: 8A 8B A5 AC 00 D8 62 5A
$6220: 48 26 62 94 88 54 44 C8
$6228: 54 68 44 E8 94 C4 B4 08
$6230: 84 74 B4 28 6E 74 F4 CC
$6238: 4A 72 F2 A4 8A 06 AA A2
$6240: A2 74 74 74 72 44 68 B2
$6248: 32 B2 72 22 72 1A 1A 26
$6250: 26 72 72 88 C8 C4 CA 26
$6258: 48 44 44 A2 C8 74 74 76
$6260: C6 00 EA EA 00 00 00 00
```

Accelerator IIe

Die neue Superkarte

Die neue Accelerator IIe der Firma Titan (die früher als Saturn firmierte) kam etwa im April dieses Jahres auf den Markt. Ich wage zu behaupten: Wer diese Karte einmal in den Apple gesteckt hat, wird sie nicht mehr herausziehen wollen. In zweierlei Hinsicht ist sie wohl eine der besten Interface-Karten, die je für den Apple IIe bzw. II Plus konzipiert wurde. Einerseits ist deren Benutzung für den reinen Apple-Anwender denkbar unkompliziert: Karte in irgendeinen Slot stecken, einschalten, fertig! Andererseits kann ein Programmierer weitergehende Features der Karte ausnutzen, so daß sich vielfältige Anwendungsmöglichkeiten im Hinblick auf die Modifizierung des weiter unten geschilderten „Pseudo-ROMs" bieten.

Accelerator II



1. Hardware und Installation

Die Accelerator IIe, die ich für den stolzen Preis von $ 623.00 bzw. umgerechnet DM 1805,00 erwarb, dient zunächst nur einem einzigen Zweck, nämlich alle Programme bis zu ca. 3,5mal schneller zu machen. Die sorgfältig verarbeitete, sehr kompakte Karte ist u.a. mit einer 4 MHz NCR-65C02C-CPU (CMOS-Technologie) sowie mit 80K RAM (dynamisch, 150 ns) bestückt. Im Gegensatz zur alten Accelerator II kann die neue Accelerator IIe in jeden beliebigen Slot gesteckt werden. (Beim Apple II Plus wird jedoch wegen der Language Card Slot 0 empfohlen.) Die alte Accelerator II mußte sich links von einer ggf. vorhandenen Z80-Karte befinden. Diese Beschränkung gilt für die Accelerator IIe nicht mehr. Ferner gab es bei der Accelerator II verschiedene Preboot-Konfigurationen (Phantom 0, Fast Applesoft, Standard und Disable), und der damaligen Anleitung war eine ellenlange Liste von (geschützten) Programmen beigefügt worden, aus der ersichtlich war, welches Programm welche Konfiguration erforderte. Auch diese Beschränkung ist bei der neuen Accelerator IIe insoweit entfallen, als erstens eine Preboot-Diskette normalerweise überflüssig ist und zweitens grundsätzlich *alle* Programme lauffähig sind, auch die „geschützten".

Die Karte verfügt über 8 Schalter und 7 Brücken. Die Schalter 1-7 sind den Slots 1-7 zugeordnet. Bei den Schaltern bedeutet ON = 4 MHz und OFF = 1 MHz. Bei zeitkritischen Interface-Karten (Disk-Controller, Modem usw., nicht dagegen z.B. Druckerkarten usw.) muß der entsprechende Slot-Schalter auf OFF (nach unten) gestellt werden. Dies gilt nicht für die 80-Zeichenkarte bzw. 64K-Karte des Apple IIe. Mit dem Schalter Nr. 8 kann man die Accelerator IIe insgesamt permanent auf 1 MHz herunterschalten (Dies entspricht dem Softswitch C086:1).

Die Brücken kontrollieren ggf. vorhandene RAM-Karten. Wenn sich in irgendeinem Slot eine RAM-Karte (128K-RAM-Karte usw.) befinden sollte, muß die entsprechende Slot-Brücke (ein kleiner Plastik-Aufsatz) abgezogen werden. Dies gilt wiederum nicht für die IIe-64K-RAM-Karte in Slot 3.

Da die CMOS-Teile der Karte sehr empfindlich gegen Elektrostatik sind, empfiehlt es sich, die Karte in eine Alu-Folie einzuwickeln, falls sie versandt werden soll. Bevor man die Karte anfaßt, z.B. wenn man sie in einen Slot steckt oder einen Schalter umlegt, sollte man die Hände an einem Metallstück „entladen".

Die Accelerator IIe wird denkbar einfach aktiviert: Man braucht lediglich den Apple einzuschalten; das ist alles. In dem Moment, da der Apple eingeschaltet wird, zieht die Karte quasi „alle Energie" an sich, d.h. der alte 6502 sowie die gesamten 64K RAM des Motherboards inklusive LC *und* das 12K-ROM ($D000-$FFFF) werden desaktiviert. Wie bereits erwähnt, umfaßt die Accelerator IIe 80K RAM, das sich wie folgt aufteilt:

– 48K „Motherboard"
– 16K „Language Card"
– 16K „Pseudo-ROM"

Beim Einschalten des Apple wird der echte ROM $D000-$FFFF in das Pseudo-ROM der Accelerator IIe kopiert, worauf softswitchmäßig „die Klappe herunterfällt", d.h. das Pseudo-ROM, das in Wirklichkeit RAM ist, wird gegen Überschreiben geschützt.

2. Geschwindigkeit

Der 4 MHz 65C02C läuft theoretisch auf 3,5 MHz heruntergetaktet, doch konnte durch detaillierte Testprogramme gezeigt werden, daß die Accelerator-CPU maximal 3,32mal schneller als die normale 6502-CPU ist. Lediglich wenn von den erweiterten 65C02-Befehlen Gebrauch gemacht wird, können Programme den Wert von 3,5 erreichen bzw. gar überschreiten. Im einzelnen wird die Accelerator IIe in folgenden Fällen kurzfristig auf 1 MHz „heruntergetaktet":

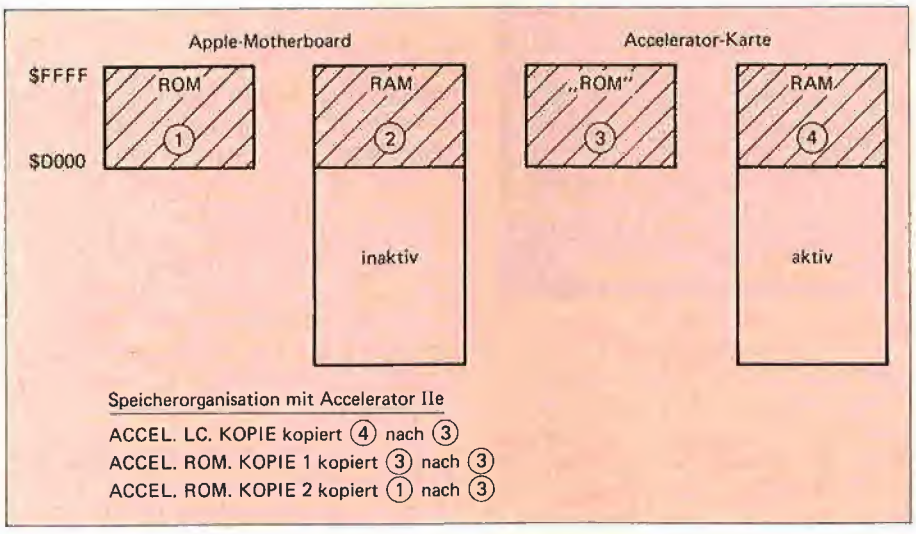
– beim Zugriff auf Peripheriegeräte wie Disk-Controller usw., bei denen der entsprechenden Slot-Schalter von vornherein auf 1 MHz eingestellt wurde.
– beim Zugriff auf RAM-Karten aller Art, insbesondere wenn diese virtuell in den Bereich $D000-$FFFF „gemappt" sind.
– beim Zugriff auf Softswitches und ROM-Routinen im Bereich $C000-$CFFF. Das Demo-Programm **„ACCEL.WAIT"** zeigt, daß das Ansprechen von INTCXROM- oder Softswitch-Adressen nur dann zur einer spürbaren Reduzierung der Geschwindigkeit führt, wenn dies mit einer nicht bis ins letzte Detail untersuchten gewissen Häufigkeit geschieht.

Praktische Beispiele

a) Das an anderer Stelle in dieser Zeitschrift gelistete Applesoft-Primzahlen-Programm ist mit der Accelerator-Karte 3,28mal schneller.

b) Die Bildschirm-Scroll-Geschwindigkeit ist speziell beim Apple IIe mit Accelerator dann kaum spürbar schneller, wenn man den IIe-Monitor benutzt, der bekanntlich permanent in den INTCXROM-Bereich $C100-$CFFF springt. Lädt man dagegen z.B. den Apple II Plus FPBASIC-File von der System Master Diskette in die Accelerator Language Card, dann ist etwa die LIST-Geschwindigkeit bei Applesoftprogrammen ca. 2,8mal schneller. Ähnliches gilt für die 80-Zeichen-Scroll-Geschwindigkeit beim Applewriter IIe, da dieser die INTCXROM-Routinen nicht benutzt.

c) Der Zugriff auf die 64K-Karte des Apple IIe ist mit Accelerator ca. 2mal schneller. Die Datenübertragungsrate von der und auf die Karte beträgt maximal 80K/s. Dies entspricht der kaleidoskopartigen Darstellung von 10 Hires-Bildern pro Sekunde.



(Zu Testzwecken wurde das AUXMOVER-Programm von der MMU-Diskette benutzt, das mit der Accelerator IIe zu einer überzeugenden „Bilderschau" führt.)

In der Anleitung zur Accelerator IIe, die leider für Programmierer und Techniker etwas dürftig ist, sind nur 3 Softswitches genannt:

C086: 1 = POKE -16250, 1 = 1 MHz (Low Speed; für Spiele usw.)

C086: 5 = POKE -16250, 5 = 3,5 MHz (High Speed; Normalzustand)

(Zwischen dem langsamen und schnellen Modus kann man beliebig oft hin- und herschalten.)

C086: A = POKE -16250, 10 (Disable; Karte abstellen)

Das Abstellen der Karte bedeutet, daß nunmehr wieder das eigentliche RAM und ROM des Apple aktiv wird. Zum Abstellen der Karte verwende man entweder die mitgelieferte Preboot-Diskette oder verfahre beim Apple IIe wie folgt:
CALL -151
C086:A
Ctrl-Geschlossener Apfel-Reset
(Ctrl-Taste, schwarze Apfel-Taste und Reset-Taste gleichzeitig drücken. Dieses Verfahren bewirkt, daß sich die Softswitches normalisieren.)

Ist die Karte erst einmal desaktiviert, besteht meines Wissens keine Möglichkeit mehr, sie wieder zu reaktivieren, d.h. man ist gezwungen, den Apple erst aus- und dann wieder einzuschalten, wenn die Accelerator wieder aktiv werden soll.

3. Reset

Der Hardware-Reset durch Drücken der Ctrl-Reset-Tasten bewirkt beim Apple IIe stets die Normalisierung der Softswitches in folgender Weise: Es werden die „unteren" 48K des Motherboards lese- *und* schreibfähig gemacht, also eine vor Reset ggf. aktive 64K-Kartes desaktiviert. Ferner wird das ROM $D000-$FFFF lesefähig und die Bank 2 der Language Card der unteren 64K schreibfähig gemacht. Schließlich werden durch die Software-Reset-Routine ab $FA62 Grafik und 80-Zeichenkarte abgestellt.
Für die Accelerator IIe gilt demgegenüber, daß eine ggf. aktive (lese-schreib-fähig gemachte) Language Card durch Ctrl-Reset nicht desaktiviert wird. Ferner „dreht" die Accelerator IIe „durch", wenn man Ctrl-Reset drückt, während sich die 64K-Karte in einem Read-Write-Enable-Zustand befindet. Obwohl dies nicht eindeutig bewiesen werden kann, dürfte in diesem Fall kurzfristig das Motherboard-RAM und -ROM aktiviert werden. Da sich dort jedoch nur „Zufallswerte" befinden, wird neu gebootet. Wenn man danach mit Ctrl-Schwarzer Apfel-Reset die Softswitches normalisiert, gerät die Accelerator IIe wieder unter Kontrolle. Andernfalls kann man nur noch das Gerät ausschalten. Ein Beispiel: Der Applewriter IIe benutzt – falls vorhanden – die 64K-Karte als Textspeicher und muß deshalb beim Scrollen usw. häufig auf die „oberen" 64K springen. Drückt man dann in einem solchen Moment Ctrl-Reset, wird neu gebootet.

4. Die Geheimnisse des „Pseudo-ROMs"

In der Betriebsanleitung der Firma Titan wird zwar beiläufig erwähnt, daß nach dem Einschalten des Apple das „richtige" ROM in das „Pseudo-ROM" der Accelerator-Karte kopiert wird, doch wie das geschieht und insbesondere, ob dieser Kopiervorgang später, *während* der Apple *noch* eingeschaltet ist, wiederholt werden kann, darüber hüllt sich Titan in Schweigen. Die Vorstellung, ggf. in dieses Pseudo-ROM einen anderen Inhalt hineinladen zu können, faszinierte mich, denn dann wäre man allen „F8-EPROM-Brennern" meilenweit überlegen, könnte man doch je nach Anwendungszweck mal diesen und mal jenen Monitor oder mal dieses und mal jenes Applesoft oder Integer-Basic usw. in das Pseudo-ROM laden. Es hat mich dann zwei schlaflose Nächte gekostet, bis ich herausfand, daß dies tatsächlich möglich ist. Insgesamt habe ich 256 verschiedene Softswitches ausprobieren müssen, wobei mir im übrigen „Kommissar Zufall" zu Hilfe kam. Durch Untersuchung der Speicherbereiche nach dem Einschalten des Apple stellte ich nämlich fest, daß ein echtes Zusatz-ROM der Accelerator-Karte ein kurzes Kopierprogramm in den Stack-Bereich $0100-$01FF verlegt. Obwohl sich dieses Programm nachher durch die übliche Verwen-

dung des Stacks von selbst überschreibt, konnte noch der wichtigste Teil rekonstruiert werden. Dieser ist in dem Programm **„ACCEL.BOOT"** gelistet. Weitere Untersuchungen ergaben dann folgendes:

```
LDA #3
STA $C086
LDA $C081
LDA $C081
```

macht das Pseudo-ROM der Accelerator IIe schreibfähig (write-enable) und zugleich das echte ROM des Apple lesefähig (read-enable).

```
LDA #5
STA $C086
LDA $C081
LDA $C081
```

macht das Pseudo-ROM der Accelerator IIe (wieder) schreibunfähig (write-disable) und zugleich das echte ROM des Apple leseunfähig (read-disable).

Damit war es möglich, folgende Programme zu schreiben:

ACCEL.LC.KOPIE kopiert den Inhalt der Accelerator Language Card Bank 2 in das Accelerator Pseudo-ROM. Zu diesem Zweck muß zuvor FPBASIC usw. in die Language Card geladen worden sein. *Nach* BRUN ACCEL.LC.KOPIE kann die Language Card dann anderweitig verwendet werden. ACCEL.LC.KOPIE kopiert zunächst den LC-Inhalt $D000-$FFFF nach $1000-$3FFF und dann wieder hinauf in das Pseudo-ROM. Anwendungszweck:

ACCEL.ROM.KOPIE1 kopiert zunächst den Inhalt des Pseudo-ROMs $D000-$FFFF nach $1000-$3FFF. Nunmehr kann man den herunterkopierten Bereich bequem patchen. Im Anschluß daran wird dann der Bereich $1000-$3FFF wieder in das Pseudo-ROM zurückkopiert. Anwendungszweck: Verwendung eines partiell modifizierten Pseudo-ROM-Inhalts.

ACCEL.ROM.KOPIE2 kopiert auf dem Umweg über $1000-$3FFF den echten Apple-ROM-Inhalt (wieder) in das Pseudo-ROM der Accelerator-Karte. Anwendungszweck: Wiederherstellung des Kaltstart-Zustandes.

Nach meinen Erfahrungen scheint es geboten zu sein, die Kopierroutinen auf dem Umweg über einen Bereich in den unteren 48K, z.B. $1000-$3FFF, vorzunehmen, da sich die Accelerator-Karte nach LDA #3 STA $C086 in einem partiellen Schwebezustand befindet, der durch permanente Softswitch-Aktivierung zu einem „Durchdrehen" der Karte führen kann.

Ein praktisches Beispiel:

Nehmen wir an, wir wollten den 65C02-Disassembler (Dateiname „D65C02") im Pseudo-ROM der Accelerator-Karte „verewigen", damit die Language Card für andere Zwecke wieder frei wird. Dies könnten wir mit folgenden Applesoft-Programm realisieren:

```
10 PRINT CHR$ (4) „BRUN ACCEL.
   ROM. KOPIE2"
20 PRINT CHR$ (4) „BRUN D65C02"
30 PRINT CHR$ (4) „BRUN ACCEL.
   LC.KOPIE"
```

Zeile 10 stellt zunächst sicherheitshalber eine „saubere" Pseudo-ROM-Kopie des Original-Apple-ROMs her. Zeile 20 kopiert dann das Pseudo-ROM einschließlich des neuen 65C02-Disassemblers in die Language Card. Und Zeile 30 kopiert dann die Language Card in das Pseudo-ROM. Diese Prozedur ließe sich natürlich dadurch vereinfachen, daß man das Move-Programm des Disassemblers dergestalt umschreibt, daß man den neuen Disassembler direkt in das Pseudo-ROM kopiert.

Es sei abschließend darauf hingewiesen, daß die Accelerator IIe mit diesem Beitrag noch lange nicht endgültig erforscht ist. Beispielsweise besteht u.U. die Möglichkeit, auch die Language Card des Apple-Motherboards (neben der LC der Accelerator-Karte) zur Datenspeicherung zu verwenden. Ob man allerdings auch die restlichen 48K des Motherboards als Datenspeicher benutzen kann, scheint zweifelhaft, wenngleich nicht unmöglich.

us

```
                        1             ORG  $300
                        2     *
                        3     * ACCEL.WAIT
                        4     * ==========
                        5     *
                        6     ADR1      EQU  $C000
                        7     ADR2      EQU  $FFFF
                        8     *
                        9     * Wenn ADR1 im Bereich
                       10     * $C000-$CFFF liegt, dann
                       11     * beträgt die Warteschleife
                       12     * ca. 30 s statt ca. 20 s
                       13     * bei einer beliebigen anderen
                       14     * ADR1. Wenn zusätzlich ADR2
                       15     * im Bereich $C000-$CFFF liegt,
                       16     * beträgt Warteschleife ebenfalls
                       17     * ca. 30 s. Wenn aber nur ADR2
                       18     * allein im Bereich $C000-$CFFF
                       19     * liegt, beträgt die Warteschleife
                       20     * wieder ca. 20 s. Fazit:
                       21     * Ein spürbare Zeitverzögerung
                       22     * ist nur dann gegeben, wenn
                       23     * eine Adresse im Bereich
                       24     * $C000-$CFFF quasi permanent
                       25     * angesprochen wird.
                       26     *
0300: A2 6F            27             LDX  #111
0302: 20 09 03         28     WAIT1   JSR  WAIT2
0305: CA               29             DEX
0306: D0 FA            30             BNE  WAIT1
0308: 60               31             RTS
0309: 8E 20 03         32     WAIT2   STX  XSAVE
030C: A2 00            33             LDX  #0
030E: A0 00            34             LDY  #0
0310: AD 00 C0         35     WAIT3   LDA  ADR1       ;Adr1
0313: CA               36             DEX
0314: D0 FA            37             BNE  WAIT3
0316: AD FF FF         38             LDA  ADR2       ;Adr2
0319: 88               39             DEY
031A: D0 F4            40             BNE  WAIT3
031C: AE 20 03         41             LDX  XSAVE
031F: 60               42             RTS
0320: 00               43     XSAVE   HEX  00
```

```
                        1             ORG  $0180
                        2     *
                        3     * ACCEL.BOOT
                        4     * ==========
                        5     *
                        6     * Diese Routine wird von einem
                        7     * nicht näher bekannten ROM-
                        8     * Bauelement der Accelerator-
                        9     * Karte nach dem Einschalten
                       10     * des Apple in den Stack
                       11     * kopiert durch eine nachfolgend
                       12     * nicht aufgeführte Routine,
                       13     * die ihrerseits im Stack
                       14     * ab $0100 liegt.
                       15     * Danach erfolgt ein Sprung
                       16     * nach $0180 zu der eigentlichen
                       17     * hier disassemblierten Routine,
                       18     * die das echte ROM $D000-$FFFF
                       19     * in das Schein-ROM der Accelerator-
                       20     * karte movt.
                       21     *
                       22     *
                       23     INDL      EQU  $04
                       24     INDH      EQU  $05
                       25     ACCEL     EQU  $C086
                       26     WRROMBK1  EQU  $C08A
                       27     WRROMBK2  EQU  $C082
                       28     RESET     EQU  $FA62
                       29     ROM       EQU  $D000
                       30     *
                       31     * LDA #$03 STA $C086 ist ein
                       32     * in der Anleitung von Titan nicht
                       33     * dokumentierter Softswitch, um
                       34     * ein WRITE-ENABLE der ROM-16K
                       35     * der Karte zu ermöglichen.
                       36     *
0180: A9 03            37             LDA  #$03
0182: 8D 86 C0         38             STA  ACCEL
                       39     *
0185: A9 D0            40             LDA  #>ROM
0187: 85 05            41             STA  INDH
0189: A0 00            42             LDY  #<ROM
018B: 84 04            43             STY  INDL
018D: B1 04            44     KOPIE   LDA  (INDL),Y
018F: 91 04            45             STA  (INDL),Y
0191: AE 8A C0         46             LDX  WRROMBK1
0194: 91 04            47             STA  (INDL),Y
0196: AE 82 C0         48             LDX  WRROMBK2
0199: 88               49             DEY
019A: D0 F1            50             BNE  KOPIE
019C: E6 05            51             INC  INDH
019E: D0 ED            52             BNE  KOPIE
                       53     *
                       54     * LDA #$05 STA $C086 ist der
                       55     * dokumentierte Softswitch,
                       56     * der auf 3,5 Megahertz
                       57     * "Full Speed" umschaltet.
                       58     *
                       59     * LDA #$01 STA $C086 schaltet
                       60     * demgegenüber auf 1 Megahertz
                       61     * herunter. Das RAM der Karte
                       62     * ist danach jedoch weiterhin
                       63     * aktiv.
                       64     *
                       65     * LDA #$0A STA $C086 schaltet
                       66     * zusätzlich das RAM der Karte
                       67     * ab, so daß nunmehr das normale
                       68     * RAM des Apple aktiv ist. Da
                       69     * sich hier jedoch nur Zufallswerte
                       70     * befinden, muß man nach diesem
                       71     * Softswitch mit Reset neu booten.
                       72     * Danach besteht offenbar keine
                       73     * Möglichkeit mehr, die Karte zu
                       74     * reaktivierten, d.h. man muß
                       75     * den Apple zunächst wieder
                       76     * ausschalten, wenn man die
                       77     * Karte wieder aktivieren will.
                       78     *
                       79     *
01A0: A9 05            80             LDA  #$05
01A2: 8D 86 C0         81             STA  ACCEL
01A5: 4C 62 FA         82             JMP  RESET
```

```
                        1             ORG  $300
                        2     *
                        3     * ACCEL.LC.KOPIE
                        4     * ==============
                        5     *
                        6     * Diese Routine kopiert zunächst
                        7     * den Accelerator-LC-Inhalt Bank2
                        8     * $D000-$FFFF nach $1000-$3FFF
                        9     * und kopiert dann $1000-$3FFF
                       10     * in das Accelerator-"ROM".
                       11     *
                       12     * Warnung: Vorher muß FPBASIC
                       13     * oder INTBASIC usw. in die LC
                       14     * Bank2 geladen worden sein!
                       15     *
                       16     * Applesoft-Beispiel:
                       17     * -------------------
                       18     *
                       19     * 10 X = PEEK (49281): X = PEEK (49281)
                       20     * 20 PRINT CHR$ (4) "BLOAD FPBASIC, A$D000"
                       21     * 30 PRINT CHR$ (4) "BRUN ACCEL.LC.KOPIE"
                       22     *
                       23     INDADR1   EQU  $D000
                       24     INDADR2   EQU  $1000       ;Anfang
                       25     INDADR3   EQU  $4000       ;Ende+1
                       26     IND1      EQU  $FC         ;-$FD
                       27     IND2      EQU  $FE         ;-$FF
                       28     ACCEL     EQU  $C086
                       29     *
                       30     *--------Kopie nach unten---------
                       31     *
```

```
                    32   * Pointer initialisieren
                    33   *
0300: A9 00         34            LDA  #<INDADR1  ;$D000
0302: 85 FC         35            STA  IND1
0304: A9 D0         36            LDA  #>INDADR1
0306: 85 FD         37            STA  IND1+1
                    38   *
0308: A9 00         39            LDA  #<INDADR2  ;$1000
030A: 85 FE         40            STA  IND2
030C: A9 10         41            LDA  #>INDADR2
030E: 85 FF         42            STA  IND2+1
                    43   *
                    44   * Accelerator-LC Bank 2 lese-
                    45   * fähig machen.
                    46   *
0310: A9 05         47            LDA  #5         ;ACCEL:5
0312: 8D 86 C0      48            STA  ACCEL
0315: AD 83 C0      49            LDA  $C083      ;RDBK2
0318: AD 83 C0      50            LDA  $C083      ;WRBK2
                    51   *
                    52   * $D000-$FFFF nach $1000-$3FFF kopieren
                    53   *
031B: A0 00         54            LDY  #0
031D: B1 FC         55   LOOP1    LDA  (IND1),Y
031F: 91 FE         56            STA  (IND2),Y
0321: C8            57            INY
0322: D0 F9         58            BNE  LOOP1
0324: E6 FD         59            INC  IND1+1
0326: E6 FF         60            INC  IND2+1
0328: A5 FD         61            LDA  IND1+1
032A: D0 F1         62            BNE  LOOP1
                    63   *
032C: AD 81 C0      64            LDA  $C081      ;RDROMBK2
032F: AD 81 C0      65            LDA  $C081      ;WRRAMBK2
                    66   *
                    67   *--------Kopie nach oben---------
                    68   *
                    69   * Pointer initialisieren
                    70   *
0332: A9 00         71            LDA  #<INDADR2  ;$1000
0334: 85 FC         72            STA  IND1
0336: A9 10         73            LDA  #>INDADR2+1
0338: 85 FD         74            STA  IND1+1
                    75   *
033A: A9 00         76            LDA  #<INDADR1  ;$D000
033C: 85 FE         77            STA  IND2
033E: A9 D0         78            LDA  #>INDADR1+1
0340: 85 FF         79            STA  IND2+1
                    80   *
                    81   * "ROM" der Accelerator-Karte
                    82   * schreibfähig machen!
                    83   *
0342: A9 03         84            LDA  #3
0344: 8D 86 C0      85            STA  ACCEL
0347: AD 81 C0      86            LDA  $C081      ;RDROMBK2
034A: AD 81 C0      87            LDA  $C081      ;WRRAMBK2
                    88   *
                    89   * $1000-$3FFF nach $D000-$FFFF,
                    90   * d.h. in das Accelerator-"ROM"
                    91   * kopieren!
                    92   *
034D: A0 00         93            LDY  #0
034F: B1 FC         94   LOOP2    LDA  (IND1),Y
0351: 91 FE         95            STA  (IND2),Y
0353: C8            96            INY
0354: D0 F9         97            BNE  LOOP2
0356: E6 FD         98            INC  IND1+1
0358: E6 FF         99            INC  IND2+1
035A: A5 FD        100            LDA  IND1+1
035C: C9 40        101            CMP  #>INDADR3
035E: D0 EF        102            BNE  LOOP2
                   103   *
                   104   * "ROM" der Accelerator wie quasi
                   105   * echtes ROM schreibunfähig machen
                   106   *
0360: A9 05        107            LDA  #5         ;ACCEL:5
0362: 8D 86 C0     108            STA  ACCEL
0365: AD 81 C0     109            LDA  $C081      ;RDROMBK2
0368: AD 81 C0     110            LDA  $C081      ;WRRAMBK2
                   111   *
036B: 60           112            RTS
```

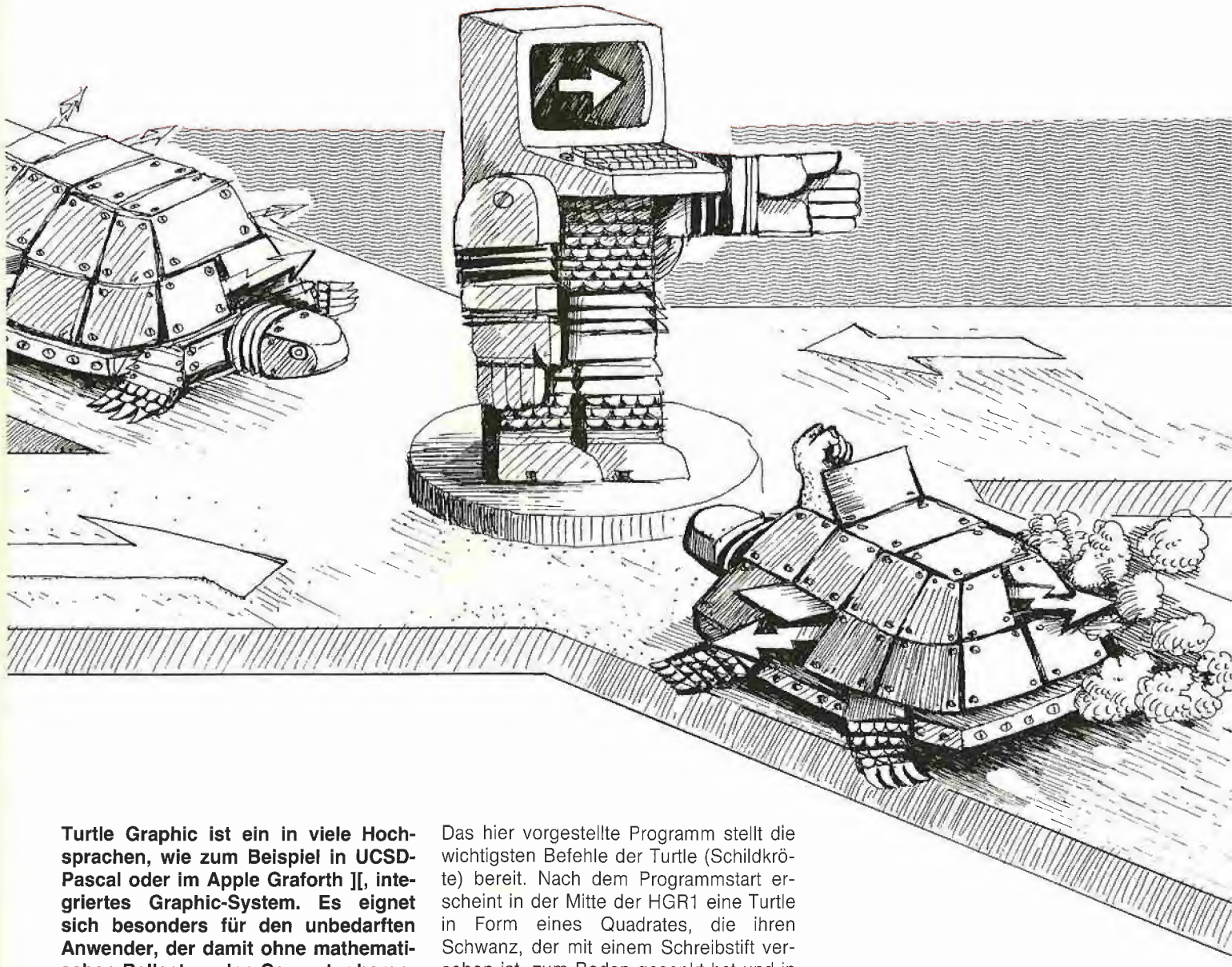
```
10  PRINT  CHR$ (4)"BLOAD ACCEL.LC.KOPIE"
20 X =  PEEK (49281):X =  PEEK (49281)
30  INPUT "LC-FILE:";X$
40  PRINT  CHR$ (4)"BLOAD";X$;",A$D000"
50  CALL 768
```

```
                        1               ORG  $300
                        2   *
                        3   * ACCEL.ROM.KOPIE1 (Pseudo-ROM)
                        4   * ========================
                        5   *
                        6   * Diese Routine kopiert zunächst
                        7   * den Accelerator-"ROM"-Inhalt
                        8   * $D000-$FFFF nach $1000-$3FFF
                        9   * und kopiert dann $1000-$3FFF
                       10   * erneut in das Accelerator-"ROM",
                       11   * nachdem man (bei Bedarf) das
                       12   * spätere "ROM" entsprechend
                       13   * gepatcht hat.
                       14   *
                       15   INDADR1  EQU  $D000
                       16   INDADR2  EQU  $1000
                       17   INDADR3  EQU  $4000
                       18   IND1     EQU  $FC          ;-$FD
                       19   IND2     EQU  $FE          ;-$FF
                       20   ACCEL    EQU  $C086
                       21   *
                       22   *--------Kopie nach unten---------
                       23   *
                       24   * Pointer initialisieren
                       25   *
0300: A9 00            26               LDA  #<INDADR1  ;$D000
0302: 85 FC            27               STA  IND1
0304: A9 D0            28               LDA  #>INDADR1
0306: 85 FD            29               STA  IND1+1
                       30   *
0308: A9 00            31               LDA  #<INDADR2  ;$1000
030A: 85 FE            32               STA  IND2
030C: A9 10            33               LDA  #>INDADR2
030E: 85 FF            34               STA  IND2+1
                       35   *
                       36   * Accelerator-"ROM" lesefähig
                       37   * machen.
                       38   *
0310: A9 05            39               LDA  #5         ;ACCEL:5
0312: 8D 86 C0         40               STA  ACCEL
0315: AD 81 C0         41               LDA  $C081      ;RDROMBK2
0318: AD 81 C0         42               LDA  $C081      ;WRRAMBK2
                       43   *
                       44   * $D000-$FFFF nach $1000-$3FFF kopieren
                       45   *
031B: A0 00            46               LDY  #0
031D: B1 FC            47   LOOP1       LDA  (IND1),Y
031F: 91 FE            48               STA  (IND2),Y
0321: C8               49               INY
0322: D0 F9            50               BNE  LOOP1
0324: E6 FD            51               INC  IND1+1
0326: E6 FF            52               INC  IND2+1
0328: A5 FD            53               LDA  IND1+1
032A: D0 F1            54               BNE  LOOP1
                       55   *
032C: AD 81 C0         56               LDA  $C081      ;RDROMBK2
032F: AD 81 C0         57               LDA  $C081      ;WRRAMBK2
                       58   *
                       59   * Hier ggf. RTS, falls späteres
                       60   * "ROM" manuell vom Monitor aus
                       61   * gepatcht werden soll.
                       62   *
0332: 20 6F 03         63               JSR  PATCH
                       64   *
                       65   *---------Kopie nach oben---------
                       66   *
                       67   * Pointer initialisieren
                       68   *
0335: A9 00            69               LDA  #<INDADR2  ;$1000
0337: 85 FC            70               STA  IND1
0339: A9 10            71               LDA  #>INDADR2+1
033B: 85 FD            72               STA  IND1+1
                       73   *
033D: A9 00            74               LDA  #<INDADR1  ;$D000
033F: 85 FE            75               STA  IND2
0341: A9 D0            76               LDA  #>INDADR1+1
0343: 85 FF            77               STA  IND2+1
                       78   *
                       79   * "ROM" der Accelerator-Karte
                       80   * schreibfähig machen!
                       81   *
0345: A9 03            82               LDA  #3         ;ACCEL:3
0347: 8D 86 C0         83               STA  ACCEL
034A: AE 81 C0         84               LDX  $C081      ;RDROMBK2
034D: AE 81 C0         85               LDX  $C081      ;WRRAMBK2
                       86   *
                       87   * $1000-$3FFF nach $D000-$FFFF,
                       88   * d.h. in das Accelerator-"ROM"
                       89   * kopieren
                       90   *
0350: A0 00            91               LDY  #0
0352: B1 FC            92   LOOP2       LDA  (IND1),Y
0354: 91 FE            93               STA  (IND2),Y
0356: C8               94               INY
0357: D0 F9            95               BNE  LOOP2
0359: E6 FD            96               INC  IND1+1
035B: E6 FF            97               INC  IND2+1
035D: A5 FD            98               LDA  IND1+1
035F: C9 40            99               CMP  #>INDADR3
0361: D0 EF           100               BNE  LOOP2
                      101   *
                      102   * "ROM" der Accelerator wie quasi
                      103   * echtes ROM schreibunfähig machen
                      104   *
0363: A9 05           105               LDA  #5
0365: 8D 86 C0        106               STA  ACCEL
0368: AD 81 C0        107               LDA  $C081      ;RDROMBK2
036B: AD 81 C0        108               LDA  $C081      ;WRRAMBK2
                      109   *
036E: 60              110               RTS
                      111   *
                      112   * Gewünschte Patches hier ein-
                      113   * fügen. Es gilt:
                      114   * $1000-$3FFF = $D000-$FFFF
                      115   *
036F: EA              116   PATCH       NOP
                      117   *
                      118   * Beispiel für einen möglichen
                      119   * Patch von XAM: Bewirkt
                      120   * nach z.B. 1000.3FFF einen
                      121   * Hexdump mit 16 statt sonst
                      122   * 8 Bytes pro Zeile.
                      123   *
0370: A9 0F           124               LDA  #15        ;sonst 7
0372: 8D A6 3D        125               STA  $3DA6      ;$FDA6
0375: 8D B0 3D        126               STA  $3DB0      ;$FDB0
                      127   *
                      128   * Beispiel für einen weiteren
                      129   * Patch: Piepston-Änderung.
                      130   *
0378: A9 20           131               LDA  #$20       ;sonst $0C
037A: 8D E5 3B        132               STA  $3BE5      ;$FBE5
037D: 60              133               RTS
```

```
                        1               ORG  $300
                        2   *
                        3   * ACCEL.ROM.KOPIE2 (Echtes ROM)
                        4   * ========================
                        5   *
                        6   * Diese Routine kopiert zunächst
                        7   * den echten Apple-ROM-Inhalt
                        8   * $D000-$FFFF nach $1000-$3FFF
                        9   * und kopiert dann $1000-$3FFF
                       10   * in das Accelerator-"ROM",
                       11   * nachdem man (bei Bedarf) das
                       12   * spätere "ROM" entsprechend
                       13   * gepatcht hat.
                       14   *
                       15   INDADR1  EQU  $D000
                       16   INDADR2  EQU  $1000
                       17   INDADR3  EQU  $4000
                       18   IND1     EQU  $FC          ;-$FD
                       19   IND2     EQU  $FE          ;-$FF
                       20   ACCEL    EQU  $C086
                       21   *
                       22   *---------Kopie nach unten---------
                       23   *
                       24   * Pointer initialisieren
                       25   *
0300: A9 00            26               LDA  #<INDADR1  ;$D000
0302: 85 FC            27               STA  IND1
0304: A9 D0            28               LDA  #>INDADR1
0306: 85 FD            29               STA  IND1+1
                       30   *
0308: A9 00            31               LDA  #<INDADR2  ;$1000
030A: 85 FE            32               STA  IND2
030C: A9 10            33               LDA  #>INDADR2
030E: 85 FF            34               STA  IND2+1
                       35   *
```

```
                        36  * Echtes Apple-ROM lesefähig
                        37  * machen.
                        38  *
0310: A9 03             39          LDA  #3         ;ACCEL:3
0312: 8D 86 C0          40          STA  ACCEL
0315: AD 81 C0          41          LDA  $C081      ;RDROMBK2
0318: AD 81 C0          42          LDA  $C081      ;WRRAMBK2
                        43  *
                        44  * $D000-$FFFF nach $1000-$3FFF kopieren
                        45  *
031B: A0 00             46          LDY  #0
031D: B1 FC             47  LOOP1   LDA  (IND1),Y
031F: 91 FE             48          STA  (IND2),Y
0321: C8                49          INY
0322: D0 F9             50          BNE  LOOP1
0324: E6 FD             51          INC  IND1+1
0326: E6 FF             52          INC  IND2+1
0328: A5 FD             53          LDA  IND1+1
032A: D0 F1             54          BNE  LOOP1
                        55  *
032C: AD 81 C0          56          LDA  $C081      ;RDROMBK2
032F: AD 81 C0          57          LDA  $C081      ;WRRAMBK2
                        58  *
                        59  * Hier ggf. RTS, falls späteres
                        60  * "ROM" manuell vom Monitor aus
                        61  * gepatcht werden soll.
                        62  *
0332: 20 6F 03          63          JSR  PATCH
                        64  *
                        65  *--------Kopie nach oben----------
                        66  *
                        67  * Pointer initialisieren
                        68  *
0335: A9 00             69          LDA  #<INDADR2  ;$1000
0337: 85 FC             70          STA  IND1
0339: A9 10             71          LDA  #>INDADR2+1
033B: 85 FD             72          STA  IND1+1
                        73  *
033D: A9 00             74          LDA  #<INDADR1  ;$D000
033F: 85 FE             75          STA  IND2
0341: A9 D0             76          LDA  #>INDADR1+1
0343: 85 FF             77          STA  IND2+1
                        78  *
                        79  * "ROM" der Accelerator-Karte
                        80  * schreibfähig machen!
                        81  *
0345: A9 03             82          LDA  #3         ;ACCEL:3
0347: 8D 86 C0          83          STA  ACCEL
034A: AE 81 C0          84          LDX  $C081      ;RDROMBK2
034D: AE 81 C0          85          LDX  $C081      ;WRRAMBK2
                        86  *
                        87  * $1000-$3FFF nach $D000-$FFFF,
                        88  * d.h. in das Accelerator-"ROM"
                        89  * kopieren
                        90  *
0350: A0 00             91          LDY  #0
0352: B1 FC             92  LOOP2   LDA  (IND1),Y
0354: 91 FE             93          STA  (IND2),Y
0356: C8                94          INY
0357: D0 F9             95          BNE  LOOP2
0359: E6 FD             96          INC  IND1+1
035B: E6 FF             97          INC  IND2+1
035D: A5 FD             98          LDA  IND1+1
035F: C9 40             99          CMP  #>INDADR3
0361: D0 EF            100          BNE  LOOP2
                       101  *
                       102  * "ROM" der Accelerator wie quasi
                       103  * echtes ROM schreibunfähig machen
                       104  *
0363: A9 05            105          LDA  #5
0365: 8D 86 C0         106          STA  ACCEL
0368: AD 81 C0         107          LDA  $C081      ;RDROMBK2
036B: AD 81 C0         108          LDA  $C081      ;WRRAMBK2
                       109  *
036E: 60               110          RTS
                       111  *
                       112  * Gewünschte Patches hier ein-
                       113  * fügen. Es gilt:
                       114  * $1000-$3FFF = $D000-$FFFF
                       115  *
036F: EA               116  PATCH   NOP
                       117  *
                       118  * Patch-Routine nicht aktiv
                       119  *
0370: 60               120          RTS
```

Applesoft simuliert Turtle Graphic



Turtle Graphic ist ein in viele Hochsprachen, wie zum Beispiel in UCSD-Pascal oder im Apple Graforth][, integriertes Graphic-System. Es eignet sich besonders für den unbedarften Anwender, der damit ohne mathematischen Ballast an den Computer herangeführt wird. Eine der bekanntesten Anwendungen dürfte wohl die Programmiersprache Logo sein, die sich vor allem für Kinder eignet.

Harald Grumser

Das hier vorgestellte Programm stellt die wichtigsten Befehle der Turtle (Schildkröte) bereit. Nach dem Programmstart erscheint in der Mitte der HGR1 eine Turtle in Form eines Quadrates, die ihren Schwanz, der mit einem Schreibstift versehen ist, zum Boden gesenkt hat und in Richtung Bildschirmoberkante schaut. Im unteren Teil des Bildschirmes erscheint der Doppelpunkt als Promptzeichen; die Turtle erwartet einen Befehl.
Gibt man nun zum Beispiel „50 MOVE" ein, so bewegt sich die Turtle um 50 Bildpunkte nach oben. „90 TURN 50 MOVE" dreht die Turtle um 90 Grad nach rechts und bewegt sie wieder um 50 Bildpunkte. Wiederholt man dies noch zweimal, so hat sie ein Quadrat gezeichnet.
Die Befehle, die die Turtle ausführen kann, lauten wie folgt:

MOVE bewegt die Turtle um die angegebene Anzahl von Punkten in die Richtung, in der sie gerade steht.

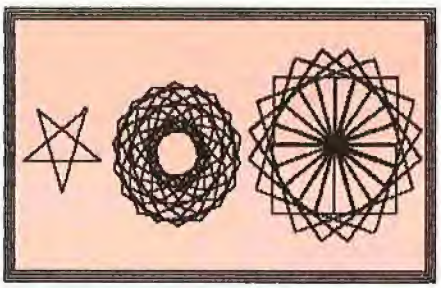
TURN dreht die Turtle um die angegebene Gradzahl, wobei positive Zahlen nach rechts drehen und negative nach links.
MOVETO bewegt die Turtle zu den angegebenen Koordinaten, wie sie bei HPLOT benutzt werden (279 · 159 Punkte).
TURNTO dreht die Turtle in die angegebene Richtung, („-90 TURNTO" dreht die Turtle in Richtung des linken Bildschirmrandes.)
TURTLE löscht den Bildschirm und positioniert die Turtle wie beim Programmstart.
PENUP hebt den Schreibstift, so daß bei der nächsten Bewegung kein Strich gezogen wird.
PENDOWN senkt den Schreibstift wieder.

Der Leser wird sich vielleicht fragen, warum die Parameter vor dem Befehl stehen. Tippt man eine Zahl ein, so wird diese auf dem sogenannten Stack (Stapel) abgelegt. Braucht die Turtle einen Parameter, so holt sie sich diesen vom Stack. Ein Quadrat läßt sich also auch durch Eingabe von „50 90 50 90 50 90 50 MOVE TURN MOVE TURN MOVE TURN MOVE" zeichnen. Das gleiche gilt auch für die Arithmetik; die Turtle kennt die vier Grundrechenarten. Zuerst werden die Operanden auf dem Stack abgelegt und dann der Operator darauf angewandt. So ergibt „20 30 + MOVE" eine Bewegung um 50 Schritte. Dieses Verfahren nennt sich Reverse Polish Notation (Umgekehrte polnische Notation) oder auch Postfix-Notation. Um dies noch einmal an einem Beispiel zu erläutern, sei der Ausdruck (1+2) * (3+4) zu berechnen: Das Ergebnis würde lauten „ 1 2 + 3 4 + * ". Die Turtle stellt nun einige Befehle zur Verfügung, um den Stack zu manipulieren:

+ − * / holen zwei Zahlen vom Stack, verknüpfen sie mit dem jeweiligen Operator und legen das Ergebnis wieder auf dem Stack ab.
PULL holt eine Zahl vom Stack, ohne sie zu benutzen.
DUP dupliziert die oberste Zahl auf dem Stack. „DUP *" wäre somit die Operation zum Potenzieren.
SWAP vertauscht die oberen beiden Zahlen auf dem Stack. „3 10 SWAP -" ergibt also 7, während „3 10 -" -7 ergibt.
PICK holt sich einen Index vom Stack und kopiert dann die von oben indizierte Zahl auf den Stack. Ein Beispiel macht dies wohl deutlicher: „10 20 30 2 Pick" ergibt auf dem Stack „10 20 30 20", „1 PICK" enspricht also dem DUP-Befehl.

Bei dem Quadrat oben würde es sich als praktisch erweisen, wenn mehrmals zu wiederholende Befehle als Schleife (Loop) eingegeben werden könnten. Selbstverständlich ist die Turtle auch dazu in der Lage. Die zu wiederholenden Befehle werden zwischen „DO" und „LOOP" eingeschlossen. Als Schleifenzähler holt sich „DO" eine Zahl vom Stack. „4 DO 50 MOVE 90 TURN LOOP" würde also unser Quadrat zeichnen.

Unsere Turtle kann jedoch noch mehr, nämlich neue Befehle lernen. Eine zu lernende Befehlsfolge wird durch „LEARN>" und „<" eingeschlossen, wobei nach „LEARN>" der Name des neuen Befehls folgt. Nach der Eingabe von „LEARN> QUADRAT 4 DO 50 MOVE 90 TURN LOOP <" zeichnet die Turtle jedesmal ein Quadrat, wenn der Befehl „QUADRAT" eingegeben wird. Der linke Stern in der **Grafik 1** wird somit gezeichnet durch die Befehle „20 DO QUADRAT 18 TURN LOOP"



Grafik 1

Es sei nicht verschwiegen, wie der Fünfzack in der abgebildeten Grafik erzeugt wird. „LEARN> FUENFZACK 216 TURNTO PENUP DUP 2 / MOVE PENDOWN 18 TURNTO 5 DO DUP MOVE 144 TURN LOOP <" zeichnet einen Fünfzack um die Stelle, an der sich die Turtle gerade befindet, in der angegebenen Größe.
Dem Leser sei es überlassen auszutüfteln, wie die Figur in der Bildmitte entsteht. Ein Tip: Es handelt sich um eine Menge verdrehter Siebenecke.

Abschließend sollen noch einige Befehle erläutert werden, die das Arbeiten mit der Turtle erleichtern:

TEXT schaltet auf nur Text um, so daß der ganze Bildschirm zur Eingabe von Befehlen zur Verfügung steht.
GRAPH schaltet zurück auf HGR1, ohne den Bildschirm zu löschen.
LIST listet alle Befehlswörter samt der neu erlernten. Nach 22 Wörtern geht es erst nach Tastendruck weiter.
FORGET vergißt alle Befehle ab dem spezifizierten. Da neue Befehle bereits neu erlernte beinhalten können, geht die Turtle davon aus, daß nach Löschen des Befehls „QUADRAT" auch der Befehl „FUENFZACK" nicht mehr ausgeführt werden kann.
STACK listet den gesamten Stack, wobei auch hier nach 22 Zahlen auf einen Tastendruck gewartet wird. Der Stack kann insgesamt 101 Zahlen beinhalten.
BYE verabschiedet sich und springt zurück in den Applesoft-Direktmodus.

Zur Eingabe sei folgendes bemerkt: In einer Eingabezeile können bis zu 50 Befehle stehen, die von mindestens einem Blank (Leerzeichen) voneinander getrennt sein müssen. Außer den Namen nach „LEARN>" und „FORGET" können sie aber auch einzeln eingegeben werden. Tritt während des Lernens ein Fehler auf, so wird alles ignoriert. Dasselbe gilt, wenn mehr „DO" als „LOOP" Befehle oder umgekehrt im Lernmodus auftauchen. Schleifen werden dadurch realisiert, daß deren Inhalt vorübergehend gelernt wird. Daher gilt für sie dasselbe wie oben. Während des Lernens führt die Turtle die Befehle nicht aus. Sollte sie also einmal wider Erwarten nicht reagieren, so kann das daran liegen, daß ein „<" oder „LOOP" fehlt.
„<" schließt im Direktmodus alle Schleifen, ohne sie auszuführen, meldet sich aber mit „MISSING DO". Die Standardbefehle können nicht mit „FORGET" gelöscht werden, und ein bereits erlernter Befehl kann nicht nochmals gelernt werden.
Die Fehlermeldungen sind im großen und ganzen selbsterklärend. „OUT OF MEMORY" tritt auf, wenn die Anzahl der erlernten Befehle plus der der aktiven Schleifen 21 überschreitet.
Während die Turtle ein Programm abarbeitet, kann sie durch Ctrl-S gestoppt werden.

Die Turtle kann sich nicht über den Bildschirmrand hinaus bewegen, sondern schneidet die Koordinaten ab, was bei größeren Figuren zu Verzerrungen führt.

Das Programm selbst ist nicht im Hinblick auf maximale Geschwindigkeit, sondern auf Lesbarkeit hin geschrieben worden.

Der einzige Beitrag zur Erhöhung der Geschwindigkeit ist die Verlegung der Initialisierungsroutine an den Schluß (Zeile 9000-9995) und die Nennung der Variablen in der Reihenfolge ihrer Häufigkeit.
In Zeile 20-60 wird auf die Eingabezeile gewartet und daraus in der Routine 8000-8999 ein einzelner Befehl abgespalten und dessen Index (Variable K) gesucht. Danach wird in der Routine 70-99 in das entsprechende Unterprogramm verzweigt. Hier wird auch das Learnflag gesetzt und mit Hilfe von Epsilon die Schleifentiefe gezählt. Ist der Befehl selbst eine Zahl, so wird K um OFFSET erhöht, somit enthält K die Information über die Zahl.
Die Routinen in 100-999 sind die verschiedenen Unterprogramme für die Turtle, die in 1000-1999 die für den Stack.
Bevor auf die LOOP- und LEARN>-teile eingegangen wird, ist es sinnvoll, die Abspeicherung der neuen Befehle zu erläutern. Für die neu zu lernenden Befehle gibt es eine Matrix L, die aus 21 Zeilen zu je 50 Spalten besteht. Ein Befehl besetzt nun eine Zeile derart, daß dessen Indizes in dieser Zeile nacheinander abgelegt werden, gefolgt von einer Null.
Wird nun einer dieser Befehle aufgerufen, so verzweigt das Programm nach 7000, wo diese einzelnen K-Werte ausgelesen und dem Verteiler ab Zeile 70 zugeführt werden. Da es möglich ist, in einem neuen Befehl einen ebenfalls neuen Befehl aufzurufen, kann dieses Unterprogramm sich selbst neu aufrufen. Damit bei dieser Rekursion keine Zeiger verlorengehen, müssen diese in Form von Vektoren L1,L2 und deren Pointer NLP aufbewahrt werden. Dasselbe gilt für die Routine in 2500-2999, die dieselbe Rekursion für geschachtelte Schleifen darstellt, wobei diese über einen anderen Pointer LLP verwaltet werden.
Bleibt noch der Schleifenstart in 2000-2499, der einsichtig ist, und die Lernroutine in 3000-3999: Dort wird der Name abgefragt, abgespeichert und die Pointer zur Abspeicherung in der L-Matrix gesetzt, beziehungsweise eine Null ans Ende der Zeile geschrieben und die Anzahl der Befehle IT um eins erhöht.
In der Fehlerbehandlungsroutine in 6000-6999 ist Zeile 6510 zu erläutern. Diese Interpreter-Routine initialisiert den Returnstack. Dies ist erforderlich, da sonst nach einem Programmabsturz der Returnstack noch belastet wäre und die für die Rekursionen benötigte Schachtelungstiefe nicht mehr erreicht würde. (Der Returnstack kann nur 25 Returnadressen aufnehmen). Es ist in diesem Zusammenhang auch davon abzuraten, die Variable NL zu vergrößern, da in nicht mehr wie 25 Unterprogrammebenen gesprungen werden kann.

Zuletzt noch ein paar Tips für diejenigen, die das Programm durch eigene Befehle erweitern wollen. Zuerst schreibe man seine Erweiterung in einen freien Bereich. Danach gebe man dieser einen Namen und füge diesen in Zeile 9220 an. Sodann erhöhe man IS in Zeile 9110 um eins und erweitere die Zeilenliste in Zeile 90 um seine neue Routine.
Es ist davon abzuraten, seinen neuen Befehl in der Mitte der vorhandenen Befehle anzuordnen, da dann die Abfragen in Zeile 70-99 nicht mehr stimmen.

```
  0 REM  ********************
  1 REM     TURTLE GRAPHIK
  2 REM  ********************
  3 REM
  4 LOMEM: 4 * 16 ↑ 3
  5 REM  Hinter HGR
 10 GOSUB 9000
 15 GOSUB 100
 19 :
 20 REM  Hole Eingabezeile und arbeite sie ab
 25 PRIGHT = 0
 30 INPUT ":";LINE$
 35 GOSUB 8000
 40 IF K = 0 THEN 25
 45 IF K = IT + 2 THEN ERFLG = 1: GOTO 6000
 50 GOSUB 70
 55 GOTO 35
 60 :
 70 REM  Verteiler
 72 IF  PEEK (KBD) = CS THEN 6000
 74 IF K = 22 THEN LRNFLG = LRNFLG - EPS: REM  IN$ = "LOOP"
 76 IF LRNFLG = 0 OR K = 24 THEN 88: REM  Direkt Modus oder
    IN$ = "<"
 78 IF LRNFLG < 0 THEN ERFLG = 5: GOTO 6000
 80 IF K = 21 THEN LRNFLG = LRNFLG + EPS: REM  IN$="DO"
 82 L(LLP,L2(LLP)) = K:L2(LLP) = L2(LLP) + 1: IF L2(LLP) <  =
    INS THEN 86
 84 ERFLG = 8: GOTO 6000
 86 RETURN
 88 IF K > IS + NL + 2 THEN 1100
 90 ON K GOTO
    100,200,300,400,500,600,650,700,750,800,900,1200,1300,
    1400,1500,1600,1600,1600,1600,1900,2000,2500,3000,3500,
    9996
 92 GOTO 7000: REM   K > IS + 1 -> Ausfuehren eines
    gelernten Befehls
 99 :
100 REM  TURTLE
110 HOME : HGR : VTAB 21
120 DIR = 0:PX = MX / 2:PY = MY / 2
130 HC = 1: HCOLOR= 3
140 :
150 REM  DRAW TURTLE
160 XDRAW 1 AT PX,PY
170 IF  NOT HC THEN 190
180 HPLOT PX,PY
190 RETURN
199 :
200 REM  MOVE
205 GOSUB 1000
210 GOSUB 150
215 PX = PX +  SIN (DIR / 180 * PI) * ST
220 PY = PY -  COS (DIR / 180 * PI) * ST
225 IF PX > MX THEN PX = MX
230 IF PX < 0 THEN PX = 0
235 IF PY > MY THEN PY = MY
240 IF PY < 0 THEN PY = 0
245 IF  NOT HC THEN 255
250 HPLOT  TO PX,PY
255 GOTO 150
299 :
300 REM  TURN
310 GOSUB 1000
320 DIR = DIR + ST
330 DIR = 360 * (DIR / 360 -  INT (DIR / 360))
340 RETURN
399 :
400 REM  MOVETO
410 GOSUB 1000
420 A = ST
430 GOSUB 1000
440 GOSUB 150: REM  Kein XDRAW der Turtle vor Pruefung des
    Stacks
450 PY = A:PX = ST
460 GOTO 225
499 :
500 REM  TURNTO
510 GOSUB 1000
520 DIR = ST
530 GOTO 330
599 :
600 REM  PENUP
610 HC = 0
620 RETURN
649 :
650 REM  PENDOWN
660 HC = 1
```

```
670 RETURN
699 :
700 REM  TEXT
710 TEXT : HOME : INVERSE
720 PRINT  SPC( 13);"TURTLE GRAPHIK"; SPC( 13)
730 NORMAL : POKE 34,1: REM  Textfenster: WINDOWTOP = 1
740 RETURN
749 :
750 REM  GRAPH
760 POKE  - 16304,0: REM  HGR Schalter; andere Schalter
    durch TURTLE gesetzt
770 HOME
780 VTAB 21
790 RETURN
799 :
800 REM  LIST
810 FOR I = 0 TO IT
820 PRINT "- ";I$(I)
830 IF (I + 1) / 22 <  >  INT ((I + 1) / 22) THEN 850
840 GET A$: IF  ASC (A$) = CS - 128 THEN I = IT
850 NEXT
860 RETURN
899 :
900 REM  FORGET
910 IF LRNFLG <  > 0 THEN 940
920 GOSUB 8000
930 IF K > IS + 1 AND K <  = IT + 1 THEN 950
940 ERFLG = 9: GOTO 6000
950 IT = K - 2
960 LLP = IT - IS - 1
970 RETURN
999 :
1000 REM  Hole vom  Stack
1010 IF SP < 0 THEN ERFLG = 2: GOTO 6000
1030 ST = ST(SP)
1040 SP = SP - 1
1050 RETURN
1099 :
1100 REM  Schiebe auf den Stack
1110 IF SP = 100 THEN ERFLG = 3: GOTO 6000
1120 SP = SP + 1
1130 ST(SP) = K - OFFSET
1140 RETURN
1199 :
1200 REM  PULL  holt ein "Dummy"
1210 SP = SP - 1
1220 IF SP <  - 1 THEN SP =  - 1
1230 RETURN
1299 :
1300 REM  DUPliziere
1310 IF SP < 0 THEN 1010
1320 K = ST(SP) + OFFSET
1330 GOTO 1110
1399 :
1400 REM  SWAP vertauscht die oberen beiden  Stackelemente
1410 A = ST(SP)
1420 ST(SP) = ST(SP - 1)
1430 ST(SP - 1) = A
1440 RETURN
1499 :
1500 REM  PICK holt das von oben indizierte Stackelement
1510 GOSUB 1000
1520 I = SP - ST + 1
1530 IF SP < 0 THEN 1010
1540 IF I > SP THEN 1580
1550 IF I < 0 THEN I = 0
1560 SP = SP + 1
1570 ST(SP) = ST(I)
1580 RETURN
1599 :
1600 REM  STACK + - * /
1610 GOSUB 1000
1620 ON K - 15 GOTO 1630,1640,1650,1660
1630 ST(SP) = ST(SP) + ST: RETURN
1640 ST(SP) = ST(SP) - ST: RETURN
1650 ST(SP) = ST(SP) * ST: RETURN
1660 IF ST = 0 THEN ERFLG = 4: GOTO 6000
1670 ST(SP) = ST(SP) / ST: RETURN
1699 :
1900 REM  STACK
1905 I =  - 1
1910 I = I + 1: IF I > SP THEN 1960
1920 PRINT  CHR$ (91);" ";ST(I);" "; CHR$ (93)
1930 IF (I + 1) / 22 <  >  INT ((I + 1) / 22) THEN 1950
1940 GET A$: IF  ASC (A$) = CS - 128 THEN I = SP
1950 GOTO 1910
1960 RETURN
1999 :
```

```
2000  REM  DO-Schleifenstart
2010 LRNFLG = LRNFLG + EPS
2020 LLP = LLP + 1
2030  IF LLP > NL THEN ERFLG = 7: GOTO 6000
2040 L2(LLP) = 0
2050  RETURN
2499 :
2500  REM  Schleifenende (LOOP)
2510 L(LLP,L2(LLP)) = 0
2520  GOSUB 1000:COUNT(LLP) = ST
2530  IF COUNT(LLP) <  = 0 THEN 2560
2540 K = IS + LLP + 2: GOSUB 7000
2550 COUNT(LLP) = COUNT(LLP) - 1: GOTO 2530
2560 LLP = LLP - 1
2570  RETURN
2999 :
3000  REM  LEARN
3010  IF LRNFLG <  > 0 THEN 3040
3020  GOSUB 8000
3030  IF K = IT + 2 THEN 3050
3040 ERFLG = 1: GOTO 6000
3050 LRNFLG = 1
3060  IF IT - IS > NL THEN ERFLG = 7: GOTO 6000
3070 I$(IT + 1) = IN$
3080 LLP = LLP + 1:L2(LLP) = 0
3090  RETURN
3499 :
3500  REM  "<" Ende des Lernmodus
3510  IF LRNFLG = 0 THEN 6200
3520 LRNFLG = LRNFLG - 1
3530  IF LRNFLG < 0 THEN ERFLG = 5: GOTO 6000
3540  IF LRNFLG > 0 THEN ERFLG = 6: GOTO 6000
3550 IT = IT + 1
3560 L(LLP,L2(LLP)) = 0
3570  RETURN
3999 :
6000  REM  Fehlerbehandlungsroutine
6010  POKE 49168,0
6020  REM  Tastaturruecksetzimpuls
6030  PRINT G$;
6040  ON ERFLG GOTO
      6060,6070,6080,6090,6100,6110,6120,6130,6140
6050  GOTO 6200
6060  PRINT "SYNTAX ERROR": GOTO 6200
6070  PRINT "STACK EMPTY": GOTO 6200
6080  PRINT "STACK FULL": GOTO 6200
6090  PRINT "DIVISION BY ZERO": GOTO 6200
6100  PRINT "MISSING DO": GOTO 6200
6110  PRINT "MISSING LOOP": GOTO 6200
6120  PRINT "OUT OF MEMORY": GOTO 6200
6130  PRINT "TO MANY INSTRUCTIONS": GOTO 6200
6140  PRINT "CAN'T FORGET"
6200 ERFLG = 0:LRNFLG = 0
6500 NIP =  - 1:LLP = IT - IS - 1
6510  CALL  - 10621
6520  REM  Initialisiere Basic-Return-Stack
6530  GOTO 25
6999 :
7000  REM  Fuehre einen gelernten Befehl aus
7010 NIP = NIP + 1
7020  IF K > IS + NL + 2 THEN K = K - OFFSET
7030 L1(NIP) = K - 2 - IS:L2(NIP) = 0
7040 K = L(L1(NIP),L2(NIP)):L2(NIP) = L2(NIP) + 1
7050  IF K = 0 THEN NIP = NIP - 1: GOTO 7070
7060  GOSUB 70: GOTO 7040
7070  RETURN
7999 :
8000  REM  Naechster Instruktions-Index
8010 LINE$ = LINE$ + " "
8020 PRIGHT = PRIGHT + 1: IF PRIGHT >  LEN (LINE$) THEN K =
      0: RETURN
8030  IF  MID$ (LINE$,PRIGHT,1) = " " THEN 8020
8040 PLEFT = PRIGHT
8050 PRIGHT = PRIGHT + 1: IF  MID$ (LINE$,PRIGHT,1) <  > " "
      THEN 8050
8060 IN$ =  MID$ (LINE$,PLEFT,PRIGHT - PLEFT)
8070  IF  VAL (IN$) = 0 AND IN$ <  > "0" THEN 8100
8080 K = OFFSET +  VAL (IN$): RETURN
8100 K = IT + 1
8110  FOR I = 0 TO IT
8120  IF I$(I) <  > IN$ THEN 8140
8130 K = I:I = IT
8140  NEXT :K = K + 1
8150  RETURN
8999 :
9000  REM  Initialisierung
```

```
9010  FOR I = 768 TO 768 + 9
9020  READ A: POKE I,A: NEXT
9030  DATA  1,0,4,0,40,54,63,36,4,0
9040  POKE 232,0: POKE 233,3
9050  ROT= 0: SCALE= 1
9060  REM  Formtabelle fuer die Turtle
9070  DIM ST(100): REM  101 Stackelemente
9080 SP =  - 1:ST = 0: REM  Stapelzeiger (Stackpointer) und
      angefordertes Stackelement
9090 NL = 20: REM  Anzahl der zu lernenden Befehle
9100 INS = 50: REM  Anzahl der Instruktiuonen pro Befehl
9110 IS = 24: REM  Anzahl der Standartbefehle
9120 IT = IS: REM  Anzahl der Befehle insgesamt
9130  DIM I$(IS + NL + 1): REM  Feld mit den Befehlsnamen
9140  FOR I = 0 TO IS
9150  READ I$(I): NEXT
9160  DATA  TURTLE,MOVE,TURN,MOVETO
9170  DATA  TURNTO,PENUP,PENDOWN
9180  DATA  TEXT,GRAPH,LIST,FORGET
9190  DATA  PULL,DUP,SWAP,PICK
9200  DATA  +,-,*,/,STACK
9210  DATA  DO,LOOP,LEARN>,<
9220  DATA  BYE
9230  DIM L(NL,INS): REM  Matrix mit NL Befehlen zu je INS
      Instruktionen
9240  DIM L1(NL),L2(NL): REM  Zeilen- und Spaltenindex der
      "L"-Matrix
9250  DIM COUNT(NL): REM  Zaehler fuer geschachtelte
      Schleifen
9260 K = 0: REM  Instruktionsindex
9270 KBD = 49152: REM  PEEK zur Tastaturabfrage
9280 CS = 147: REM  CS = ASC (CTRL-S)
9290 OFFSET = 2 ↑ 15: REM  K wird um OFFSET erhoeht um
      Zahlen als Instruktionen abzuspeichern
9300 LLP =  - 1: REM  Zaehler fuer LEARN> und aktive
      Schleifen
9310 NIP = LLP: REM  Zaehler fuer geschachtelte Befehle
9320 LRNFLG = 0: REM  Wird waehrend des Lernmodus auf 1
      gesetzt
9330 ERRFLG = 0: REM  Fehlercode fuer Fehlerbehandlung
9340 EPS = 1 / 64: REM  LRNFLG wird bei jedem DO um EPS
      erhoeht und bei jedem LOOP erniedrigt
9350 PI =  ATN (1) * 4
9360 PX = 0:PY = 0: REM  Plot-Koordinaten
9370 HC = 0: REM  Logische Variable fuer PENUP/PENDOWN
9380 DIR = 0: REM  Richtung der Turtle in Grad
9390 MX = 279:MY = 159: REM  HGR Fenster
9400 PLEFT = 0: REM  Linker Zeiger der Eingabezeile
9410 PRIGHT = 0: REM  Rechter Zeiger der Eingabezeile
9420 A = 0:I = 0: REM  Allgemeine Variablen
9430 G$ =  CHR$ (135): REM  Bell
9440  REM  LINE$,IN$ sind die Eingabezeile und die daraus
      abgespaltenen Befehle
9450  RETURN
9995 :
9996  REM  Ende des Programms
9997  TEXT : HOME
9998  PRINT "END OF TURTLE GRAPHIC"
9999  END
10000 :
10001  REM  HINWEIS:
10002  REM  Wer dieses Programm mit allen REM's abtippt
10003  REM  und es dann ausfuehrt wird sich wundern.
10004  REM  Programme, die mehr als 25 Diskettensektoren
      lang sind
10005  REM  und die HGR aufrufen, loeschen sich zum Teil
      selbst.
```

Peeker-Sammeldiskette

Die abgedruckten größeren Programme sind auch als Sammeldisketten erhältlich, die etwa jeden zweiten Monat erscheinen und die Listings der vorangehenden zwei Hefte zusammenfassen.

APPLE
MASCHINEN SPRACHE

Der Einstieg über BASIC!

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

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

te·wi

Weiterführende Literatur...

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

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

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

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

6502 –
Programmieren in Assembler
(L. Leventhal)
Dieser Titel aus einer ganzen Reihe von Büchern über die Assemblersprachen-Programmierung befaßt sich ausführlich mit dem weitverbreiteten Mikroprozessor 6502.
704 Seiten, Softcover, DM 59,–

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

tm 4045

Noch im Programm:
VisiCalc, 50 Programme auf Diskette, DM 79,– ● 77 BASIC-Programme, A4, DM 39,– ● CP/M und Wordstar, DM 29,80.

Wie man die Grafik verdoppelt

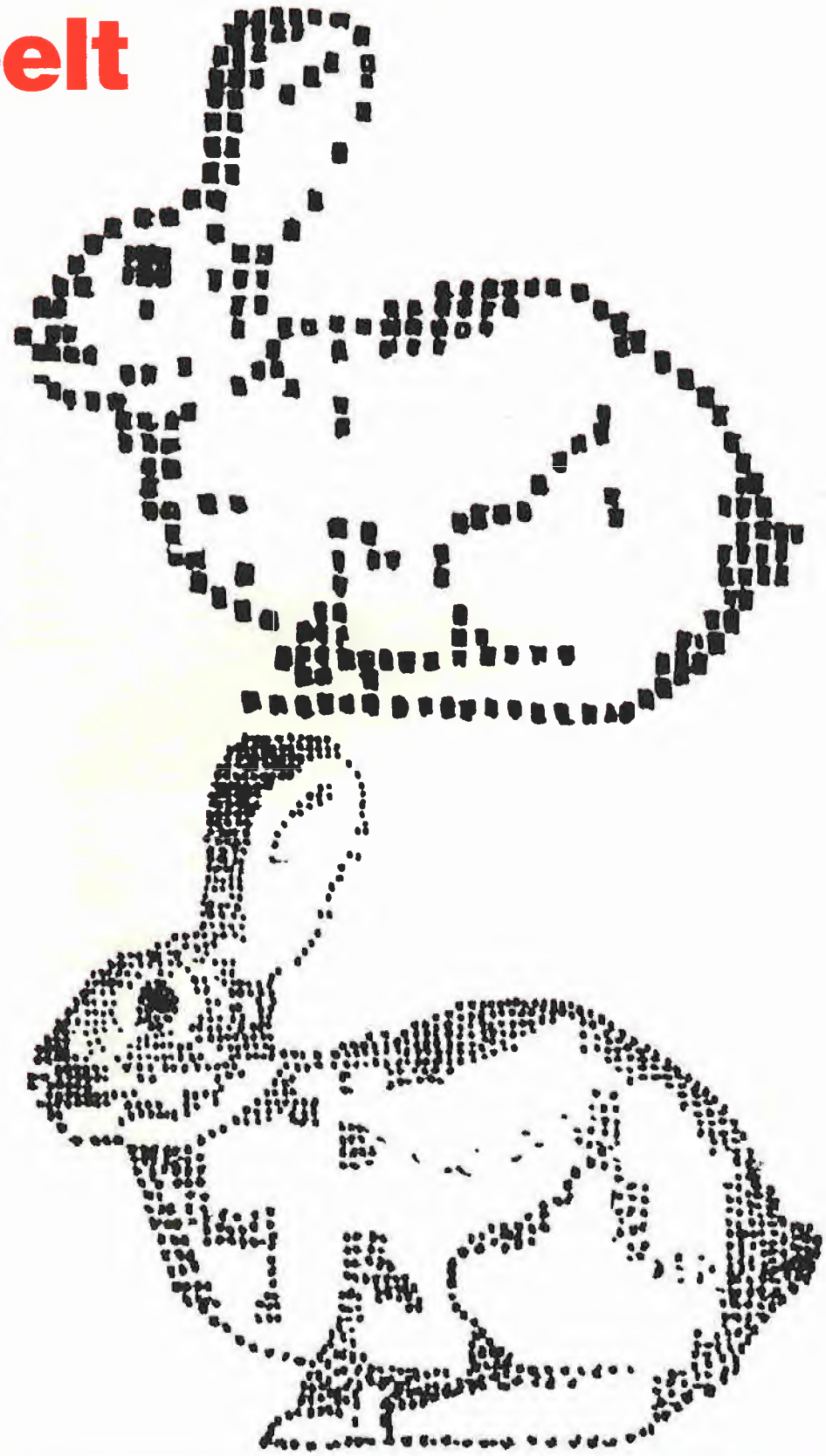
Teil 1: Double Lores

Die erweiterte 80-Zeichenkarte des Apple IIe = 64K-Karte läßt sich auf vielfältige Weise einsetzen, z.B. als

– 80-Zeichen-Bildschirm-Teilspeicher
– RAM-Diskspeicher
– Teilspeicher für doppelte hochauflösende Grafik (Double Hires: 560 mal 192 statt 280 mal 192 Bildpunkte)
– Teilspeicher für doppelte niedrigauflösende Grafik (Double Lores: 80 mal 48 statt 40 mal 48 Kästen).

Zur Double Hires werden von Apple in den technischen Unterlagen nur vage Angaben gemacht, und die Möglichkeit von Double Lores wird überhaupt nicht angedeutet. Im ersten Teil dieser Folge wird auf die Double Lores eingegangen, während sich der zweite Teil mit der Double Hires befaßt.

Die Double Lores wird meines Wissens hier erstmals geschildert. Mangels technischer Informationen war es nicht einfach herauszufinden, ob sie überhaupt und, wenn ja, wie möglich ist, doch zeigte sich im nachhinein, daß Double Lores nicht nur verblüffend einfach ist, sondern auch in einem reinen Applesoft-Programm benutzt werden kann, da hier keine Geschwindigkeitsprobleme auftreten. (Umgekehrt muß die Double Hires in Assembler geschrieben werden, da sonst die HPLOT-Funktion viel zu langsam ist.)



Ulrich Stiehl

1. Normale Lores-Grafik

Bevor wir auf die Double Lores eingehen, seien die normalen Lores-Grafik-Befehle kurz wiederholt. Der Lores-Grafikbildschirm ist speichermäßig mit dem 40-Zeichen-Textbildschirm identisch. Während der 40-Zeichen-Textbildschirm jedoch 24 Zeilen zu je 40 Zeichen umfaßt, kann bei dem Lores-Grafikbildschirm genau die doppelte Menge an Grafik-Kästchen geplottet werden. Dies rührt daher, daß in jeder Bildschirmspeicherstelle anstelle *eines* Buchstabens je *zwei* Kästchen untergebracht werden. Der Bildschirm nimmt den Speicherbereich $0400-$07FF ein ($0400 = dezimal 1024). Jedes Bytes dieses Bereichs – mit Ausnahme sog. versteckter Bildschirmspeicherstellen – kann mithin zwei Kästchen aufnehmen. Es gilt also:

Textbildschirm: 24 Zeilen zu je 40 Zeichen
Loresbildschirm: 48 Zeilen zu je 40 Kästchen
Geteilter Loresbildschirm: 40 Zeilen zu je 40 Kästchen plus 4 Textzeilen am unteren Bildschirmrand.

Experiment 1

Zu Testzwecken schalten wir mit dem Befehl
GR
auf Lores-Grafik um und gehen dann mit
CALL -151
in den Monitor. Nunmehr geben wir
400:0F
ein und sehen, daß ganz links oben (= Zeile 0, Spalte 0) ein Kästchen geplottet wurde. Geben wir jetzt
400:F0
ein, wird ein Kästchen in derselben Spalte, aber statt in der ersten in der zweiten Zeile geplottet (= Zeile 1, Spalte 0). Geben wir abschließend
400:FF
ein, dann wird sowohl das obere wie das untere Kästchen sichtbar. $FF ist 1 Byte und besteht aus 2 Halbbytes $F und $F. Da jedes Halbbyte einen Wert von $0-$F oder dezimal 0 bis 15 einnehmen kann, gibt es theoretisch 16 Farben von 0 (= schwarz) bis 15 (= weiß), die allerdings auf einem grünen Monitor nur als unterschiedliche Weiß-Grün-Abstufungen sichtbar werden. Wer über einen Farbmonitor verfügt, kann mit dem Befehl
COLOR = F
wobei F als Variablenwert im Bereich 0-15 liegt, die gewünschte Farbe definieren, die dann für alle nachfolgend geplotteten Kästchen gilt.

Lores-Befehle

Soviel zur Theorie des Lores-Bildschirmspeichers. Um die Plot-Befehle zu verstehen, arbeite man am besten mit einem Koordinatensystem in der Art des abgebildeten **Diagramms 1**. In Abweichung vom „normalen" Koordinatensystem ist der Origo = Ursprung links oben statt links unten. Die X-Achse oder Abszisse kann Werte im Bereich 0-39 und die Y-Achse oder Ordinate Werte im Bereich 0-39 (bei geteiltem Bildschirm) oder 0-47 (bei vollem Bildschirm) einnehmen. Eine waagrechte Linie von X1 bis X2 heißt Zeile, und eine senkrechte Linie von Y1 bis Y2 heißt Spalte.

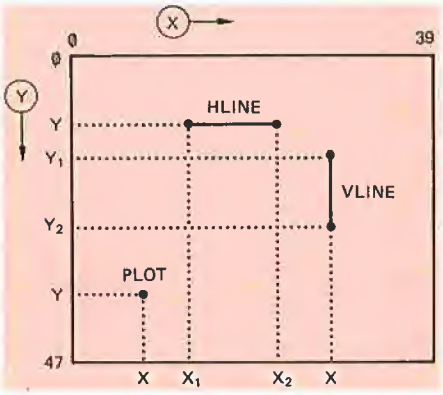


Diagramm 1

GR

schaltet auf Lores-Grafik mit geteiltem Bildschirm und Farbe = 0 = schwarz um. Deshalb muß man danach mit

COLOR = F, z.B. COLOR = 15

die gewünschte Farbe festlegen.

POKE -16302, 0

schaltet vom geteilten zum vollen Bildschirm um. Danach kann man bei Bedarf mit

CALL -1998

die unteren Zeilen des Lores-Bildschirms löschen ($F832 = -1998 = CLRSCR).

TEXT

hebt den Grafik-Modus wieder auf.

PLOT X, Y, z.B. PLOT 39, 39

plottet ein Kästchen im Schnittpunkt von horizontaler X- und senkrechter Y-Achse.

HLIN X1, X2 AT Y, z.B. HLIN 0, 39 AT 10

plottet einen Balken (= Kästchen-Zeile) vom linken X1-Wert bis zum rechten X2-Wert in der Zeile Y.

VLIN Y1, Y2 AT X, z.B. VLIN 0, 47 AT 20

plottet einen Balken (= Kästchen-Spalte) vom oberen Y1-Wert bis zum unteren Y2-Wert in der Spalte X.

F = SCRN (X, Y), z.B. F = SCRN (10, 20)

ermittelt die Farbe oder Farbnummer (0-15) im Schnittpunkt von X und Y.

2. 80-Zeichen-Textbildschirm

Wie ist nunmehr doppelte Lores-Grafik möglich? Wir erinnern uns, daß der Lores-Grafikbildschirm speichermäßig mit dem 40-Zeichen-Textbildschirm identisch ist. Schaltet man den Apple IIe mit
PR#3
auf 80-Zeichendarstellung, dann umfaßt der 80-Zeichen-Textbildschirm 24 Zeilen zu je 80 Zeichen. Der Bildschirmspeicher selbst ist dann in 2 Hälften unterteilt, nämlich in

a) den unteren Bereich $0400-$07FF auf dem 64K-Motherboard, und
b) den oberen Bereich $0400-$07FF auf der 64K-Karte

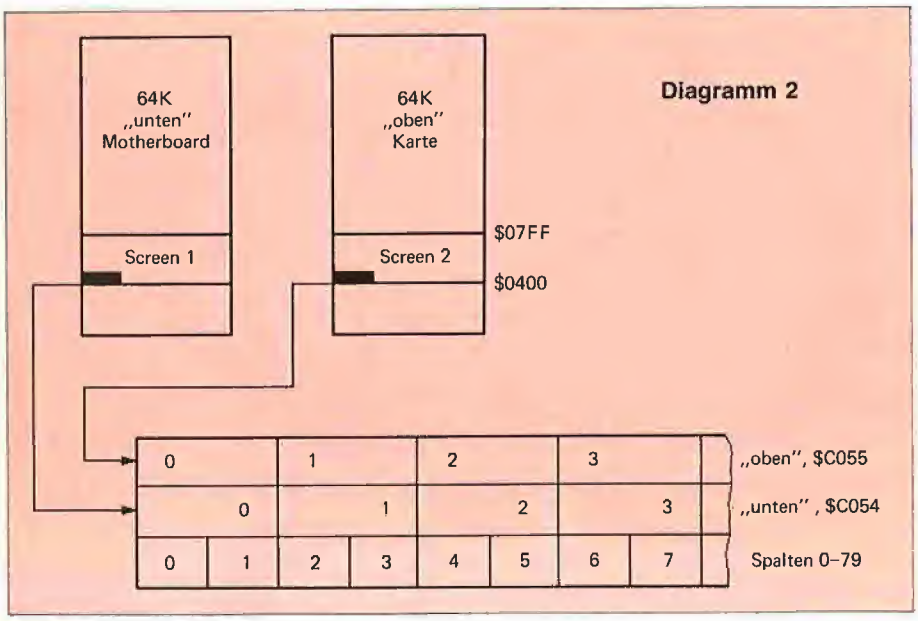
Die 80 Spalten oder Stellen einer 80 Zeichen umfassenden Zeile werden von 0-79 numeriert, wobei zwischen den ungeraden und geraden Spalten unterschieden wird:

1, 3, 5, ... 77, 79 ungerade = oben
0, 2, 4, ... 76, 78 gerade = unten

Damit gibt es „oben" 40 ungerade und „unten" 40 gerade Spalten. Die 80-Zeichen-Bildschirmroutinen funktionieren im Prinzip wie die entsprechenden 40-Zeichen-Routinen:

a) Zunächst wird ermittelt, ob die Spaltenzahl im Bereich 0-79 eine gerade oder eine ungerade Zahl ist.
b) Dann wird durch einen Softswitch entweder die „obere" Hälfte für die ungerade oder die „untere" Hälfte des Bildschirmspeichers für die gerade Spalte aktiviert.
c) Nunmehr wird die Spaltenzahl durch 2 geteilt und dann die der 40-Zeichen-Bildschirmroutine entsprechende Funktion aufgerufen.

Um dies besser zu verstehen, beschränken wir uns auf einen Abschnitt der ersten Zeile des 80-Zeichen-Textbildschirms (**Diagramm 2**). HTAB-Befehle arbeiten applesoftmäßig mit Spaltenzahlen im Bereich 1-80 und monitormäßig mit Spaltenzahlen im Bereich 0-79. Die eigentliche Bildschirm-Poke-Routine arbeitet hingegen mit Spaltenzahlen im Bereich 0-39, die jedoch zweimal vorkommen, je nachdem ob der Softswitch $C055 (= Page 2) für die obere oder der Softswitch $C054 (= Page 1) für die untere Hälfte aktiviert wurde.



Experiment 2

Zu Testzwecken schalten wir mit
PR#3
die 80-Zeichenkarte ein und gehen dann mit
CALL -151
in den Monitor. Wenn wir nun
C055:0 N 400:C1
eingeben, erscheint ein „A" (= Hex-Code $C1) in VTAB 1, HTAB 1. Dieses „A" ist intern in der Spalte 0 der oberen Bildschirmhälfte abgelegt. Gibt man nun
C054:0 N 400:C2
ein, dann erscheint rechts neben dem „A" ein „B" (= Hex-Code $C2), das somit die Position VTAB 1, HTAB 2 einnimmt und in der Spalte 0 der unteren Bildschirmhälfte liegt.

3. Doppelte Lores-Grafik

Mit dem Wissen über die einfache Lores-Grafik sowie der groben Kenntnis der Speicherorganisation der 80-Zeichenkarte können wir nunmehr die doppelte Lores-Grafik in den Griff bekommen.

Experiment 3

Zu Testzwecken schalten wir mit
PR#3
die 80-Zeichenkarte ein und dann mit
GR
auf Lores-Modus um. Nun gehen wir erneut mit
CALL -151
in den Monitor und geben
C05E:0
ein. Dieser Softswitch, der eigentlich den Annunciator 3 betrifft, aktiviert die doppelte Lores-Grafik. Wir werden jetzt lauter gesperrte Lores-Kästchen sehen, gesperrt deshalb, weil in jeder Zeile nur jedes zweite Kästchen geplottet ist. Diese Kästchen rühren daher, daß nach PR#3 der Text-Bildschirmspeicher gelöscht, d.h. mit $A0 = Spaces = Leertasten gefüllt wurde. Nach GR wurde die *obere* Hälfte des Bildschirmspeichers nicht gelöscht, d.h. mit $00 gefüllt, weil die Lores-Monitor-Routinen nichts von der Double Lores wissen. Folglich werden die Spaces jetzt als Kästchen in der oberen Hälfte interpretiert. Wir entfernen diese momentan störenden Kästchen-Muster mit

C055:0 N F832G
und geben danach ggf. noch
C054:0

ein, damit sich auch die unteren 4 Textzeilen des gemischen Lores-Text-Bildschirms normalisieren. Wir werfen nun ei-

nen kurzen Blick auf **Diagramm 3**. In der linken unteren Ecke dieses Schaubildes sehen wir ein Low-High-Nibble-Muster für die Speicherstelle $0400, das wir wie folgt verdeutlichen können:

Ll Lr = L links + rechts
Hl Hr = H links + rechts

Diagramm 3

	$0400								
$C055	0		1		2		3		
$C054	0		1		2		3		
Spalten	0	1	2	3	4	5	6	7	
Low	L	L	L	L	L	L	L	L	
High	H	H	H	H	H	H	H	H	

Stellen wir uns vor, daß die 2 oberen L's und die 2 unteren H's zusammen 4 Double-Lores-Kästchen darstellen, wobei L für Low Nibble und H für High Nibble steht. Bei der einfachen Lores-Grafik enthielt beim Experiment 1 die Speicherstelle $0400 1 Byte oder 2 Halbbytes = Nibbles, die zwei aufeinandergesetzte Lores-Kästchen in der Form

L
H

bildeten. Bei der doppelten Lores-Grafik existiert jedoch die Speicherstelle $0400 physisch zweimal, nämlich „unten" (Page 1) und „oben" (Page 2). Infolgedessen muß man nunmehr zwischen linken und rechten Low-High-Nibbles differenzieren. Dies läßt sich beweisen, indem wir folgende 4 Zeilen eingeben:

C055:0 N 400:0F
plottet das Ll-Kästchen (links oben).
C055:0 N 400:F0
plottet das Hl-Kästchen (links unten).
C054:0 N 400:0F
plottet das Lr-Kästchen (rechts oben).
C054:0 N 400:F0
plottet das Hr-Kästchen (rechts unten).

Bei der Double Lores gibt es mithin 80 statt 40 Kästchen pro Zeile. Da indessen weder im Applesoft-Interpreter noch im Monitor spezielle Double-Lores-Routinen enthalten sind, muß das Aktivieren der oberen respektive unteren Bildschirmhälfte durch Softswitch-Befehle des Applesoft- bzw. Assembler-Programms vorgenommen werden. Die Assembler-Befehle

LDA $C050 (Grafik)
LDA $C056 (Lores)
LDA $C052 (No Mix, Full Screen)
STA $C001 (80-Store)
STA $C00D (80-Column)
LDA $C05E (Double on)

schalten auf den Double-Lores-Modus um. Danach kann man nach Belieben mit

LDA $C055 (Page 2 = oben)
LDA $C054 (Page 1 = unten)

zwischen den zwei Bildschirmhälften $0400-$07FF umschalten. Der aufmerksame Leser wird jetzt sicher einwenden, daß in den Apple-Handbüchern die Page 1 als der Bereich $0400-$07FF sowie die Page 2 als der Bereich $0800-$0BFF beschrieben wird. Dies trifft auch zu und ist weiterhin gültig für die *einfache* Lores-Grafik. Bei der *doppelten* Lores-Grafik erhalten die Softswitches $C055 und $C054 jedoch vorübergehend die Funktion des Umschaltens zwischen unterem und oberem $0400-$07FF-Bereich, wobei $0800-$0BFF unberührt bleibt. Dies setzt jedoch voraus, das vorher das ganze Sortiment der obigen Softswitches, insbesondere $C001, $C00D und $C05E geschaltet worden sind. Man beachte im übrigen den Unterschied zwischen LDA (= PEEK) und STA (= POKE), der unbedingt eingehalten werden sollte.

Um die doppelte Lores-Grafik abzuschalten, sind folgende Softswitches erforderlich:

LDA $C054 (Page 1)
LDA $C051 (Text)
LDA $C05F (Double off)

STA $C000 (40-Store)
STA $C00C (40-Column)

die letzten beiden Softswitches dürfen nicht aktiviert werden, wenn *vor* dem Einschalten von Double Lores die 80-Zeichenkarte *bereits* eingeschaltet war, da

sonst die ROM-Routinen der 80-Zeichenkarte „durchdrehen".

Wenn Sie einen Apple IIe mit 80-Zeichenkarte besitzen und bei den obigen Experimenten keine Double Lores sahen, kann dies folgende hardware-technische Gründe haben:

– Sie besitzen keinen Apple IIe mit Platine Version B. Dann ist weder doppelte Loresnoch doppelte Hires-Grafik möglich. Allerdings sollen angeblich in Westdeutschland nur B-Version-Apples ausgeliefert worden sein.
– Sie haben die Brücke (Jumper), die mit der 80-Zeichenkarte mitgeliefert wurde, nicht auf die entsprechenden Pins gesteckt (siehe Apple-Ergänzungsblatt zur Bedienungsanleitung). Bei neueren 80-Zeichenkarten soll dies jedoch angeblich nicht mehr erforderlich sein.
– Sie haben eine Nachbau-80-Zeichenkarte, die nicht über die entsprechenden technischen Möglichkeiten verfügt.

4. Hinweise zu den Programmen

1. Double-Lores-Softswitch-Demo

Dieses Demo veranschaulicht, welche Softswitches in Applesoft aktiviert und desaktiviert werden müssen, um Double Lores ein- und abschalten zu können. Es empfiehlt sich, die 80-Zeichenkarte vor Double Lores mit
PRINT CHR$ (21)
oder mit
ESC Ctrl-Q
abzustellen, weil man dann eine bessere Kontrolle über die Softswitches hat, denn die ROM-Routinen betätigen einen Teil der Softswitches, die für Double Lores benötigt werden, so daß Speicherkonflikte auftreten können. Beispielsweise könnte aus Versehen der Bereich $0800-$0BFF als Textspeicher aktiviert werden, womit das dort befindliche Applesoft-Programm zerstört würde.

2. Double-Lores-Applesoft-Demo

Dieses leichtverständliche Demo zeichnet zunächst vertikale Linien in wechselnden Farben (Programmzeilen 37 – 55) und plottet dann eine schräge Linie (Programmzeilen 61 – 82). Man beachte, daß man bei schrägen sowie auch bei horizontalen Linien permanent zwischen Page 1 und Page 2 alternieren muß, da die ROM-Plot-Routinen bekanntlich nicht auf die obere Bildschirmhälfte umschalten können.

3. DOUBLE.LORES (Assembler)

Dieses Assembler-Demo dient zur Veranschaulichung der erforderlichen Softswitches sowie zur Erläuterung der Lores-Monitor-Routinen. Es zeichnet übrigens eine Art „Fußballtor".

4. AMPER.DOUBLE.LORES (Applesoft und Assembler)

Wir erinnern uns, daß eine Double-Lores-Zeile zwar 80 Kästchen – numeriert von 0-79 – umfaßt, jedoch wegen des in obere und untere Hälfte aufgeteilten Bildschirms nach Aktivierung der jeweiligen Page-Schalter immer nur Kästchen im Bereich 0-39 geplottet werden können. Dies ist hinderlich für die Programmierung, insbesondere wenn schräge und horizontale Linien sowie Kurven geplottet werden können. Besser wäre es, wenn man einen Plot-Befehl in der Art
PLOT X, Y
geben könnte, wobei X Werte im Bereich 0-79 annehmen würde. Dies ist jedoch wegen der unzureichenden ROM-Routinen nicht direkt möglich. Deshalb enthält die Ampersand-Utility AMPER.DOUBLE-.LORES eine Konvertierungsroutine, die X-Werte bis 79 zuläßt. Im einzelnen wurden folgende Befehle implementiert, die nach BRUN AMPER.DOUBLE.LORES über die Tastatur oder aus einem Applesoft-Programm heraus aufgerufen werden können:

& GR
schaltet auf Double-Lores-Modus um. Damit entfällt die lästige „Softswitcherei".
& TEXT
schaltet wieder auf 40-Zeichen-Textmodus zurück. Vor Anwendung der Utility AMPER.DOUBLE.LORES sollte man daher die 80-Zeichenkarte bereits abgestellt haben.
& PLOT X, Y
plottet ein Double-Lores-Kästchen in der zuvor durch COLOR = F gewählten Farbe, wobei X den Bereich 0-79 und Y den Bereich 0-47 umfassen kann. Werte außerhalb dieses Bereichs werden mit einem Piepston „quittiert".

An dieser Stelle soll nicht näher auf den Ampersand-Befehl (&) eingegangen werden, doch soll zumindest der Amper-Plot kurz erläutert werden:
Der Textpointer ist der Zeiger, der auf die jeweils gerade zu bearbeitende Stelle des Applesoft-Programms zeigt.

10 & PLOT X, Y
würde im Applesoft-Programmspeicher hexadezimal folgendermaßen aussehen:

1. $AF = Token für &
2. $8D = Token für PLOT
3. $58 = X
4. $2C = ,
5. $59 = Y

a) Wenn der Applesoft-Interpreter auf das Token für & stößt, lädt er das nachfolgende Byte, also in unserem Fall $8D, in den Akkumulator und springt nach $03F5, wo seinerseits durch AMPER.DOUBLE.LORES ein Sprung nach $0310 (siehe Zeile 62 des Assembler-Listings) installiert wurde.
b) Zeilen 62-68 prüfen, welches Token vorliegt. Im Falle von PLOT erfolgt dann ein Sprung zur Zeile 105 ($034F).
c) Der Textpointer wird nun in Zeile 105 durch die Routine CHRGET um 1 erhöht, womit gleichzeitig der Akkumulator mit dem Byte $58 für X geladen wird.
d) Dies ist die Voraussetzung für die Routine GETBYT, welche den *Wert* der Variablen X ermittelt und dann im X-Register des 6502-Prozessors hinterläßt. Dieser Wert wird zunächst zwischengespeichert (Zeile 107).
e) Danach wird in Zeile 108 durch die Routine CHKCOM (Check Comma) das Komma $2C übersprungen, wonach CHKCOM automatisch das nachfolgende Byte, also $59 für Y in den Akkumulator überträgt.
f) Nunmehr wird durch erneutes JSR GETBYT in Zeile 109 der Y-Wert ermittelt, der ebenfalls zunächst zwischengespeichert wird.
g) Jetzt wird es interessant (Zeile 114-119). Der Akkumulator wird mit dem X-Wert, der im Bereich 0-79 liegen darf, geladen und durch den Befehl LSR (Logical Shift Right) durch 2 geteilt, wodurch gleichzeitig Bit 0 in das Übertragsflag „Carry" übertragen wird. Nehmen wir an, der X-Wert sei dezimal 79 = hexadezimal $4F = binär %01001111, dann gilt:

76543210 = Bit-Nummern 0-7
01001111 = 79 vor LSR
00100111 = 39 nach LSR (Carry = 1)
Wenn Bit 0 vor LSR eine 1 enthielt, dann ist der X-Wert ungerade (BCS), andernfalls gerade (BCC). Wenn er ungerade ist, wird Page 1 aktiviert, andernfalls Page 2. Danach – d.h. nach der Teilung durch 2 durch LSR – kann die normale Monitor-PLOT-Routine $F800 (Zeile 133) aufgerufen werden.

h) Mit der Routine DATA (Zeile 141), die den Textpointer zum nächsten Statement vorrückt, und der Aktivierung von Page 1 (sicher ist sicher!) kehrt die Ampersand-Utility in das Applesoft-Programm zurück.

Das Applesoft-Demo AMPER.DOUBLE-.LORES.DEMO zeichnet übrigens eine Kugel, die durch Änderung der Variablen R für Radius sowie F1 und F2 modifiziert werden kann.

5. Schlußbemerkung

Double Lores ist eine nützliche Erweiterung der Grafikfähigkeiten des Apple IIe/IIc und empfiehlt sich stets dann, wenn die einfache Lores-Grafik zu grob und die einfache Hires-Grafik zu fein sein sollte, da die Double Lores ein Mittelding zwischen beiden normalen Grafik-Modi einnimmt. Mit der Utility AMPER.DOUBLE.LORES sind bei weitem noch nicht alle Möglichkeiten erschöpft. Beispielsweise fehlen noch folgende Befehle

& SCRN X, Y
& HLIN X1, X2 AT Y
& VLIN Y1, Y2 AT X
& PLOT X1, Y1 TO X2, Y2

Vielleicht entwickelt ein Leser dieser Zeitschrift diese zusätzlichen Ampersand-Befehle. Im nächsten Heft von „Peeker" wird dann über Double Hires berichtet.

```
1            ORG  $300
2   *
3   * DOUBLE.LORES
4   * ============
5   * von U.Stiehl 1984
6   *-------------------------
7   * Waagrechte X-Achse
8   * 0 (links) bis 39 (rechts)
9   *
10  * Senkrechte Y-Achse:
11  * 0 (oben) bis 39 (unten)   MIX
12  * 0 (oben) bis 47 (unten)   FULL
13  *-------------------------
14  * H2=X2; V2=Y2 Linien-Endpunkte
15  *
16  H2       EQU  $2C        ;X2
17  V2       EQU  $2D        ;Y2
18  *-------------------------
19  * Mixed-Lores- und Text-Modus
20  *
21  SETGR    EQU  $FB40      ;GR-mixed
22  *
23  INIT     EQU  $FB39      ;Textmodus
24  *
25  PAGE1    EQU  $C054
26  PAGE2    EQU  $C055
27  *-------------------------
28  * SETCOL: Farbe festlegen
29  * Color-Nr. (0-15) in A-Register
30  *
31  SETCOL   EQU  $F864
32  *
33  * SCRN: Farbe eines Punktes finden
34  * X-Abschnitt in Y-Register
35  * Y-Abschnitt in A-Register
36  * Danach Farbe in A-Register
37  *
38  SCRN     EQU  $F817
39  *-------------------------
40  * CLRSCR: Zeilen 0-47 löschen
41  * CLRTOP: Zeilen 0-39 löschen
42  *
43  CLRSCR   EQU  $F832      ;0-47
44  CLRTOP   EQU  $F836      ;0-39
45  *-------------------------
```

```
                        46  * PLOT: Punkt plotten
                        47  * X-Abschnitt in Y-Register
                        48  * Y-Abschnitt in A-Register
                        49  *
                        50  PLOT     EQU  $F800
                        51  *-------------------------------
                        52  * HLINE: Horizontale Linie plotten
                        53  * X1-links    in Y-Register
                        54  * X2-rechts   in H2
                        55  * Y-Abschnitt in A-Register
                        56  *
                        57  HLINE    EQU  $F819
                        58  *-------------------------------
                        59  * VLINE: Vertikale Linie plotten
                        60  * X-Abschnitt in Y-Register
                        61  * Y1-oben     in A-Register
                        62  * Y2-unten    in V2
                        63  *
                        64  VLINE    EQU  $F828
                        65  *
                        66  *-------------------------------
                        67  *
                        68  * Demo
                        69  * ====
                        70  *
0300: AD 50 C0          71          LDA  $C050      ;GRAFIK
0303: AD 56 C0          72          LDA  $C056      ;LORES
0306: AD 52 C0          73          LDA  $C052      ;NOMIX
0309: 8D 01 C0          74          STA  $C001      ;80STORE
030C: 8D 0D C0          75          STA  $C00D      ;80COL
030F: AD 5E C0          76          LDA  $C05E      ;80AN3
                        77  *
0312: AD 54 C0          78          LDA  PAGE1
0315: 20 32 F8          79          JSR  CLRSCR
0318: AD 55 C0          80          LDA  PAGE2
031B: 20 32 F8          81          JSR  CLRSCR
                        82  *
031E: A9 0A             83          LDA  #10
0320: 20 64 F8          84          JSR  SETCOL
                        85  *
                        86  * Punkt in der Mitte der
                        87  * untersten Zeile plotten
                        88  *
0323: AD 55 C0          89          LDA  PAGE2
0326: A0 13             90          LDY  #19        ;X
0328: A9 2F             91          LDA  #47        ;Y
032A: 20 00 F8          92          JSR  PLOT
                        93  *
032D: AD 54 C0          94          LDA  PAGE1
0330: A0 14             95          LDY  #20        ;X
0332: A9 2F             96          LDA  #47        ;Y
0334: 20 00 F8          97          JSR  PLOT
                        98  *
                        99  * Waagrechte Linie am oberen
                        100 * Lores-Bildschirmrand
                        101 *
0337: AD 55 C0          102         LDA  PAGE2
033A: A0 00             103         LDY  #0         ;X1
033C: A9 27             104         LDA  #39        ;X2
033E: 85 2C             105         STA  H2
0340: A9 00             106         LDA  #0         ;Y
0342: 20 19 F8          107         JSR  HLINE
                        108 *
0345: AD 54 C0          109         LDA  PAGE1
0348: A0 00             110         LDY  #0         ;X1
034A: A9 27             111         LDA  #39        ;X2
034C: 85 2C             112         STA  H2
034E: A9 01             113         LDA  #1         ;Y
0350: 20 19 F8          114         JSR  HLINE
                        115
                        116 * Senkrechte Linie am rechten
                        117 * Lores-Bildschirmrand
                        118 *
0353: AD 55 C0          119         LDA  PAGE2
0356: A0 26             120         LDY  #38        ;X
0358: A9 2F             121         LDA  #47        ;Y2
035A: 85 2D             122         STA  V2
035C: A9 02             123         LDA  #2         ;Y1
035E: 20 28 F8          124         JSR  VLINE
                        125 *
0361: AD 54 C0          126         LDA  PAGE1
0364: A0 27             127         LDY  #39        ;X
0366: A9 2F             128         LDA  #47        ;Y2
0368: 85 2D             129         STA  V2
036A: A9 02             130         LDA  #2         ;Y1
036C: 20 28 F8          131         JSR  VLINE
                        132 *
```

```
                        133 * Senkrechte Linie am linken
                        134 * Lores-Bildschirmrand
                        135 *
036F: AD 55 C0          136         LDA  PAGE2
0372: A0 00             137         LDY  #0         ;X
0374: A9 2F             138         LDA  #47        ;Y2
0376: 85 2D             139         STA  V2
0378: A9 02             140         LDA  #2         ;Y1
037A: 20 28 F8          141         JSR  VLINE
                        142 *
037D: AD 54 C0          143         LDA  PAGE1
0380: A0 01             144         LDY  #1         ;X
0382: A9 2F             145         LDA  #47        ;Y2
0384: 85 2D             146         STA  V2
0386: A9 02             147         LDA  #2         ;Y1
0388: 20 28 F8          148         JSR  VLINE
                        149 *
038B: 2C 10 C0          150 KEY1    BIT  $C010      ;STROBE
038E: AD 00 C0          151 KEY2    LDA  $C000      ;KEYBD
0391: 10 FB             152         BPL  KEY2
0393: 2C 10 C0          153         BIT  $C010      ;STROBE
0396: AD 54 C0          154         LDA  PAGE1
0399: 8D 00 C0          155         STA  $C000      ;40STORE
039C: 8D 0C C0          156         STA  $C00C      ;40COL
039F: AD 5F C0          157         LDA  $C05F      ;40AN3
03A2: 4C 39 FB          158         JMP  INIT

165 bytes
```

```
                        1           ORG  $300
                        2   *
                        3   * AMPER.DOUBLE.LORES
                        4   * ===================
                        5   *
                        6   * von U.Stiehl 1984
                        7   *
                        8   * Hex-Werte für Token
                        9   *
                        10  GRTOK    EQU  $88
                        11  TEXTTOK  EQU  $89
                        12  PLOTTOK  EQU  $8D
                        13  *
                        14  * Zwischenspeicher für X und Y
                        15  *
                        16  XACHSE   EQU  $00FE
                        17  YACHSE   EQU  $00FF
                        18  *
                        19  CLRSCR   EQU  $F832
                        20  PLOT     EQU  $F800
                        21  BELL     EQU  $FBDD
                        22  *
                        23  * CHRGET erhöht Textpointer um 1.
                        24  *
                        25  * DATA erhöht Textpointer bis
                        26  *  EOL oder ":".
                        27  *
                        28  * GETBYT lädt Zahl im Bereich
                        29  *  0-255 von Textpointer-Stelle
                        30  *  in X-Register. Vor JSR GETBYT
                        31  *  muß A-Register Textpointer-
                        32  *  Byte enthalten.
                        33  *
                        34  * CHKCOM prüft, ob Textpointer-
                        35  *  Byte "," enthält. Wenn ja,
                        36  *  wird Textpointer um 1 erhöht
                        37  *  und A-Register mit nächsten
                        38  *  Textpointer-Byte geladen.
                        39  *
                        40  CHRGET   EQU  $00B1
                        41  AMPERVEK EQU  $03F5
                        42  DATA     EQU  $D995
                        43  GETBYT   EQU  $E6F8      ;X
                        44  CHKCOM   EQU  $DEBE      ;","
                        45  *
                        46  PAGE1    EQU  $C054
                        47  PAGE2    EQU  $C055
                        48  *
                        49  * Ampersand-Vektor setzen
                        50  *
0300: A9 4C             51  START    LDA  #$4C       ;JMP
0302: 8D F5 03          52          STA  AMPERVEK
0305: A9 10             53          LDA  #<AMPER
0307: 8D F6 03          54          STA  AMPERVEK+1
030A: A9 03             55          LDA  #>AMPER
```

```
030C: 8D F7 03  56            STA  AMPERVEK+2
030F: 60        57            RTS
                58  *
                59  * A-Register enthält bereits Token
                60  * nach &  -->  &GR, &TEXT, &PLOT
                61  *
0310: C9 88     62  AMPER     CMP  #GRTOK
0312: F0 0B     63            BEQ  AMPGR
0314: C9 89     64            CMP  #TEXTTOK
0316: F0 28     65            BEQ  AMPTEXT
0318: C9 8D     66            CMP  #PLOTTOK
031A: F0 33     67            BEQ  AMPPLOT
031C: 4C 7D 03  68            JMP  ERROR
                69  *
                70  * & GR
                71  * ====
                72  *
                73  * Auf Double.Lores umschalten
                74  * und Bildschirm löschen
                75  *
031F: AD 50 C0  76  AMPGR     LDA  $C050      ;GRAFIK
0322: AD 56 C0  77            LDA  $C056      ;LORES
0325: AD 52 C0  78            LDA  $C052      ;NOMIX
0328: 8D 01 C0  79            STA  $C001      ;80STORE
032B: 8D 0D C0  80            STA  $C00D      ;80COL
032E: AD 5E C0  81            LDA  $C05E      ;80AN3
0331: AD 54 C0  82            LDA  PAGE1
0334: 20 32 F8  83            JSR  CLRSCR
0337: AD 55 C0  84            LDA  PAGE2
033A: 20 32 F8  85            JSR  CLRSCR
033D: 4C 80 03  86            JMP  EXIT
                87  *
                88  * & TEXT
                89  * ======
                90  *
                91  * Auf Text.Modus zurückschalten
                92  *
0340: 8D 00 C0  93  AMPTEXT   STA  $C000      ;40STORE
0343: 8D 0C C0  94            STA  $C00C      ;40COL
0346: AD 5F C0  95            LDA  $C05F      ;40AN3
0349: AD 51 C0  96            LDA  $C051      ;TEXT
034C: 4C 80 03  97            JMP  EXIT
                98  *
                99  * & PLOT X, Y
                100 * ==========
                101 *
                102 * X-Achse in Y-Register (0-79)
                103 * Y-Achse in A-Register (0-47)
                104 *
034F: 20 B1 00  105 AMPPLOT   JSR  CHRGET
0352: 20 F8 E6  106           JSR  GETBYT     ;X->X-Reg.
0355: 86 FE     107           STX  XACHSE
0357: 20 BE DE  108           JSR  CHKCOM     ;Komma
035A: 20 F8 E6  109           JSR  GETBYT     ;Y->X-Reg.
035D: 86 FF     110           STX  YACHSE
                111 *
                112 * Page 1 oder 2?
                113 *
035F: A5 FE     114           LDA  XACHSE
0361: 4A        115           LSR             ;A=A/2
0362: 90 05     116           BCC  PAGER2
0364: AC 54 C0  117 PAGER1    LDY  PAGE1
0367: B0 03     118           BCS  PARAMS
0369: AC 55 C0  119 PAGER2    LDY  PAGE2
                120 *
                121 * Wertebereich okay?
                122 *
036C: A8        123 PARAMS    TAY             ;X->Y-Reg.
036D: C0 28     124           CPY  #40        ;>=39?
036F: B0 0C     125           BCS  ERROR
                126 *
0371: A5 FF     127           LDA  YACHSE     ;Y->A-Reg.
0373: C9 30     128           CMP  #48        ;>=47?
0375: B0 06     129           BCS  ERROR
                130 *
                131 * Plotten und zurück zu Applesoft
                132 *
0377: 20 00 F8  133           JSR  PLOT
037A: 4C 80 03  134           JMP  EXIT
                135 *
                136 * Unzulässige Parameter
                137 *
037D: 20 DD FB  138 ERROR     JSR  BELL
                139 *
0380: AD 54 C0  140 EXIT      LDA  PAGE1
0383: 4C 95 D9  141           JMP  DATA

134 bytes
```

Double-Lores-Softswitch-Demo

```
10 P =  PEEK (49232): REM GR
11 POKE 49153,0: REM 80-STORE
12 POKE 49165,0: REM 80-COLUMN
13 P =  PEEK (49234): REM NO MIX
14 P =  PEEK (49246): REM DOUBLE ON
15 P =  PEEK (49237): CALL 63538: REM Clear Page2
16 P =  PEEK (49236): CALL 63538: REM Clear Page1
17 P =  PEEK (49237): POKE 1024,15: GET X$: REM Page2 =
   linker Punkt "oben"
18 P =  PEEK (49236): POKE 1024,15: GET X$: REM Page1 =
   rechter Punkt "unten"
19 TEXT : PRINT  CHR$ (4)"PR#3": PRINT  CHR$ (4)"PR#3": REM
   Softswitches normal; 2 * PR#3
```

Double-Lores-Applesoft-Demo

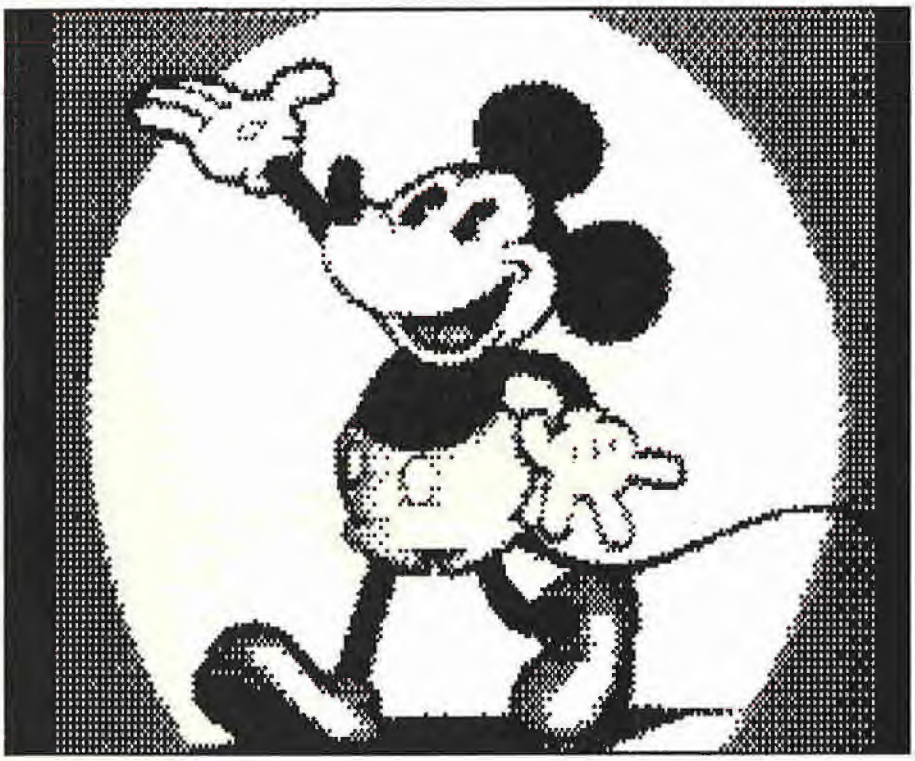
```
10  PRINT "DOUBLE.LORES IN APPLESOFT"
13 P =  PEEK (49232): REM GR
16  POKE 49153,0: REM 80STORE
19  POKE 49165,0: REM 80COL
22 P =  PEEK (49234): REM NOMIX
25 P =  PEEK (49246): REM AN3
28 P =  PEEK (49237): CALL 63538: REM CLRP2
31 P =  PEEK (49236): CALL 63538: REM CLRP1
34 P =  PEEK (49152): ON P < 128 GOTO 34:P =  PEEK (49168):
   REM KEY
37  FOR C = 15 TO 0 STEP  - 1: COLOR= C
40  FOR X = 0 TO 36 STEP 4
43 P =  PEEK (49237): REM PAGE2
46  VLIN 0,47 AT X
49 P =  PEEK (49236): REM PAGE1
52  VLIN 0,47 AT X + 1
55  NEXT X,C
58  COLOR= 10
61 X = 0:Y = 0
64 P =  PEEK (49237): REM PAGE2
67  PLOT X,Y
70 Y = Y + 1
73 P =  PEEK (49236): REM PAGE1
76  PLOT X,Y
79 X = X + 1:Y = Y + 1
82  IF Y < 48 GOTO 64
85 P =  PEEK (49152): ON P < 128 GOTO 85:P =  PEEK (49168):
   REM KEY
88  POKE 49152,0: REM 800FF
91  POKE 49164,0: REM 40COL
94 P =  PEEK (49236): REM PAGE1
97  TEXT
```

Amper.Double.Lores.Demo

```
10  PRINT  CHR$ (21): PRINT  CHR$ (4)"BRUN
   AMPER.DOUBLE.LORES"
15  REM Amper.Double.Lores.Demo zeichnet eine Kugel
20  TEXT : &  GR
25 XE = 79:YE = 47:A = XE / 2:B = YE / 2:F1 = 7:F2 = 23
30  FOR C = 1 TO 15: COLOR= C:R = C + C
35  FOR X = A - R TO A + R: IF X < 0 OR X > XE GOTO 60
40 H = R * R - (X - A) * (X - A): IF H <  = 0 GOTO 60
45 Y1 = B +  SQR (H) - F1:Y2 = B -  SQR (H) + F1
50  IF Y1 > 0 AND Y1 <  = YE THEN  &  PLOT X,Y1
55  IF Y2 > 0 AND Y2 <  = YE THEN  &  PLOT X,Y2
60  NEXT X
65  NEXT C
70  GET X$
75  &  TEXT : HOME
```

Hires-Grafik-Dump für Epsondrucker

Mit Hilfe des Applesoft-Programms HIRES und des Maschinen-Programms PRINTHIRES ist es möglich, die hochauflösende Grafik des Apple auf einem Epson Drucker, der für die Ausgabe der Grafik vorgesehen ist (MX-80, MX-82, MX-100 Typ 3 sowie RX-80 FX-80 etc.), auszugeben. Durch leichte Modifikationen der Programme ist es auch möglich, sie an einen anderen Drucker oder ein anderes Interface anzupassen.



Jürgen und Dieter Geiss

Nachdem Sie den Maschinenspracheteil des Programms in den Computer eingegeben haben, speichern Sie ihn bitte unter dem Namen „PRINTHIRES" mit einem BSAVE PRINTHIRES, A$1C00, L$0400. Danach geben Sie das Applesoft-Programm HIRES ein, und speichern Sie es dann unter dem Namen HIRES mit dem SAVE-Befehl ab.

Beschreibung des Applesoft-Programms

(Die halbfette Zahlen beziehen sich im nachfolgenden auf die Applesoft-Programmzeilen)

1000: ST ist die Startadresse für die Parameter, die an das Maschinenprogramm übergeben werden. Die Startadresse wird auf 768 Dezimal = $0300 Hex gesetzt.
1010: CO ist die Adresse für die Kontrollzeichensequenz, um den Drucker auf einen Grafikmodus zu schalten.
1020: BY ist die Anzahl der Bytes/Zeilen, die der Drucker nun als Grafikzeichen interpretiert.
1030: IM ist die Stärke (Impression), mit der gedruckt werden soll. Man unterscheidet:
a) Single Strike: Eine Grafikzeile wird einmal ausgedruckt.
b) Double Strike: Eine Grafikzeile wird ausgedruckt, dann wird ein Wagenrücklauf und ein Zeilenvorschub um 1/216 Inch bewirkt und die Zeile nochmals ausgedruckt.
c) Quadruple Strike: Eine Grafikzeile wird ausgedruckt, ein Wagenrücklauf ohne Zeilenvorschub erwirkt, die Zeile wird noch-

mals ausgedruckt. Dann wird wie beim Double Strike Modus ein Zeilenvorschub um 1/216 Inch bewirkt und die Zeile wieder zweimal gedruckt.

Anmerkung zu b) und c): Funktioniert nur bei Druckern der Marke Epson. Bei anderen Druckern müssen die Unterprogramme Linesp und LS1 im Assemblerprogramm entsprechend modifiziert werden.

1040: XO ist das Exclusive-Or-Byte. Die Grafikseite kann entweder positiv oder negativ ausgedruckt werden ($00 oder $FF).
a) positiv: Ein Punkt, der auf der Grafikseite gesetzt ist, wird auf dem Drucker ausgegeben.
b) negativ: Ein Punkt, der auf der Grafikseite nicht gesetzt ist, wird auf dem Drukker ausgegeben.

1050: DI ist die Richtung (Direction), in der gedruckt werden soll. Es gibt die Möglichkeiten horizontal oder vertikal.

1060: FR gibt an, ob ein Rand (Frame) das Hi-Res-Bild einrahmen soll. Der Rand ist ein Bit breit.

1070: PA ist die Grafikseite (Page), die ausgedruckt werden soll.
1. Page = $2000-$3FFF
2. Page = $4000-$5FFF

1080: RF bestimmt den Wiederholungsfaktor (Repeat Factor). Ein (Teil-)Bild kann mehrere Male nebeneinander ausgedruckt werden.

1090: RD ist die Wiederholungsdistanz (Repeat Distance) und gibt die Anzahl der Bits zwischen den wiederholten Bildern an.

1100: LE ist der linke Rand (left) des Hires-Bildes. Die Werte in LE und LE + 1 berechnen sich folgendermaßen: Das höherwertige Byte (LE + 1) gibt das horizontale Byte der Hires-Page an, das niederwertige Byte das Bit in diesem Byte. Ist der linke Rand also 10, so muß in die Speicherzelle LE + 1 eine 1, in LE eine 3 gepokt werden (1 * 7 + 3, da sieben Bits pro Byte ohne das Farbenbit gezählt werden).

1110: RI ist der rechte (right) Rand. Die Berechnung entspricht der des linken Randes.

1120: TP ist der obere (top) Rand. Er kann Werte zwischen 0 und 191 annehmen.

1130: BO ist der untere (bottom) Rand. Auch hier treten Werte zwischen 0 und 191 auf.

1140: XF ist der X-Faktor. Das Bild kann in X-Richtung (horizontal) gestreckt werden. Eine 1 entspricht der Originalgröße, bei einer 2 wird das Bild doppelt so groß in X-Richtung. Der Maximale Wert ist 31.

```
                        1           ORG  $1C00
                        2
                        3   ******************************************************
                        4   * PRINTHIRES                                         *
                        5   * FX-80, FX-100, RX-80, RX-100, MX-80, MX-82, MX-100 *
                        6   ******************************************************
                        7
                        8   Base     EQU  $1C        ;Adresse fuer Graphikzeile
                        9   Frameadr EQU  $1E        ;Hilfsadresse fuer Bildrahmen
                       10
                       11   Start    EQU  $0300      ;Start fuer Parameteruebergabe
                       12   Amprsand EQU  $03F6      ;Basic &-Vector
                       13
                       14   Control  EQU  Start+$00  ;Graphik Steuerzeichen fuer
                       15   *                        ;Drucker ohne ESC-Zeichen
                       16   Bytes    EQU  Start+$02  ;Anzahl der Graphikbytes pro Zeile
                       17   Impres   EQU  Start+$04  ;1-, 2- oder 4-fach Dichte
                       18   XOR      EQU  Start+$06  ;Exklusives oder fuer jedes Byte
                       19   Direct   EQU  Start+$08  ;Richtung (0=horiz., 1=vertik.)
                       20   Frame_yn EQU  Start+$0A  ;0=Rand, 1=kein Rand
                       21   Page     EQU  Start+$0C  ;Graphik Seite 1 oder 2
                       22   Rfactor  EQU  Start+$0E  ;Anzahl der Bilder pro Zeile
                       23   Rdistnce EQU  Start+$10  ;Distanz zw. 2 oder mehr Bildern
                       24   Left     EQU  Start+$12  ;linker Bildrand (0..279)
                       25   Right    EQU  Start+$14  ;rechter Bildrand (0..279)
                       26   Top      EQU  Start+$16  ;oberer Bildrand (0..191)
                       27   Bottom   EQU  Start+$18  ;unterer Bildrand (0..191)
                       28   Xfactor  EQU  Start+$1A  ;Dehnung in X-Richtung
                       29   Yfactor  EQU  Start+$1C  ;Dehnung in Y-Richtung
                       30
                       31   Lnumber  EQU  Start+$20  ;Graphikzeilennummer
                       32   Asave    EQU  Start+$21
                       33   Ysave    EQU  Start+$22
                       34   Hcount   EQU  Start+$23
                       35   Vcount   EQU  Start+$24
                       36   Bitcount EQU  Start+$25
                       37   Counter  EQU  Start+$26
                       38   Rcounter EQU  Start+$27
                       39   Xcount   EQU  Start+$28
                       40   Ycount   EQU  Start+$29
                       41
                       42   LF       EQU  $0A        ;ASCII Code fuer Zeilenvorschub
                       43   CR       EQU  $0D        ;ASCII Code fuer Wagenruecklauf
                       44   ESC      EQU  $1B        ;ASCII Code fuer ESCAPE
                       45
                       46   Byt0_255 EQU  $0200      ;Zwischenspeicherung
                       47
                       48   Prout    EQU  $C090      ;Zeichen an Drucker (parallel)
                       49   Prstrb   EQU  $C1C1      ;Drucker bereit?
                       50
                       51   ******************************************************
                       52
1C00: A9 0B            53           LDA  #<Pgmstart ;&-Vektor fuer Basic an den Pro-
                       54   *                       ;grammstart einstellen
1C02: 8D F6 03         55           STA  Amprsand
1C05: A9 1C            56           LDA  #>Pgmstart
1C07: 8D F7 03         57           STA  Amprsand+1
1C0A: 60               58           RTS
                       59
                       60   ******************************************************
                       61
1C0B: AD 0C 03         62   Pgmstart LDA  Page       ;initialisieren von Seite 1 oder 2
1C0E: D0 07            63            BNE  Page_2
1C10: A9 20            64            LDA  #$20
1C12: 8D 0C 03         65            STA  Page
1C15: D0 05            66            BNE  Frametst
1C17: A9 40            67   Page_2   LDA  #$40       ;Graphik-Seite 2
1C19: 8D 0C 03         68            STA  Page
                       69
1C1C: AD 0A 03         70   Frametst LDA  Frame_yn   ;Test, auf Bildrahmen
1C1F: D0 03            71            BNE  No_Frame
1C21: 20 04 1F         72            JSR  Frame       ;zeichne Rahmen
                       73
1C24: AD 08 03         74   No_Frame LDA  Direct     ;Test auf horiz. oder vert.
1C27: F0 03            75            BEQ  Horprint
1C29: 4C 76 1D         76            JMP  Verprint
                       77
                       78   ******************************************************
                       79
1C2C: A9 17            80   Horprint LDA  #23        ;23/216" Zeilenvorschub
1C2E: 20 D6 1E         81            JSR  Linesp
                       82
1C31: AC 16 03         83            LDY  Top         ;Zeile mit der begonnen wird
1C34: 8C 20 03         84            STY  Lnumber
                       85
1C37: A9 01            86   Hstart   LDA  #$01       ;Zaehler fuer Dehnung in
1C39: 8D 26 03         87            STA  Counter     ;Y-Richtung
                       88
```

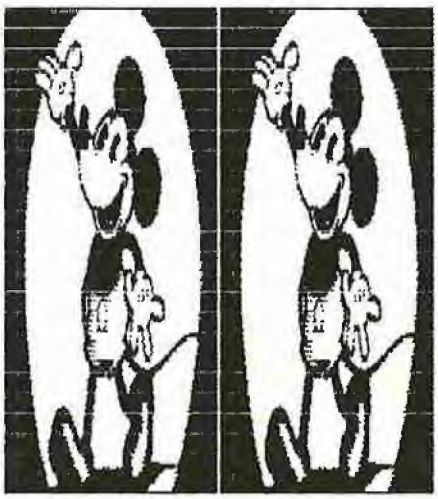
1150: YF ist der Y-Faktor. Das Bild kann in der Vertikalen gestreckt werden. Eine 1 ist auch hier die Originalgröße.
1180 – 1210: Es wird die Möglichkeit gegeben, das augenblicklich im Speicher in der ersten Grafikseite befindliche Bild abzuspeichern. Dies wird immer unter dem Namen „PICTURE" gemacht.
1220 – 1230: Das Maschinenprogramm „PRINTHIRES" wird geladen und gestartet. Es stellt den Ampersand-Vektor (&) auf den Wert $1C0B ein. Damit kann das eigentliche Druckprogramm mit einem einfachen &-Befehl gestartet werden.
1250 – 1290: Es wird gefragt, ob sich die Grafik schon im Speicher befindet. Bei einem No wird der Disketteninhalt gezeigt und es kann eine Grafik geladen werden.
1300 – 1330: Der Bildschirm wird auf Grafik umgeschaltet.
1340 – 1370: Wenn auf die Frage „Grundeinstellung?" mit Yes geantwortet wird, werden die Parameter voreingestellt. Und zwar:

CO	= ASC („K")	
BY	= 280 Bytes	
IM	= 0	= single strike
XO	= 0	= positiv
DI	= 0	= horizontal
FR	= 0	= Rand
PA	= 0	= erste Seite
RF	= 1	
RD	= 0	
LE	= 0	
RI	= 6/39	= low/high
TP	= 0	
BO	= 191	
XF	= 1	
YF	= 1	

1380 – 1460: Abfrage der Parameter
1470 – 1640: Abfrage der zu ladenden Grafik
1650 – 1700: Abfrage, ob richtiges Bild vorhanden ist
1710 – 1760: Abfrage, ob andere Page gezeigt werden soll
1770 – 1830: Zeigen der zweiten Grafikseite
1840 – 1850: Zurück zur Abfrage der zu ladenden Grafik
1860 – 1900: Druck der Grafik
1910 – 1980: Abfrage des nochmaligen Druckens und Endes des Hauptprogramms
1990 – 2080: Voreinstellung der Parameter
2090 – 2130: Abfrage eines Kommandos
2140 – 2320: Einstellen des Druckmodus
2340 – 2400: Einstellen der Druckstärke

```
1C3C: 20 5E 1C  89  Hstart1  JSR  Hlline     ;Bei Dehnung wird horizontale
1C3F: AD 26 03  90           LDA  Counter    ;Zeile entsprechend oft gedruckt
1C42: CD 1C 03  91           CMP  Yfactor
1C45: F0 06     92           BEQ  Hstart2
1C47: EE 26 03  93           INC  Counter
1C4A: 4C 3C 1C  94           JMP  Hstart1
                95
1C4D: AD 20 03  96  Hstart2  LDA  Lnumber    ;Neue Zeile
1C50: 18        97           CLC
1C51: 69 08     98           ADC  #$08       ;8 Bits pro Zeile werden gedruckt
1C53: 8D 20 03  99           STA  Lnumber
1C56: CD 18 03  100          CMP  Bottom
1C59: 90 DC     101          BCC  Hstart
1C5B: 4C 99 1F  102          JMP  Quit
                103
1C5E: 20 E6 1E  104 Hlline   JSR  CRLF       ;Drucke horizontale Zeile
                105
1C61: 20 8F 1C  106 Hrepeat1 JSR  PrHline
1C64: 20 BD 1E  107          JSR  Sendbits
1C67: CE 27 03  108          DEC  Rcounter   ;Zaehler fuer Bilderanzahl/Zeile
1C6A: D0 F5     109          BNE  Hrepeat1
                110
1C6C: 20 4A 1D  111          JSR  PrHagain   ;if Impression = 4-fach Dichte
                112
1C6F: 20 F4 1E  113          JSR  LS1        ;1/216" Zeilenvorschub
                114
1C72: AE 04 03  115          LDX  Impres
1C75: E0 01     116          CPX  #$01       ;doppelte Dichte
1C77: F0 07     117          BEQ  Hrptl
1C79: E0 02     118          CPX  #$02
1C7B: D0 11     119          BNE  SkpDstrk   ;keine doppelte Dichte
                120
1C7D: 20 4A 1D  121          JSR  PrHagain
1C80: 20 92 1E  122 Hrptl    JSR  Initline
1C83: 20 8F 1C  123 Hrepeat2 JSR  PrHline
1C86: 20 BD 1E  124          JSR  Sendbits
1C89: CE 27 03  125          DEC  Rcounter
1C8C: D0 F5     126          BNE  Hrepeat2
1C8E: 60        127 SkpDstrk RTS
                128
                129 ************************************************************
                130
1C8F: AD 12 03  131 PrHline  LDA  Left       ;druckt eine horizontale Zeile
1C92: 8D 25 03  132          STA  Bitcount   ;holt linker Rand in bits
1C95: AC 13 03  133          LDY  Left+1     ;hires byte
1C98: 8C 22 03  134 X_loop   STY  Ysave
1C9B: AD 20 03  135          LDA  Lnumber
1C9E: 8D 24 03  136          STA  Vcount
1CA1: A9 08     137          LDA  #$08       ;8 Bit werden gedruckt
1CA3: 8D 23 03  138          STA  Hcount
                139
1CA6: AD 24 03  140 Getbyte  LDA  Vcount     ;Zeilenadresse berechnen
1CA9: 20 76 1F  141          JSR  Bascalc
1CAC: B1 1C     142          LDA  (Base),Y   ;hole ein Byte
1CAE: AE 25 03  143          LDX  Bitcount   ;Rechtsverschiebung bis linker
1CB1: 4A        144 Getbytel LSR             ;Rand im Byte erreicht ist
1CB2: CA        145          DEX             ;hole jeweils ein Bit aus den
1CB3: 10 FC     146          BPL  Getbytel   ;naechsten 8 Zeilen und vereinige
1CB5: 2E 21 03  147          ROL  Asave      ;diese zu einem Byte
1CB8: CE 23 03  148          DEC  Hcount
1CBB: F0 15     149          BEQ  Hgood
1CBD: AD 24 03  150          LDA  Vcount
1CC0: CD 18 03  151          CMP  Bottom
1CC3: F0 05     152          BEQ  Getbyte2
1CC5: EE 24 03  153          INC  Vcount
1CC8: D0 DC     154          BNE  Getbyte
1CCA: 0E 21 03  155 Getbyte2 ASL  Asave
1CCD: CE 23 03  156          DEC  Hcount
1CD0: D0 F8     157          BNE  Getbyte2
                158
1CD2: A0 08     159 Hgood    LDY  #$08       ;8 Bits
1CD4: AD 21 03  160          LDA  Asave      ;vergroessere in Y-Richtung um
1CD7: 4A        161 Hloop1   LSR             ;Y-Faktor
1CD8: 08        162          PHP
1CD9: AE 1C 03  163          LDX  Yfactor
1CDC: 8E 29 03  164          STX  Ycount
1CDF: AE 1C 03  165 Hloop2   LDX  Yfactor
1CE2: 8E 23 03  166          STX  Hcount
1CE5: A2 00     167          LDX  #$00
1CE7: 3E 00 02  168 Hloop3   ROL  Byt0_255,X
1CEA: E8        169          INX
1CEB: CE 23 03  170          DEC  Hcount
1CEE: D0 F7     171          BNE  Hloop3
1CF0: 28        172          PLP
1CF1: 08        173          PHP
1CF2: CE 29 03  174          DEC  Ycount
1CF5: D0 E8     175          BNE  Hloop2
1CF7: 28        176          PLP
```

2410 – 2470: Einstellen des Positiv/Negativ-Drucks
2480 – 2540: Einstellen der Druckrichtung
2550 – 2610: Einstellen der Bildumrandung
2620 – 2680: Wählen der Grafikseite
2700 – 2990: Einstellen und Ausrechenen der restlichen Parameter
3000 – 3060: Berechnen der Byteanzahl (BY) in Abhängigkeit vom linken/rechten Rand und vom X- oder Y-Faktor.



Bedienung des Applesoft-Programms

Das Applesoft-Programm ist menüartig gesteuert und somit leicht zu bedienen. Die Abfragen erwarten entweder ein J/N oder Y/N oder eine Zahl. In beiden Fällen braucht die RETURN-Taste nicht betätigt zu werden. Es ist egal, ob Sie Groß- oder Kleinbuchstaben eingeben. Nur bei der Frage „Welche Grafik laden?“ muß nach dem Filenamen die RETURN-Taste betätigt werden.

Modifikation des Applesoft-Programms für andere Drucker

2140 – 2320: Je nach Möglichkeiten des Druckers müssen in CO und CO + 1 die Steuerzeichen (ohne führendes ESC) geschrieben werden. Als Beispiel sei die Zeile 2260 angeführt, die in CO den ASCII-Wert von K schreibt, damit das Maschinenprogramm an den Drucker die Sequenz ESC „K“ schickt. Dies schaltet einen Epson Drucker auf Normalgraphik.
Das Applesoft-Programm läuft mit den meisten Interfaces.

```
1CF8: 88          177           DEY
1CF9: D0 DC       178           BNE  Hloop1
                  179
1CFB: AE 26 03    180           LDX  Counter
1CFE: 8E 24 03    181           STX  Vcount
1D01: A0 08       182 Hloop4    LDY  #$08
1D03: 18          183 Hloop5    CLC
1D04: AE 1C 03    184           LDX  Yfactor
1D07: CA          185           DEX
1D08: 7E 00 02    186 Hloop6    ROR  Byt0_255,X
1D0B: CA          187           DEX
1D0C: 10 FA       188           BPL  Hloop6
1D0E: 2A          189           ROL
1D0F: 88          190           DEY
1D10: D0 F1       191           BNE  Hloop5
1D12: CE 24 03    192           DEC  Vcount
1D15: D0 EA       193           BNE  Hloop4
                  194
1D17: 4D 06 03    195           EOR  XOR
1D1A: AE 1A 03    196           LDX  Xfactor    ;vergroessert in X-Richtung um
                  197
1D1D: 20 78 1E    198 Hagain    JSR  Print      ;X-Faktor
1D20: CA          199           DEX
1D21: D0 FA       200           BNE  Hagain
                  201
1D23: AC 22 03    202           LDY  Ysave      ;rechter Bildschirmrand schon
1D26: CC 15 03    203           CPY  Right+1    ;erreicht?
1D29: D0 09       204           BNE  Hagain1
1D2B: AD 25 03    205           LDA  Bitcount
1D2E: CD 14 03    206           CMP  Right
1D31: D0 01       207           BNE  Hagain1
1D33: 60          208           RTS             ;ja, dann Ende
1D34: EE 25 03    209 Hagain1   INC  Bitcount
1D37: AD 25 03    210           LDA  Bitcount   ;alle 7 Bits schon abgearbeitet?
1D3A: C9 07       211           CMP  #$07       ;7 hires Bits pro Byte
1D3C: F0 03       212           BEQ  Hagain2
1D3E: 4C 98 1C    213           JMP  X_loop     ;nein
1D41: C8          214 Hagain2   INY             ;naechstes Byte
1D42: A9 00       215           LDA  #$00
1D44: 8D 25 03    216           STA  Bitcount
1D47: 4C 98 1C    217           JMP  X_loop
                  218
                  219 ************************************************************
                  220
1D4A: AD 04 03    221 PrHagain LDA  Impres
1D4D: C9 02       222           CMP  #$02       ;vierfache Dichte
1D4F: D0 0E       223           BNE  PrHagan2
1D51: 20 92 1E    224           JSR  Initline
1D54: 20 8F 1C    225 PrHagan1 JSR  PrHline
1D57: 20 BD 1E    226           JSR  Sendbits
1D5A: CE 27 03    227           DEC  Rcounter
1D5D: D0 F5       228           BNE  PrHagan1
1D5F: 60          229 PrHagan2 RTS
                  230
                  231 ************************************************************
                  232
1D60: AD 04 03    233 PrVagain LDA  Impres
1D63: C9 02       234           CMP  #$02       ;vierfache Dichte
1D65: D0 0E       235           BNE  PrVagan2
1D67: 20 92 1E    236           JSR  Initline
1D6A: 20 D1 1D    237 PrVagan1 JSR  PrVline
1D6D: 20 BD 1E    238           JSR  Sendbits
1D70: CE 27 03    239           DEC  Rcounter
1D73: D0 F5       240           BNE  PrVagan1
1D75: 60          241 PrVagan2 RTS
                  242
                  243 ************************************************************
                  244
1D76: A9 14       245 Verprint LDA  #20         ;20/216" Zeilenvorschub
1D78: 20 D6 1E    246           JSR  Linesp     ;der vertikale Druck funktioniert
                  247
1D7B: AC 15 03    248           LDY  Right+1    ;wie der horizontale Druck, mit
1D7E: 8C 22 03    249 Vstart    STY  Ysave      ;dem Unterschied, dass immer nur
1D81: A9 01       250           LDA  #$01       ;7 Bits pro Zeile ausgedruckt wer-
1D83: 8D 26 03    251           STA  Counter    ;den.
                  252
1D86: 20 E6 1E    253 Countrlp JSR  CRLF
1D89: 20 D1 1D    254 Vrepeat1 JSR  PrVline
1D8C: 20 BD 1E    255           JSR  Sendbits
1D8F: CE 27 03    256           DEC  Rcounter
1D92: D0 F5       257           BNE  Vrepeat1
                  258
1D94: 20 60 1D    259           JSR  PrVagain
                  260
1D97: 20 F4 1E    261           JSR  LS1
                  262
1D9A: AE 04 03    263           LDX  Impres
1D9D: E0 01       264           CPX  #$01
1D9F: F0 07       265           BEQ  Not_ds
```

Modifikation des Maschinen-Programms für andere Drucker oder Interfaces

379 – 382: In Asave befindet sich das Byte, das auszudrucken ist. Ist der Drukker noch nicht fertig, wartet er in der Prwait-Schleife. Das Programm wartet, wenn der Printerstrobe negativ ist. Der Printerstrobe kann in Zeile 48 geändert werden. Ist der Drucker fertig, wird das Zeichen an den Drucker gesandt. Dies geschieht mit einem STA $C090 (gültig für Slot 1). In Zeile 47 kann das geändert werden.
Wichtig ist also, wie das Interface erkennt, ob der Drucker schon druckbereit ist, und wie ein Zeichen an den Drucker geschickt wird.
429 – 438: Hier muß die Kontrollsequenz untergebracht sein, die bewirkt, daß der Drucker auf n/216 Inch Zeilenabstand umschaltet. Beim Epson ist dies ESC „3".
449 – 457: Hier muß sich die Kontrollsequenz befinden, die das Papier um 1/216 Inch vorwärtsbewegt. Beim Epson ist dies ESC „J" 1.

Hinweis: Eine modifizierte Version von PRINTHIRES für den Image-Writer befindet sich in Vorbereitung für die Peeker-Sammeldiskette.


```
1DA1: E0 02     266           CPX  #$02
1DA3: D0 11     267           BNE  Skip_ds    ;keine doppelte Dichte
                268
1DA5: 20 60 1D  269           JSR  PrVagain
1DA8: 20 92 1E  270  Not_ds   JSR  Initline
1DAB: 20 D1 1D  271  Vrepeat2 JSR  PrVline
1DAE: 20 BD 1E  272           JSR  Sendbits
1DB1: CE 27 03  273           DEC  Rcounter
1DB4: D0 F5     274           BNE  Vrepeat2
                275
1DB6: AD 26 03  276  Skip_ds  LDA  Counter
1DB9: CD 1A 03  277           CMP  Xfactor
1DBC: F0 06     278           BEQ  Next_X
1DBE: EE 26 03  279           INC  Counter
1DC1: 4C 86 1D  280           JMP  Countrlp
                281
1DC4: AD 22 03  282  Next_X   LDA  Ysave
1DC7: A8        283           TAY
1DC8: 88        284           DEY
1DC9: CD 13 03  285           CMP  Left+1
1DCC: D0 B0     286           BNE  Vstart
1DCE: 4C 99 1F  287           JMP  Quit
                288
                289  ************************************************************
                290
1DD1: AD 16 03  291  PrVline  LDA  Top
1DD4: 8D 20 03  292           STA  Lnumber
1DD7: 20 76 1F  293  Lineloop JSR  Bascalc
1DDA: AC 22 03  294           LDY  Ysave
1DDD: B1 1C     295           LDA  (Base),Y
1DDF: 8D 21 03  296           STA  Asave
1DE2: CC 15 03  297           CPY  Right+1
1DE5: D0 16     298           BNE  RMbit      ;rightmost bit
1DE7: A9 06     299           LDA  #$06
1DE9: 38        300           SEC
1DEA: ED 14 03  301           SBC  Right
1DED: F0 0E     302           BEQ  RMbit
1DEF: AA        303           TAX
1DF0: E8        304           INX
1DF1: A9 FF     305           LDA  #$FF
1DF3: 4A        306  Lnloopl  LSR
1DF4: CA        307           DEX
1DF5: D0 FC     308           BNE  Lnloopl
1DF7: 2D 21 03  309           AND  Asave
1DFA: 8D 21 03  310           STA  Asave
1DFD: AD 21 03  311  RMbit    LDA  Asave
1E00: AC 13 03  312           LDY  Left+1
1E03: CC 22 03  313           CPY  Ysave
1E06: D0 0E     314           BNE  Vok
1E08: AE 12 03  315           LDX  Left
1E0B: F0 09     316           BEQ  Vok
1E0D: A9 FF     317           LDA  #$FF
1E0F: 0A        318  Genmask  ASL
1E10: CA        319           DEX
1E11: D0 FC     320           BNE  Genmask
1E13: 2D 21 03  321           AND  Asave
1E16: A0 07     322  Vok      LDY  #$07
1E18: 4A        323  Vok1     LSR
1E19: 08        324           PHP
1E1A: AE 1A 03  325           LDX  Xfactor
1E1D: 8E 28 03  326           STX  Xcount
1E20: AE 1A 03  327  Vok20    LDX  Xfactor
1E23: 8E 23 03  328           STX  Hcount
1E26: A2 00     329           LDX  #$00
1E28: 3E 00 02  330  Vok2     ROL  Byt0_255,X
1E2B: E8        331           INX
1E2C: CE 23 03  332           DEC  Hcount
1E2F: D0 F7     333           BNE  Vok2
1E31: 28        334           PLP
1E32: 08        335           PHP
1E33: CE 28 03  336           DEC  Xcount
1E36: D0 E8     337           BNE  Vok20
1E38: 28        338           PLP
1E39: 88        339           DEY
1E3A: D0 DC     340           BNE  Vok1
                341
1E3C: AE 26 03  342           LDX  Counter
1E3F: 8E 24 03  343           STX  Vcount
1E42: A0 07     344  Vok3     LDY  #$07       ;7 Bits pro Zeile
1E44: 18        345  Vok4     CLC
1E45: AE 1A 03  346           LDX  Xfactor
1E48: CA        347           DEX
1E49: 7E 00 02  348  Vok5     ROR  Byt0_255,X
1E4C: CA        349           DEX
1E4D: 10 FA     350           BPL  Vok5
1E4F: 2A        351           ROL
1E50: 88        352           DEY
1E51: D0 F1     353           BNE  Vok4
```

```
1E53: CE 24 03   354           DEC  Vcount
1E56: D0 EA      355           BNE  Vok3
                 356
1E58: 4D 06 03   357           EOR  XOR
1E5B: 29 7F      358           AND  #$7F
1E5D: AC 1C 03   359           LDY  Yfactor
                 360
1E60: 20 78 1E   361  Vagain    JSR  Print
1E63: 88         362           DEY
1E64: D0 FA      363           BNE  Vagain
                 364
1E66: AE 20 03   365           LDX  Lnumber
1E69: EE 20 03   366           INC  Lnumber
1E6C: AD 20 03   367           LDA  Lnumber
1E6F: EC 18 03   368           CPX  Bottom
1E72: B0 03      369           BCS  Vagainl
1E74: 4C D7 1D   370           JMP  Lineloop
                 371
1E77: 60         372  Vagainl  RTS
                 373
                 374  ************************************************************
                 375
1E78: 8D 21 03   376  Print     STA  Asave       ;diese Unterprogramm, das mit ei-
1E7B: 8A         377           TXA              ;nem Parallel Interface arbeitet,
1E7C: 48         378           PHA              ;schickt ein Zeichen an den
1E7D: 98         379           TYA              ;Drucker. Will man andere Schnitt-
1E7E: 48         380           PHA              ;stellen benutzen, so muss dieses
1E7F: AD 21 03   381           LDA  Asave       ;UP geaendert werden
1E82: 2C C1 C1   382  Prwait    BIT  Prstrb      ;ist Drucker bereit?
1E85: 30 FB      383           BMI  Prwait
1E87: 8D 90 C0   384           STA  Prout       ;wenn ja, sende ein Zeichen
1E8A: 68         385           PLA
1E8B: A8         386           TAY
1E8C: 68         387           PLA
1E8D: AA         388           TAX
1E8E: AD 21 03   389           LDA  Asave
1E91: 60         390           RTS
                 391
                 392  ************************************************************
                 393
1E92: A9 0D      394  Initline  LDA  #CR         ;neue Zeile initialisieren
1E94: 20 78 1E   395           JSR  Print
1E97: A9 1B      396           LDA  #ESC        ;das ESC-Zeichen geht den Graphik-
1E99: 20 78 1E   397           JSR  Print       ;steuerzeichen immer vorraus
1E9C: AD 00 03   398           LDA  Control     ;Steuerzeichen fuer Graphikmodus
1E9F: 20 78 1E   399           JSR  Print
1EA2: AD 01 03   400           LDA  Control+1
1EA5: F0 03      401           BEQ  Initlnl
1EA7: 20 78 1E   402           JSR  Print
1EAA: AD 02 03   403  Initlnl   LDA  Bytes       ;Anzahl der Graphikbytes, zuerst
1EAD: 20 78 1E   404           JSR  Print       ;low, dann high
1EB0: AD 03 03   405           LDA  Bytes+1
1EB3: 20 78 1E   406           JSR  Print
1EB6: AD 0E 03   407           LDA  Rfactor
1EB9: 8D 27 03   408           STA  Rcounter
1EBC: 60         409           RTS
                 410
                 411  ************************************************************
                 412
1EBD: AD 10 03   413  Sendbits  LDA  Rdistnce    ;Zwischenraum zwischen 2 oder mehr
1EC0: AA         414           TAX              ;Bildern pro Zeile
1EC1: AD 11 03   415           LDA  Rdistnce+1
1EC4: A8         416           TAY
1EC5: D0 03      417           BNE  Sendthem
1EC7: 8A         418           TXA
1EC8: F0 0B      419           BEQ  Exitsend
                 420
1ECA: A9 00      421  Sendthem  LDA  #$00        ;Zwischenraum wird mit $00 ge-
1ECC: 20 78 1E   422  Sendl     JSR  Print       ;fuellt
1ECF: CA         423           DEX
1ED0: D0 FA      424           BNE  Sendl
1ED2: 88         425           DEY
1ED3: 10 F7      426           BPL  Sendl
1ED5: 60         427  Exitsend RTS
                 428
                 429  ************************************************************
                 430
1ED6: 48         431  Linesp    PHA              ;Inhalt des Akkus * 1/216" Zeilen-
1ED7: A9 1B      432           LDA  #ESC        ;vorschub
1ED9: 20 78 1E   433           JSR  Print
1EDC: A9 33      434           LDA  #'3'
1EDE: 20 78 1E   435           JSR  Print
1EE1: 68         436           PLA
1EE2: 20 78 1E   437           JSR  Print
1EE5: 60         438           RTS
                 439
                 440  ************************************************************
                 441
1EE6: A9 0D      442  CRLF      LDA  #CR         ;neue Zeile
```

PRINTHIRES Hex-Dump

```
$1C00: A9 0B 8D F6 03 A9 1C 8D
$1C08: F7 03 60 AD 0C 03 D0 07
$1C10: A9 20 8D 0C 03 D0 05 A9
$1C18: 40 8D 0C 03 AD 0A 03 D0
$1C20: 03 20 04 1F AD 08 03 F0
$1C28: 03 4C 76 1D A9 17 20 D6
$1C30: 1E AC 16 03 8C 20 03 A9
$1C38: 01 8D 26 03 20 5E 1C AD
$1C40: 26 03 CD 1C 03 F0 06 EE
$1C48: 26 03 4C 3C 1C AD 20 03
$1C50: 18 69 08 8D 20 03 CD 18
$1C58: 03 90 DC 4C 99 1F 20 E6
$1C60: 1E 20 8F 1C 20 BD 1E CE
$1C68: 27 03 D0 F5 20 4A 1D 20
$1C70: F4 1E AE 04 03 E0 01 F0
$1C78: 07 E0 02 D0 11 20 4A 1D
$1C80: 20 92 1E 20 8F 1C 20 BD
$1C88: 1E CE 27 03 D0 F5 60 AD
$1C90: 12 03 8D 25 03 AC 13 03
$1C98: 8C 22 03 AD 20 03 8D 24
$1CA0: 03 A9 08 8D 23 03 AD 24
$1CA8: 03 20 76 1F B1 1C AE 25
$1CB0: 03 4A CA 10 FC 2E 21 03
$1CB8: CE 23 03 F0 15 AD 24 03
$1CC0: CD 18 03 F0 05 EE 24 03
$1CC8: D0 DC 0E 21 03 CE 23 03
$1CD0: D0 F8 A0 08 AD 21 03 4A
$1CD8: 08 AE 1C 03 8E 29 03 AE
$1CE0: 1C 03 8E 23 03 A2 00 3E
$1CE8: 00 02 E8 CE 23 03 D0 F7
$1CF0: 28 08 CE 29 03 D0 E8 28
$1CF8: 88 D0 DC AE 26 03 8E 24
$1D00: 03 A0 08 18 AE 1C 03 CA
$1D08: 7E 00 02 CA 10 FA 2A 88
$1D10: D0 F1 CE 24 03 D0 EA 4D
$1D18: 06 03 AE 1A 03 20 78 1E
$1D20: CA D0 FA AC 22 03 CC 15
$1D28: 03 D0 09 AD 25 03 CD 14
$1D30: 03 D0 01 60 EE 25 03 AD
$1D38: 25 03 C9 07 F0 03 4C 98
$1D40: 1C C8 A9 00 8D 25 03 4C
$1D48: 98 1C AD 04 03 C9 02 D0
$1D50: 0E 20 92 1E 20 8F 1C 20
$1D58: BD 1E CE 27 03 D0 F5 60
$1D60: AD 04 03 C9 02 D0 0E 20
$1D68: 92 1E 20 D1 1D 20 BD 1E
$1D70: CE 27 03 D0 F5 60 A9 14
$1D78: 20 D6 1E AC 15 03 8C 22
$1D80: 03 A9 01 8D 26 03 20 E6
$1D88: 1E 20 D1 1D 20 BD 1E CE
$1D90: 27 03 D0 F5 20 60 1D 20
$1D98: F4 1E AE 04 03 E0 01 F0
$1DA0: 07 E0 02 D0 11 20 60 1D
$1DA8: 20 92 1E 20 D1 1D 20 BD
$1DB0: 1E CE 27 03 D0 F5 AD 26
$1DB8: 03 CD 1A 03 F0 06 EE 26
$1DC0: 03 4C 86 1D AD 22 03 A8
$1DC8: 88 CD 13 03 D0 B0 4C 99
$1DD0: 1F AD 16 03 8D 20 03 20
$1DD8: 76 1F AC 22 03 B1 1C 8D
$1DE0: 21 03 CC 15 03 D0 16 A9
$1DE8: 06 38 ED 14 03 F0 0E AA
$1DF0: E8 A9 FF 4A CA D0 FC 2D
$1DF8: 21 03 8D 21 03 AD 21 03
$1E00: AC 13 03 CC 22 03 D0 0E
$1E08: AE 12 03 F0 09 A9 FF 0A
$1E10: CA D0 FC 2D 21 03 A0 07
$1E18: 4A 08 AE 1A 03 8E 28 03
$1E20: AE 1A 03 8E 23 03 A2 00
$1E28: 3E 00 02 E8 CE 23 03 D0
$1E30: F7 28 08 CE 28 03 D0 E8
$1E38: 28 88 D0 DC AE 26 03 8E
$1E40: 24 03 A0 07 18 AE 1A 03
$1E48: CA 7E 00 02 CA 10 FA 2A
$1E50: 88 D0 F1 CE 24 03 D0 EA
$1E58: 4D 06 03 29 7F AC 1C 03
$1E60: 20 78 1E 88 D0 FA AE 20
$1E68: 03 EE 20 03 AD 20 03 EC
$1E70: 18 03 B0 03 4C D7 1D 60
$1E78: 8D 21 03 8A 48 98 48 AD
$1E80: 21 03 2C C1 C1 30 FB 8D
$1E88: 90 C0 68 A8 68 AA AD 21
$1E90: 03 60 A9 0D 20 78 1E A9
$1E98: 1B 20 78 1E AD 00 03 20
```

```
1EE8: 20 78 1E  443          JSR  Print
1EEB: A9 0A     444          LDA  #LF
1EED: 20 78 1E  445          JSR  Print
1EF0: 20 92 1E  446          JSR  Initline
1EF3: 60        447          RTS
                448
                449  ************************************************************
                450
1EF4: A9 1B     451  LS1      LDA  #ESC       ;1/216" Zeilenvorschub
1EF6: 20 78 1E  452          JSR  Print
1EF9: A9 4A     453          LDA  #'J'
1EFB: 20 78 1E  454          JSR  Print
1EFE: A9 01     455          LDA  #$01
1F00: 20 78 1E  456          JSR  Print
1F03: 60        457          RTS
                458
                459  ************************************************************
                460
1F04: A0 00     461  Frame    LDY  #$00       ;Dieses UP malt einen Rahmen um
1F06: AD 0C 03  462          LDA  Page       ;das Bild (zum besseren Ausschnei-
1F09: 84 1E     463          STY  Frameadr   ;den)
1F0B: 85 1F     464          STA  Frameadr+1
1F0D: A9 FF     465          LDA  #$FF
1F0F: 4D 06 03  466          EOR  XOR
1F12: 91 1E     467  Frame1   STA  (Frameadr),Y
1F14: C8        468          INY
1F15: C0 28     469          CPY  #$28
1F17: 90 F9     470          BCC  Frame1
1F19: A9 BF     471          LDA  #191
1F1B: 20 76 1F  472          JSR  Bascalc
1F1E: A0 00     473          LDY  #$00
1F20: A9 FF     474          LDA  #$FF
1F22: 4D 06 03  475          EOR  XOR
1F25: 91 1C     476  Frame2   STA  (Base),Y
1F27: C8        477          INY
1F28: C0 28     478          CPY  #$28
1F2A: 90 F9     479          BCC  Frame2
1F2C: A2 00     480          LDX  #$00
1F2E: 8A        481  Frame3   TXA
1F2F: 20 76 1F  482          JSR  Bascalc
1F32: A0 00     483          LDY  #$00
1F34: B1 1C     484          LDA  (Base),Y
1F36: 8D 21 03  485          STA  Asave
1F39: A9 01     486          LDA  #$01
1F3B: 4D 06 03  487          EOR  XOR
1F3E: 30 05     488          BMI  Frame4     ;negativ
1F40: 0D 21 03  489          ORA  Asave
1F43: D0 03     490          BNE  Frame5     ;unbedingter Sprung
1F45: 2D 21 03  491  Frame4   AND  Asave
1F48: 8D 21 03  492  Frame5   STA  Asave
1F4B: AD 21 03  493          LDA  Asave
1F4E: 91 1C     494          STA  (Base),Y
1F50: A0 27     495          LDY  #$27
1F52: B1 1C     496          LDA  (Base),Y
1F54: 8D 21 03  497          STA  Asave
1F57: A9 40     498          LDA  #$40
1F59: 4D 06 03  499          EOR  XOR
1F5C: C9 40     500          CMP  #$40
1F5E: D0 05     501          BNE  Frame6     ;negativ
1F60: 0D 21 03  502          ORA  Asave
1F63: D0 03     503          BNE  Frame7
1F65: 2D 21 03  504  Frame6   AND  Asave
1F68: 8D 21 03  505  Frame7   STA  Asave
1F6B: AD 21 03  506          LDA  Asave
1F6E: 91 1C     507          STA  (Base),Y
1F70: E8        508          INX
1F71: E0 C0     509          CPX  #192
1F73: 90 B9     510          BCC  Frame3
1F75: 60        511          RTS
                512
                513  ************************************************************
                514
1F76: 48        515  Bascalc  PHA             ;Berechnet die Basisadresse der
1F77: 29 C0     516          AND  #$C0       ;Zeile, die im Akku uebergeben
1F79: 85 1C     517          STA  Base       ;wurde und modifiziert sie ent-
1F7B: 4A        518          LSR             ;sprechend der 1. oder 2. Graphik-
1F7C: 4A        519          LSR             ;seite
1F7D: 05 1C     520          ORA  Base
1F7F: 85 1C     521          STA  Base
1F81: 68        522          PLA
1F82: 85 1D     523          STA  Base+1
1F84: 0A        524          ASL
1F85: 0A        525          ASL
1F86: 0A        526          ASL
1F87: 26 1D     527          ROL  Base+1
1F89: 0A        528          ASL
1F8A: 26 1D     529          ROL  Base+1
1F8C: 0A        530          ASL
1F8D: 66 1C     531          ROR  Base
```

```
1F8F: A5 1D     532           LDA  Base+1
1F91: 29 1F     533           AND  #$1F
1F93: 0D 0C 03  534           ORA  Page
1F96: 85 1D     535           STA  Base+1
1F98: 60        536           RTS
                537
                538  ************************************************************
                539
1F99: A9 1B     540  Quit     LDA  #ESC        ;Zeilenabstand auf 12 Bits
1F9B: 20 78 1E  541           JSR  Print
1F9E: A9 32     542           LDA  #'2'
1FA0: 20 78 1E  543           JSR  Print
1FA3: A9 0A     544           LDA  #LF         ;2 mal Zeilenvorschub
1FA5: 20 78 1E  545           JSR  Print
1FA8: A9 0A     546           LDA  #LF
1FAA: 20 78 1E  547           JSR  Print
                548
1FAD: 60        549           RTS              ;zurueck zu BASIC
                550
                551  ************************************************************
```

```
1000 ST = 768: REM START
1010 CO = ST + 00: REM CONTROL CHARACTERS
1020 BY = ST + 02: REM BYTES
1030 IM = ST + 04: REM IMPRESSION
1040 XO = ST + 06: REM EXCLUSIVE OR
1050 DI = ST + 08: REM DIRECTION
1060 FR = ST + 10: REM FRAME
1070 PA = ST + 12: REM PAGE
1080 RF = ST + 14: REM REPEAT FACTOR
1090 RD = ST + 16: REM REPEAT DISTANCE
1100 LE = ST + 18: REM LEFT
1110 RI = ST + 20: REM RIGHT
1120 TP = ST + 22: REM TOP
1130 BO = ST + 24: REM BOTTOM
1140 XF = ST + 26: REM X FACTOR
1150 YF = ST + 28: REM Y FACTOR
1170  TEXT : HOME
1180  PRINT "Bild speichern ? (j/n)";: GET A$
1190  IF (A$ <  > "Y") AND (A$ <  > "y") AND (A$ <  > "J")
     AND (A$ <  > "j") GOTO 1220
1200  PRINT
1210  PRINT CHR$ (4) "BSAVE PICTURE,A$2000,L$2000"
1220  HOME : VTAB 22: INVERSE : PRINT "GRAPHIK HARDCOPY
     FUER EPSON DRUCKER": NORMAL
1230  PRINT CHR$ (4) "BRUN PRINTHIRES"
1240  HOME : VTAB (22)
1250  PRINT "Graphik schon im Speicher ? (j/n)";
1260  GET A$
1270  IF (A$ = "N") OR (A$ = "n") GOTO 1470
1280  IF (A$ = "Y") OR (A$ = "y") OR (A$ = "J") OR (A$ =
     "j") GOTO 1590
1290  GOTO 1250
1300  POKE  - 16300,0: REM LOW PAGE
1310  POKE  - 16301,0: REM MIXED MODE
1320  POKE  - 16297,0: REM HIRES MODE
1330  POKE  - 16304,0: REM GRAPHICS MODE
1340  HOME : VTAB (22)
1345  FOR I = ST TO ST + 29: POKE I,0: NEXT : REM CLEAR
     ALL PARAMETERS
1350  GOSUB 2000: REM SET DEFAULT PARAMETERS
1360  PRINT "Grundeinstellung ? (j/n)";: GET A$
1370  IF (A$ = "Y") OR (A$ = "y") OR (A$ = "J") OR (A$ =
     "j") GOTO 1860
1380  TEXT
1390  GOSUB 2150
1400  GOSUB 2340
1410  GOSUB 2420
1420  GOSUB 2490
1430  GOSUB 2560
1440  GOSUB 2630
1450  GOSUB 2700
1460  GOTO 1860
1470  HOME : VTAB (22)
1480  PRINT "Diskette mit Graphikfiles einschieben": PRINT
     "und Taste druecken";
1490  GET A$
1500  TEXT
1510  PRINT
1520  PRINT CHR$ (4) "CATALOG"
1530  PRINT
1540  INPUT "Welche Graphik laden ? ";A$
1550  IF A$ = "" THEN A$ = "PICTURE"
1560  HGR
1570  HOME : VTAB (22)
```

```
$1EA0: 78 1E AD 01 03 F0 03 20
$1EA8: 78 1E AD 02 03 20 78 1E
$1EB0: AD 03 03 20 78 1E AD 0E
$1EB8: 03 8D 27 03 60 AD 10 03
$1EC0: AA AD 11 03 A8 D0 03 8A
$1EC8: F0 0B A9 00 20 78 1E CA
$1ED0: D0 FA 88 10 F7 60 48 A9
$1ED8: 1B 20 78 1E A9 33 20 78
$1EE0: 1E 68 20 78 1E 60 A9 0D
$1EE8: 20 78 1E A9 0A 20 78 1E
$1EF0: 20 92 1E 60 A9 1B 20 78
$1EF8: 1E A9 4A 20 78 1E A9 01
$1F00: 20 78 1E 60 A0 00 AD 0C
$1F08: 03 84 1E 85 1F A9 FF 4D
$1F10: 06 03 91 1E C8 C0 28 90
$1F18: F9 A9 BF 20 76 1F A0 00
$1F20: A9 FF 4D 06 03 91 1C C8
$1F28: C0 28 90 F9 A2 00 8A 20
$1F30: 76 1F A0 00 B1 1C 8D 21
$1F38: 03 A9 01 4D 06 03 30 05
$1F40: 0D 21 03 D0 03 2D 21 03
$1F48: 8D 21 03 AD 21 03 91 1C
$1F50: A0 27 B1 1C 8D 21 03 A9
$1F58: 40 4D 06 03 C9 40 D0 05
$1F60: 0D 21 03 D0 03 2D 21 03
$1F68: 8D 21 03 AD 21 03 91 1C
$1F70: E8 E0 C0 90 B9 60 48 29
$1F78: C0 85 1C 4A 4A 05 1C 85
$1F80: 1C 68 85 1D 0A 0A 0A 26
$1F88: 1D 0A 26 1D 0A 66 1C A5
$1F90: 1D 29 1F 0D 0C 03 85 1D
$1F98: 60 A9 1B 20 78 1E A9 32
$1FA0: 20 78 1E A9 0A 20 78 1E
$1FA8: A9 0A 20 78 1E 60 00 00
```

Softbreaker 1.0

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

von U. Stiehl

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

Softbreaker ist ein Assemblerprogramm, mit dessen Hilfe Programme, die sich von der 64K-Karte (= Extended 80 Column Card für den Apple IIe) starten lassen, unterbrochen, gespeichert, geladen und exakt an der Stelle der Unterbrechung fortgeführt werden können. Dadurch ist es auch möglich, Sicherungskopien von sogenannten kopiergeschützten Programmen herzustellen.

Mit Softbreaker unterbrochene Programme werden komplett, d. h. die ganzen 64K einschließlich Language Card, in nur ca. 11 Sekunden auf einer formatierten Diskette gesichert.

Gerätevoraussetzung: Apple IIe mit 64K-Karte

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

```
1580  PRINT CHR$ (4) "BLOAD";A$: REM LOW PAGE
1590  POKE  - 16300,0: REM LOW PAGE
1600  POKE  - 16301,0: REM MIXED MODE
1610  POKE  - 16297,0: REM HIRES MODE
1620  POKE  - 16304,0: REM GRAPHICS MODE
1630 P = 0
1640  POKE PA,0
1650  HOME : VTAB (22)
1660  PRINT "Ist dies das richtige Bild ? (j/n)";
1670  GET A$
1680  IF (A$ = "Y") OR (A$ = "y") OR (A$ = "J") OR (A$ =
     "j") GOTO 1300
1690  IF (A$ = "N") OR (A$ = "n") GOTO 1710
1700  GOTO 1630
1710  HOME : VTAB (22)
1720  PRINT "Wollen Sie die andere Graphikseite": PRINT
     "sehen ? (j/n)";
1730  GET A$
1740  IF (A$ = "Y") OR (A$ = "y") OR (A$ = "J") OR (A$ =
     "j") GOTO 1770
1750  IF (A$ = "N") OR (A$ = "n") GOTO 1840
1760  GOTO 1710
1770  POKE  - 16299,0: REM HIGH PAGE
1775  POKE  - 16302,0: REM GRAPHICS ONLY
1780 P = 1
1790  POKE PA,1
1800  GET A$
1810  PRINT
1820  POKE  - 16300,0: REM LOW PAGE
1825  POKE  - 16301,0: REM MIXED MODE
1830  GOTO 1650
1840  POKE  - 16303,0: REM RESET TEXT
1850  GOTO 1470: REM TRY AGAIN
1860  POKE  - 16302,0: REM  GRAPHICS ONLY
1870  IF P <  > 0 THEN  POKE  - 16299,0
1880  POKE  - 16297,0
1890  POKE  - 16304,0
1900  & : REM CALL MACHINE ROUTINE ($1C0B)
1910  TEXT
1920  HOME : VTAB (22)
1930  PRINT "Nochmal ? (j/n)";
1940  GET A$
1950  IF (A$ = "Y") OR (A$ = "y") OR (A$ = "J") OR (A$ =
     "j") GOTO 1240
1960  IF (A$ = "N") OR (A$ = "n") GOTO 1980
1970  GOTO 1920
1980  HOME : END
1990  REM ********************
2000  POKE C0, ASC ("K")
2010  POKE BY,24: POKE BY + 1,1: REM 280 BITS PER
     HORIZONTAL LINE
2020  POKE RF,1
2030  POKE RI,6
2040  POKE RI + 1,39
2050  POKE B0,191
2060  POKE XF,1
2070  POKE YF,1
2080  RETURN
2090  REM ********************
2100  PRINT.: PRINT "Welches Kommando: ";
2110  GET C$: IF (C$ = " ") OR (C$ =  CHR$ (13)) THEN C$ =
     "l"
2120  IF  VAL (C$) = 0 THEN C$ = "l"
2130  RETURN
2140  REM ********************
2150  HOME
2160  PRINT "<1> Normal density": PRINT
2170  PRINT "<2> Dual density": PRINT
2180  PRINT "<3> Double speed, dual density": PRINT
2190  PRINT "<4> Quadruple density": PRINT
2200  PRINT "<5> CRT graphics": PRINT
2210  PRINT "<6> Plotter graphics": PRINT
2220  PRINT "<7> CRT graphics II": PRINT
2230  POKE C0, ASC ("*")
2240  GOSUB 2100
2250  ON  VAL (C$) GOTO 2260,2270,2280,2290,2300,2310,2320
2260  POKE C0, ASC ("K"): RETURN
2270  POKE C0, ASC ("L"): RETURN
2280  POKE C0, ASC ("Y"): RETURN
2290  POKE C0, ASC ("Z"): RETURN
2300  POKE C0 + 1,4: RETURN
2310  POKE C0 + 1,5: RETURN
2320  POKE C0 + 1,6: RETURN
2330  REM ********************
2340  HOME
2350  PRINT "<1> Single strike": PRINT
2360  PRINT "<2> Double strike": PRINT
2370  PRINT "<3> Quadruple strike": PRINT
```

```
2380  GOSUB 2100
2390  POKE IM, VAL (C$) - 1
2400  RETURN
2410  REM ********************
2420  HOME
2430  PRINT "<1> Positiv": PRINT
2440  PRINT "<2> Negativ": PRINT
2450  GOSUB 2100
2460  IF C$ = "2" THEN  POKE XO,255
2470  RETURN
2480  REM ********************
2490  HOME
2500  PRINT "<1> Horizontal": PRINT
2510  PRINT "<2> Vertikal": PRINT
2520  GOSUB 2100
2530  POKE DI, VAL (C$) - 1
2540  RETURN
2550  REM ********************
2560  HOME
2570  PRINT "<1> Rahmen": PRINT
2580  PRINT "<2> Kein Rahmen": PRINT
2590  GOSUB 2100
2600  POKE FR, VAL (C$) - 1
2610  RETURN
2620  REM ********************
2630  HOME
2640  PRINT "<1> Seite 1": PRINT
2650  PRINT "<2> Seite 2": PRINT
2660  GOSUB 2100
2670  POKE PA, VAL (C$) - 1
2680  RETURN
2690  REM ********************
2700  HOME
2710  INPUT "Wiederholungsfaktor : ";I
2720 I =  ABS (I): IF I > 255 THEN I = 1
2730  IF I = 0 THEN I = 1
2740  POKE RF,I
2750  INPUT "Wiederholungsdistanz: ";I
2760 I =  ABS (I): IF I > 255 THEN I = 0
2770  POKE RD,I
2780  INPUT "Linker Rand          : ";I
2790 I =  ABS (I): IF I > 279 THEN I = 0
2800  POKE LE + 1, INT (I / 7)
2810  POKE LE,I -  INT (I / 7) * 7
2820  INPUT "Rechter Rand         : ";I
2830 I =  ABS (I): IF I > 279 THEN I = 279
2840  POKE RI + 1, INT (I / 7)
2850  POKE RI,I -  INT (I / 7) * 7
2860  INPUT "Oberer Rand          : ";I
2870 I =  ABS (I): IF I > 191 THEN I = 0
2880  POKE TP,I
2890  INPUT "Unterer Rand         : ";I
2900 I =  ABS (I): IF I > 191 THEN I = 191
2910  POKE BO,I
2920  INPUT "X-Faktor             : ";I
2930 I =  ABS (I): IF I > 31 THEN I = 1
2940  IF I = 0 THEN I = 1
2950  POKE XF,I
2960  INPUT "Y-Faktor             : ";I
2970 I =  ABS (I): IF I > 31 THEN I = 1
2980  IF I = 0 THEN I = 1
2990  POKE YF,I
3000  REM ********************
3010  IF  PEEK (DI) = 0 THEN BT =  PEEK (XF) * ( PEEK (RI)
     +  PEEK (RI + 1) * 7 -  PEEK (LE) -  PEEK (LE + 1) *
     7 + 1)
3020  IF  PEEK (DI) = 1 THEN BT =  PEEK (YF) * ( PEEK (BO)
     +  PEEK (BO + 1) * 7 -  PEEK (TP) -  PEEK (TP + 1) *
     7 + 1)
3030 BT = (BT +  PEEK (RD)) *  PEEK (RF)
3040  POKE BY + 1, INT (BT / 256)
3050  POKE BY,BT -  INT (BT / 256) * 256
3060  RETURN
```

Die aktuelle Fachzeitschrift für

Chemiker, Physiker, Mediziner, Ingenieure und Naturwissenschaftler

INFORMATIONEN über NEUE PRODUKTE

AKTUELLE NACHRICHTEN

ANKÜNDIGUNGEN von VERANSTALTUNGEN
mit Schwerpunkt auf Fortbildungsveranstaltungen

REVIEWS

PROGRAMME
Programme für Laboranwendungen werden vollständig aufgelistet und erläutert

WISSENSCHAFTLICHE ORIGINALBEITRÄGE
und Kurzmitteilungen über den Einsatz von Computern im Labor-Anwenderbereich

HARDWARE- und SOFTWARELÖSUNGEN

SERIEN
z. Zt. laufen Serien über Interfacing, Personal Computer im Labor, Assemblersprache FORTH, gebräuchliche Techniken der Datenanalyse

BUCHBESPRECHUNGEN

Bestellcoupon ausschneiden und einsenden an:

CAL – Computer-Anwendung im Labor, Vertriebsservice, im Weiher 10, 6900 Heidelberg

☐ Ja, ich möchte **1 Probeheft CAL**

☐ **Abonnement/s für 1985 zum Preis von DM 60,– incl. Mwst., zzgl. Versandspesen**

Die Bestellung kann ich innerhalb von 7 Tagen schriftlich beim Verlag widerrufen

Meine Anschrift:

Datum Unterschrift

Turbo-Pascal Schritt für Schritt

Dr. Ekkehard Kaier

Das TURBO-PASCAL von „Borland International" zeichnet sich durch einfache Handhabung, Schnelligkeit und einen günstigen Preis aus. Mit dem folgenden Artikel wenden wir uns insbesondere an den „PASCAL-Einsteiger".

1. Programmerstellung in 11 Schritten

Voraussetzung: CP/M gestartet, TURBO-Diskette Seite 1 in Laufwerk A: eingelegt, am Bildschirm steht das CP/M-Promptzeichen.

Schritt 1: Installation des Systems durch File TINST

– TINST eintippen, um das Installationsprogramm zu starten.
– S eintippen für S)creen Installation. Dann eine der im Menü gezeigten Nummern eingeben, z.B. 26 für Zenith.
– Eine C)ommand Installation ist bei der für den Apple II gelieferten Diskette nur erforderlich, wenn die Steuerkommandos für den Editor geändert werden sollen.
– Mit Q für Quit das Programm TINST verlassen.

Schritt 2: System aktivieren durch File TURBO

– TURBO eintippen, um das Minimal-System mit Editor, Compiler und Run-Time-Bibliothek zu laden.
– Frage „Include error messages (Y/N)?" mit „N" beantworten. Damit werden für Fehler nur deren Nummern, nicht aber deren Texte genannt (bei Antwort „Y" würde das File TURBO.MSG geladen, was 1.5 KBytes an Platz kostet).
– Am Bildschirm steht das folgende Hauptmenü:

```
Logged drive: A
Work file:
Main file:
Edit Compile Run Save
eXecute Dir Quit compiler Options

Text: 0 bytes (71F1-71F1)
Free: 22036 bytes (71F2-C806)
```

Angeboten werden die 11 Befehle L, W, M, E, C, R, S, X, D, Q und O. Durch Eintippen dieser Buchstaben werden sie ausgeführt.

Schritt 3: Workfile benennen durch Befehl W

– Im Hauptspeicher wird eine Arbeitsdatei (Workfile) verwaltet, auf die sich Eingeben (Edit bzw. E), Übersetzen (Compile bzw. C), Ausführen (Run bzw. R) und Speichern (Save bzw. S) beziehen. Das Workfile (es gibt nur *eines*) ist derzeit leer (siehe Hauptmenü oben).
– W für Workfile eintippen und auf die Frage „Work file name:" den Programmnamen ERSTPROG eingeben. Unser erstes Programm soll also ERSTPROG heißen.
– Die Ausgabe von „Loading A:ERSTPROG.PAS" und „New File" besagt, daß im Laufwerk A: derzeit *kein* File dieses Namens gefunden wurde und deshalb ein neues File im Hauptspeicher eingerichtet wird.
– Nach der Eingabe von Return erscheint wieder das Hauptmenü. Freier Speicherplatz nun 22035 Bytes (ein Byte wurde für den Eintrag des Programmnamens verbraucht).

Schritt 4: Programmtext eingeben durch Befehl E

– E eintippen: Der Editor wird aufgerufen, um die Eingabe unseres Programms zu überwachen.
– Oben am Bildschirmrand erscheint diese Statuszeile:
Line 1 Col 1 Insert Indent A:ERSTPROG.PAS
Nach Line bzw. Col wird die Zeilen- bzw. Spaltennummer des Cursors angezeigt. „Insert" heißt, daß unsere Texteingabe eingefügt wird (am Ende oder dazwischen). Tippen wir Ctrl-V, erscheint statt „Insert" der Hinweis „Overwrite": Unsere Texteingabe überschreibt dann den bisherigen Text. Erneutes Tippen von Ctrl-V wechselt wieder zu „Insert".
– Wir tippen den folgenden Programmtext mit 10 Zeilen ein:

```
PROGRAM ERSTPROG;
VAR
ZAHL: INTEGER;
BEGIN
WRITELN ('Welche Zahl?');
READLN (ZAHL);
ZAHL:=ZAHL+5;
WRITE ('Erhöhte Zahl: ');
WRITELN (ZAHL)
END.
```

Wichtig: „;" am Zeilenende, einfache statt doppelte Anführungszeichen sowie Punkt

nach END. Am Ende jeder Zeile Return tippen.
– Abschließend verlassen wir den Editor durch Eintippen von Ctrl-K Ctrl-D. Am Bildschirm steht wieder das Hauptmenü (ggf. nochmals Return tippen).
– Ein Tip: Falls ein Apple II benutzt wird, bei dem die Umlaute ä, ö, ü und das ß über Ctrl-Tasten erzeugt werden, sind diese Ctrl-Tasten zur Steuerung des Editors natürlich gesperrt.
Oft zählt dazu auch das Ctrl-K für die „eckige Klammer". In diesem Fall müssen Sie über das Installationsprogramm TINST die Kommandos neu festlegen (z.B. Ctrl-H Ctrl-D zum Verlassen des Editors anstelle von Ctrl-K Ctrl-D).
– Unten im Hauptmenü wird jetzt angezeigt, daß sich der im Hauptspeicher freie Speicherplatz vermindert hat:

Text: 166 bytes (71F1-7297)
Free: 21870 bytes (7298-C806)

Schritt 5: Workfile auf Diskette speichern durch Befehl S

– Bei Stromausfall wäre alle Arbeit umsonst. Durch Eintippen des Befehls S speichern wir den Inhalt des Workfiles unter seinem Namen ERSTPROG auf Diskette ab.
– Die Ausgabe „Saving A:ERSTPROG-.PAS" besagt, daß das Programm ERSTPROG als Pascal-Quellenprogramm (deshalb die Dateitypbezeichnung PAS) in Laufwerk A: abgespeichert worden ist.
Das Quellenprogramm nennt man auch Quellcode, Textfile oder Sourcefile bzw. Source.
– Am Bildschirm steht wieder das Hauptmenü.

Schritt 6: „Memory to Memory"-Übersetzung durch Befehl C

– Das als Quellcode im Hauptspeicher (genauer: im Workfile) stehende Programm ERSTPROG muß nun in einen Objektcode übersetzt bzw. compiliert werden. Erst danach kann das Programm ausgeführt werden.
– Wir tippen den Befehl C ein, und der Compiler als Übersetzungsprogramm meldet sich wie folgt:

Compiling
10 lines
Code: 125 bytes (7299-7316)
Free: 22762 bytes (7316-CC00)
Data: 6 bytes (CC00-CC06)

Es wurden also alle 10 Programmzeilen fehlerlos übersetzt. Die Angaben in Klammern geben die entsprechenden Adressen im RAM in hexadezimaler Schreibweise wieder.
– Im Hauptspeicher befindet sich nun neben dem Quell- auch das übersetzte Objektprogramm (auch Objektcode, Zielprogramm oder Maschinencode genannt). Deswegen die Bezeichnung „Memory to Memory"-Übersetzung (Memory für Hauptspeicher).

Schritt 7: Programmausführung durch Befehl R

– Nach Eingabe von R (für Run) wird der übersetzte Objektcode zur Ausführung gebracht:
Running
Welche Zahl?
1000
Erhöhte Zahl: 1005
– Wir geben nochmals R ein:
Running
Welche Zahl?
2
Erhöhte Zahl: 7
– Das Programm kann also beliebig oft mittels R ausgeführt werden.
– Findet der Befehl R)un kein Übersetzer-Programm, so leitet er automatisch eine Übersetzung (vgl. Schritt 6) ein.

Schritt 8: Inhaltsverzeichnis durch Befehl D

– Nach Eingabe von D (für Directory) wird folgendes Inhaltsverzeichnis der auf Diskette gespeicherten Files ausgegeben:

TURBO OVR
TURBOMSG OVR
TINST COM
TINSTMSG OVR
TURBO COM
TINST DTA
ERROR DOC
TLIST COM
ERSTPROG PAS
ERSTPROG BAK
Bytes Remaining on A: 39K

– COM bedeutet „COMmand File", OVR „OVeRlay", DTA „DaTA". Das Programm ERSTPROG ist zweimal gespeichert: mit der Filetyp-Bezeichnung PAS für „PAScal-Quellcode" und BAK für „BAcK-up-Version des Quellenprogramms". Nach jeder Eingabe des Befehls S (für Save) wird das aktuelle Workfile als „...PAS" und das bisher auf Diskette abgelegte Workfile zur Sicherheit als „...BAK" gespeichert.
– Das Inhaltsverzeichnis zeigt uns, daß das übersetzte Workfile (es trägt den Namen ERSTPROG.COM) noch *nicht* auf Diskette gesichert worden ist. Der Befehl S)ave speichert somit nur das Quellprogramm.

Schritt 9: Compiler-Option ändern durch Befehl O

– Nach Eintippen des Befehls O (für „compiler Options") erhalten wir folgendes Menü
compile → Memory
Com-file
cHn-file
Find run-time error Quit
mit den 5 Options-Befehlen M, C, H, F und Q.

– Der Pfeil nach „compile" weist auf „Memory": Das übersetzte Programm wird stets im Hauptspeicher abgelegt und durch den Befehl R)un dort aufgerufen. Man nennt diese Betriebsart „Alles im RAM", „In Memory" oder „Memory to Memory" und benutzt sie während der Programmerstellung (siehe Schritt 6).
– Wir geben den Befehl C ein: Damit wird das übersetzte Programm – also das Maschinenprogramm als Objektcode – vom Compiler automatisch auf Diskette gespeichert, und zwar unter dem Namen des Workfiles mit der Typbezeichnung COM. Man benutzt diese Betriebsart „Memory to Diskette" nur dann, wenn alle Testläufe fehlerfrei abgeschlossen sind.
– Die Meldung
Start address: 1076 (min 1D76)
End address: C953 (max CC06)
informiert über Start- und Endadresse des compilierten Programms.
– Mit Q kehren wir wieder ins Hauptmenü zurück.

Schritt 10: „Memory to Diskette"-Übersetzung durch Befehl C

– Durch Eingabe von C (für Compile) übersetzen wir das Workfile erneut. Die Compiler-Meldung
Compiling A:ERSTPROG.COM
10 lines
Code: 125 bytes (1D76-1DF3)
Free: 43738 bytes (1DF3-C8CD)
Data: 134 bytes (C8CD-C953)
besagt, daß nun (anders als bei der „Alles im RAM"-Übersetzung von Schritt 6) der

erzeugte Code auf Laufwerk A: unter dem Namen ERSTPROG.COM gespeichert wurde.
– Automatische Rückkehr ins Hauptmenü.
– Zur Kontrolle lassen wir uns mit D das Directory zeigen. Drei „Versionen" unseres Programms ERSTPROG erscheinen:
A: ERSTPROG PAS: ERSTPROG BAK: ERSTPROG COM
Bytes Remaining on A: 31 K
– ERSTPROG.PAS wurde zuletzt mittels S)ave gespeichert und ERSTPROG.BAK durch das vorangegangene S)ave. ERSTPROG.COM hingegen wurde durch den Befehl C)ompile (mit Com-Option) auf Diskette gespeichert.

Schritt 11: Programm kopieren in CP/M-Ebene

– Mit Q verlassen wir das TURBO-System, um mit dem Befehl PIP das soeben erstellte Programm auf eine Anwenderdiskette zu kopieren.
– Mittels PIP B:=A:ERSTPROG.PAS kopieren wir das Quellprogramm von der Systemdiskette in Laufwerk A: auf eine (zuvor durch FORMAT leer formatierte) Anwenderdiskette.
Mittels PIP B:=A:ERSTPROG.COM kopieren wir den Objektcode.
– Mittels ERA A:ERSTPROG.* löschen wir alls drei Versionen unseres ersten Programms auf der TURBO-Systemdiskette, um Platz zu schaffen für weitere Entwicklungen.
Ein Löschen von Files ist im TURBO-System selbst (noch?) nicht möglich.

Nach dem Durchlaufen der Schritte 1 bis 11 liegt unser erstes Anwenderprogramm namens ERSTPROG getestet auf Diskette vor. Soll ein weiteres Programm erstellt werden, kehren wir zu Schritt 3 zurück, um das Workfile neu zu benennen.

2. TURBO-PASCAL und UCSD-PASCAL

Das „Apple II Pascal 1.1" bzw. das „UCSD p System IV.0" sind die auf dem Apple II mit Abstand am weitesten verbreiteten PASCAL-Systeme. Im folgenden stellen wir die wesentlichen Unterschiede zum TURBO-PASCAL gegenüber:

a) Betriebssystem:
TURBO wird in verschiedenen Versionen angeboten. Auf Apple II arbeitet es unter CP/M 2.2. Eine Z-80-Karte ist somit erforderlich. UCSD beinhaltet ein eigenes Betriebssystem mit abgeschlossener Programmierumgebung.

b) Übersetzung der Programmiersprache:
TURBO hat einen Native-Code-Compiler, der den Programmtext als Quellcode in direkten Maschinencode übersetzt. Deshalb die unterschiedlichen TURBO-Versionen für Z-80, CP/M 86, MS-DOS usw. UCSD dagegen erzeugt aus dem Quellcode (Textfile genannt) als Zwischenform den p-Code (Codefile genannt), der erst während der Ausführung interpretiert wird. P-Code kann als Maschinencode für eine virtuelle Maschine aufgefaßt werden; der Apple II wird nur als Gast-Computer (sog. Host) angesehen.

c) Systemumfang:
Das TURBO-System findet auf einer (z.B. für IBM PC) bzw. auf zwei Disketten (z.B. Z-80-Version für Apple II) Platz. Das UCSD-System umfaßt drei oder mehr Disketten (z.B. SYSTEM1:, SYSTEM2:, CODE: und UTILITY: beim UCSD IV.0).

d) Anzahl der Diskettenlaufwerke:
TURBO kann mit einem Laufwerk betrieben werden. Das (Minimal-) System mit Compiler, Editor und Run-Time-Bibliothek beansprucht nur ungefähr 28 KBytes (File TORBO.COM). Für UCSD sind zwei Laufwerke unbedingt erforderlich.

e) Editor:
Der TURBO-Editor orientiert sich an der Wordstar-Bedienung. Die Editor-Steuertasten können jedoch leicht angepaßt werden: Falls man z.B. das von Applesoft-BASIC her gewohnte Ctrl-I für „nach oben" beibehalten möchte, kann man dies über das Installationsprogramm TINST leicht (da menügesteuert) tun (das Ctrl-E vom Wordstar in Ctrl-I austauschen). Der UCSD-Editor arbeitet weitgehend ohne Ctrl-Taste (Ausnahme: Ctrl-C für „Ende").

f) Dateien auf Diskette:
TURBO kann Dateien (Files) auf Diskette erzeugen, nicht aber löschen. Zum Löschen muß TURBO mit Q verlassen werden, um dann auf der CP/M-Ebene die entsprechenden Files zu löschen und dann wieder ins TURBO-System zurückzukehren. In UCSD können Dateien erzeugt und gelöscht werden (über den Filer).

g) Anweisungen von PASCAL:
Beide Sprachen gehen über das Standard-PASCAL (Jensen/Wirth) hinaus und ähneln sich im Befehlsvorrat (z.B. Datentyp STRING) weitgehend. UCSD kennt GET und PUT für den Dateizugriff, während TURBO dafür die erweiterten Funktionen READ und WRITE vorsieht.

h) Arbeitsgeschwindigkeit:
Da TURBO echten Maschinencode erzeugt, arbeitet es wesentlich schneller als UCSD, das den erzeugten p-Code noch interpretieren muß.

Die Gegenüberstellung zeigt, daß TURBO-PASCAL zumindest für den Einsteiger und Preisbewußten geeignet ist, PASCAL zu erlernen.

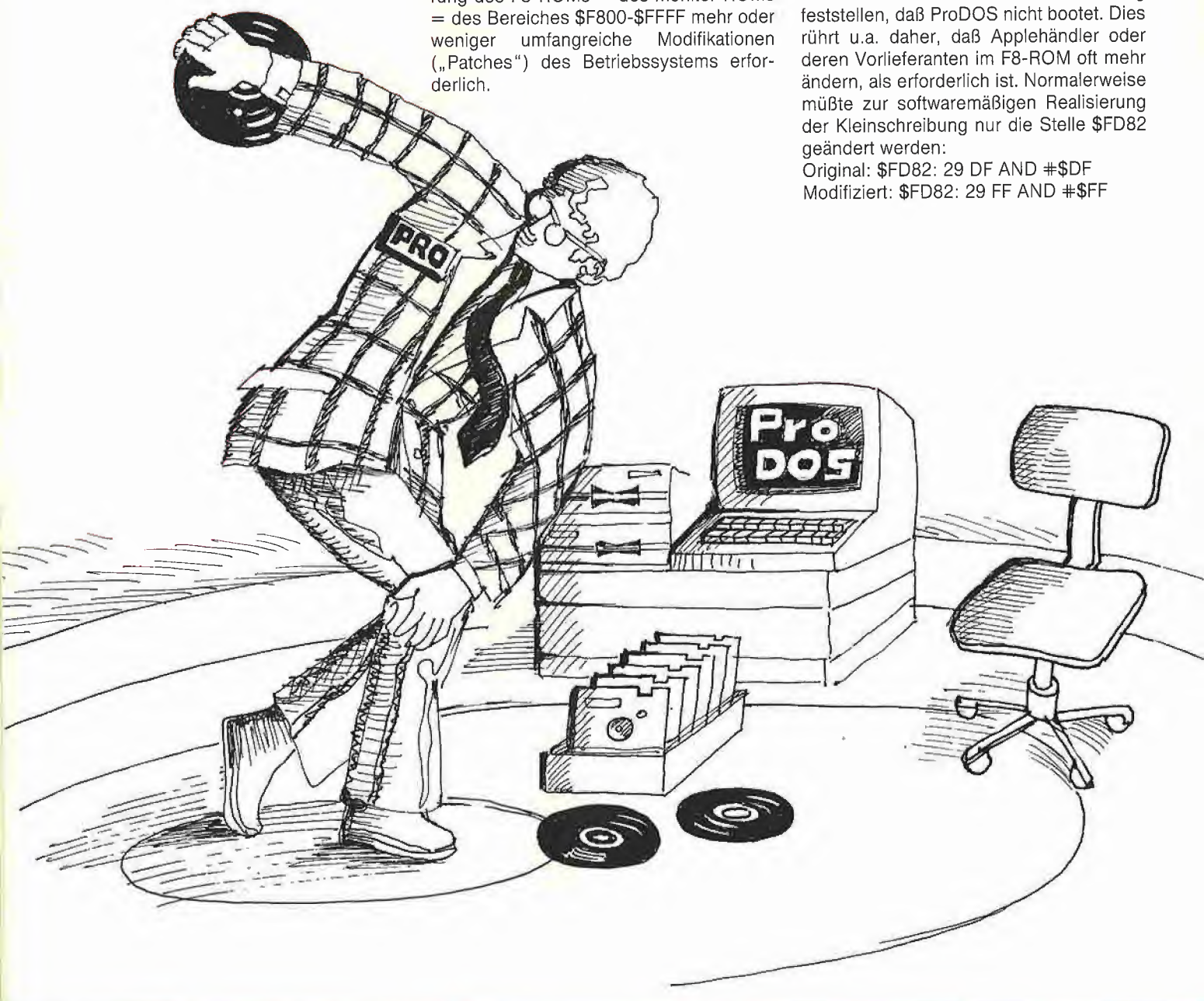
3. TURBO-PASCAL und BASIC

BASIC ist für die meisten die Einstiegsprache. Werden die Programme umfangreicher, wünscht man sich PASCAL: Das selbstdokumentierende Listing und der modulare, strukturierte Aufbau erleichtern die Problemlösung ganz wesentlich. Daß dieser Wunsch oft unerfüllt bleibt, liegt weniger an der Programmiersprache selbst als an der umständlicheren Handhabung des Systems:
Das BASIC-Programm tippt man einfach ein, um es nach Testlauf und Korrektur auf Diskette zu speichern. Der BASIC-Interpreter „übersetzt" das Programm dabei bei jedem Lauf neu.
Zur Eingabe des PASCAL-Programms hingegen muß man einen Editor als Überwachungsprogramm laden. Später muß man einen Compiler als Übersetzungsprogramm aktivieren, der aus dem Programmtext einen Programmcode in Maschinensprache erzeugt. Dann erst kann das Programm ausgeführt werden. Tritt nun ein Fehler auf (bei der Übersetzung und/oder der Ausführung), muß man wieder den Editor laden, um den Programmtext zu korrigieren. Diese Arbeitsfolge „Editieren - Compilieren - Ausführen" und das damit verbundene Aufrufen von Systemprogrammen ist bei vielen PASCAL-Systemen sehr kompliziert (z.B. bei PASCAL MT+), bei nur wenigen Systemen relativ schnell erlernbar (z.B. bei UCSD-PASCAL). Bei TURBO-PASCAL dagegen ist der Umgang mit dem System fast so einfach wie mit Applesoft-BASIC oder MBASIC unter CP/M. Zudem kommt man mit *einem* Laufwerk aus, da auf der Diskette das gesamte TURBO-System mit Compiler, Editor und Workfile (Programm als unübersetzter Text sowie als übersetzter Code) Platz findet. Aus diesen Gründen macht es TURBO-PASCAL auch dem zögernden BASIC-Anhänger leicht, mit PASCAL zu arbeiten.

ProDOS-Patch für geänderte F8-ROMs

Das unveränderte ProDOS-Betriebssystem 1.0.1 läuft nur auf den Originalgeräten Apple II Plus, Apple IIe und Apple IIc. Bei Apple-Nachbauten sind je nach Änderung des F8-ROMs = des Monitor-ROMs = des Bereiches $F800-$FFFF mehr oder weniger umfangreiche Modifikationen („Patches") des Betriebssystems erforderlich.

Wer sich hingegen z.B. in seinen *Original* Apple II Plus einen Kleinschreibumrüstsatz mit entsprechend geändertem F8-ROM hat einbauen lassen, wird unter Umständen ebenfalls zu seiner Überraschung feststellen, daß ProDOS nicht bootet. Dies rührt u.a. daher, daß Applehändler oder deren Vorlieferanten im F8-ROM oft mehr ändern, als erforderlich ist. Normalerweise müßte zur softwaremäßigen Realisierung der Kleinschreibung nur die Stelle $FD82 geändert werden:
Original: $FD82: 29 DF AND #$DF
Modifiziert: $FD82: 29 FF AND #$FF



Dieser Patch, der sich auch im Apple IIe ROM befindet und deshalb nicht überprüft wird, bewirkt, daß über die Tastatur eingegebene Kleinbuchstaben *nicht* in Großbuchstaben umgewandelt werden. ProDOS prüft u.a. das Versionsbyte bei Speicherstelle $FBB3 (enthält $06 für Apple IIe und IIc sowie $EA für II Plus) sowie den Namen „APPLE“ einschließlich Klammern ab $FB09. Im letzteren Fall wären z.B. zulässig: „APPLE“, „Apple“, „ApPLE“, also auch gemischte Groß- und Kleinschreibung, nicht dagegen „EXPL1“ o.ä., wie man es oft in illegalen Nachbauten findet.

Wer besagte Schwierigkeiten auf einem Apple II Plus mit Kleinschreibumrüstsatz hat, der modifiziere zunächst das BASIC.SYSTEM auf einem Apple IIe wie folgt:

1. BLOAD BASIC.SYSTEM, A$2000, TSYS
2. CALL -151
3. 2054: A9 62 8D 98 BF
4. Ctrl-C
5. UNLOCK BASIC.SYSTEM
6. BSAVE BASIC.SYSTEM, A$2000, L$2800, TSYS
7. LOCK BASIC.SYSTEM

Die hexadezimale Folge A9 62 8D 98 BF besagt LDA #$62 STA $BF98. $62 steht für Apple II Plus mit 64K RAM und 80-Zeichenkarte.

Falls das BASIC.SYSTEM gleich gar nicht eingeladen wird, muß auch das später in der Language Card befindliche PRODOS gepatcht werden:

1. BLOAD PRODOS, A$2000, TSYS
2. CALL -151
3. 23B4: A2 EA EA
4. 2659: EA EA
5. Ctrl-C
6. UNLOCK PRODOS
7. BSAVE PRODOS, A$2000, L$3C00, TSYS
8. LOCK PRODOS

Der Patch bei $23B4 ersetzt LDA $FBB3 durch LDX #$EA, wobei $EA das Erkennungsbyte für den Apple II Plus darstellt. Der Patch bei $2659 legt die Prüfroutine für den Firmennamen „APPLE“ lahm.

Normalerweise würde man zur Ermittlung seines rechtmäßigen Firmennamens den String „APPLE“ im Programm „PRODOS“ ablegen und dann mit der entsprechenden F8-ROM-Stelle ab $FB09 direkt vergleichen, z.B. durch CMP $FB09, X unter Berücksichtigung der möglichen Kleinschreibung von „Apple“ statt „APPLE“. Statt dessen greift der Programmierer der Firma Apple zu folgenden Maßnahmen:

a) Zunächst wird das Teilprogramm $2401-$2447 in die Zero-Page ab $0080 geschoben.

b) Dann werden in die Speicherstellen $000A-$000B die Werte $0A und $00 gepokt, also so:
000A: 0A 00

c) Nun wird die sich ursprünglich ab $243A befindliche Subtraktionsroutine aufgerufen, die $0500 von $000A mit Underflow abzieht, was $FB09 ergibt.

d) Schließlich wird ab $263D mit LDA ($0A), Y der String ab $FB09 geladen und aufgrund undurchsichtiger Operationen geprüft.

Wie gesagt, *normalerweise*. Daß man zu derart pathologisch erscheinenden Manipulationen greifen muß, um seinen rechtmäßigen Firmennamen vor Nachbauern zu schützen, zeigt, daß die Zeiten alles andere als normal sind. Um dies besser zu verstehen, muß man Beispiele aus anderen Bereichen des Urheber- und Namensrechts zitieren. Nehmen wir an, Goethe hätte in panischer Angst gelebt, daß Schiller Goethes „Faust“ unter Schillers Namen hätte veröffentlichen können. Was hätte er dann in seiner krankhaften Phobie tun können? Seinen Namen chiffriert auf den „Faust“ drucken lassen? Oder überhaupt den ganzen „Faust“ chiffrieren? All dies mutet wenig normal an und belegt auf das anschaulichste, daß die Mikrocomputer-Industrie noch lange nicht erwachsen geworden ist. Nachbauer, die das Wort „APPLE“ durch „EXPL1“ o.ä. ersetzen, wecken die Erinnerung an die Kolonialgeschichte von dem Neger auf der Baumwollplantage, der einen Zylinder auf das Glasauge seines Vorarbeiters legte, damit er von ihm nicht gesehen werden konnte. Offenbar fehlt vielen Nachbauern die Einsicht, daß der Inhalt des F8-ROMs, falls er ein urheberrechtlich schutzfähiges Werk darstellt, nicht seine Schutzfähigkeit dadurch verliert, daß man den Namen „APPLE“ entfernt, wie ja auch Goethes „Faust“ weiterhin Goethes „Faust“ bleibt, auch wenn man „GOETHE“ auskratzt oder durch „Exemplar 1“ ersetzt...

us

Patch für PRODOS.INIT 80 Spuren

Um mit „PRODOS“ eine Diskette formatieren zu können, gibt es zwei Möglichkeiten:

a) den „FILER“ von Apple oder
b) das Programm „PRODOS.INIT“ von U.Stiehl verwenden.

Das Programm unter Punkt (a) hat leider den Nachteil, daß nur 35-Track-Disketten formatiert werden können. Das Programm unter Punkt (b) – gelistet in „Apple ProDOS, Band 1“ – kann zwar Disketten bis zu 255 Tracks initialisieren, konfiguriert diese Disketten jedoch ab 64 Spuren nicht mehr einwandfrei. Es hat sich nämlich in der Routine zur Umrechnung der Tracks in Blocks ein kleiner Fehler eingeschlichen. Verwendet man die Original-Routine aus dem genannten Buch, dann kann man, wie auch dort erwähnt, keine Disketten mit mehr als 64 Spuren oder 512 Blocks anlegen. Das nun unten aufgeführte Assemblerprogramm ist jetzt in der Lage, eine Diskette mit 2040 Blocks (1020K) anzulegen. Vorausgesetzt, die angeschlossene Hardware macht mit. Ein Laufwerk vom Typ TEAC FD-55E/55F ist somit sehr leicht auf eine Kapazität von 640 Blocks (320K) zu bringen. Selbstverständlich lassen sich auch andere 35/40/80-Spur-Laufwerke hiermit formatieren.

Wenn man das kleine Basicprogramm zur Initialisierung seiner Disketten verwendet, ist es sehr leicht, sich eine Diskette nach den eigenen Wünschen zu konfigurieren. Man denke nur an das Programm „MOUSEPAINT“, wo man auf der Bootdiskette nur noch zusätzlich 2 erstellte Bilder abspeichern kann. Kopiert man hier die Originaldiskette auf eine 320K-Disk, bleiben einem immer noch 394 Blocks.

Im folgenden Teil wird nun erklärt, wie das Programm „PRODOS.INIT“ geändert werden muß:

a) Man lädt „PRODOS.INIT“ mit dem Befehl BLOAD PRODOS.INIT, A$8000.
b) An den Speicherstellen $8814 und $8980 wird vom Monitor aus (nach CALL-151) folgende Änderung eingegeben:

```
8814: 4C 80 89 28 43 29 20 20
881C: 48 47 48 00 00 00 00 00
8824: 00 00 00 00 00 00 00 00
882C: 00 00 00
8980: AD 06 80 8D 2D 88 A9 00
8988: 8D 2E 88 A2 04 0E 2E 88
8990: CA F0 0B 0E 2D 88 90 F5
8998: 2E 2E 88 4C 90 89 60
```

c) BSAVE PRODOS.INIT, A$8000, L$0E00

Nach dieser Prozedur wird noch das entsprechende Applesoft-Programm eingegeben. Der Source-Code des Assembler-Patch-Programms sowie das Basic-Programm sind nachfolgend gelistet.

Vergleichende Tests haben ergeben, daß sowohl bei den Apple Disk II wie auch bei den TEAC FD-55F Laufwerken die maximale Einlesegeschwindigkeit bei ca. 8400 Bytes pro Sekunde liegt.

Anmerkung des Redakteurs: Ich freue mich immer, wenn jemand einen Bug (oder auch mehrere) in meinen Programmen findet, zeigt dies doch immerhin, daß man an den Programmen interessiert ist, sonst würde man sich nicht die Mühe machen, den Fehler zu suchen. Ein bekannter amerikanischer Autor – der Name tut hier nichts zur Sache – hatte einmal im Scherz gesagt, er würde absichtlich einige Fehler stehenlassen „...to keep the readers happy and alert“.
Was das PRODOS.INIT-Programm anlangt, so handelt es sich im übrigen um eine teils gekürzte und teils erweiterte RWTS des alten DOS 3.3. Als ich diese zu einer ProDOS-Formatierungsroutine umschrieb, verfügte ich nur über die alten DISK-II-Laufwerke, bei denen ich nur mit der 36. Spur experimentieren konnte. Als ich dann bei einem Apple-Händler an 80-Track-Cumana-Laufwerken einen endgültigen Test durchführte, zeigte sich der Fehler in der Multiplikationsroutine. Diese hatte ich seinerzeit „auf die schnelle“ gestrickt, ohne sie gesondert auszutesten. Damit habe ich selbst gegen meine Programmierer-Maxime verstoßen, die mir sonst heilig ist. Diese besagt: „Teste jedes Teil-Modul für sich, bevor Du alle Module zum Gesamtprogramm zusammenfügst“.

Hans-Georg Hüneke

```
100  D$ = CHR$ (4)
110  PRINT D$"BLOAD PRODOS.INIT"
120  REM
130  REM
140  REM
150  HOME
160  PRINT : PRINT : PRINT
170  INPUT "Wieviele Spuren ? ";TRACK$
180  IF  VAL (TRACK$) < 1 OR  VAL (TRACK$)
     > 255 THEN 460
190  PRINT : PRINT
200  INPUT "Welcher Slot ? ";S
210  IF S < 1 OR S > 7 THEN 460
220  PRINT : PRINT
230  INPUT "Welches Drive ? ";D
240  IF D < 1 OR D > 2 THEN 460
250  REM
260  REM
270  INIT = 32768: REM  $8000
280  SLOT = 32771: REM  $8003
290  DRIVE = 32772: REM $8004
300  TRACK = 32774: REM $8006
310  REM
320  POKE SLOT,S
330  POKE DRIVE,D
340  SELUNIT =  PEEK (49289 + S * 16 + D)
350  REM
360  REM
370  REM
380  POKE TRACK, VAL (TRACK$)
390  REM
400  UNITEIN =  PEEK (49289 + S * 16)
410  CALL INIT
420  END
430  REM
440  REM
450  REM
460  PRINT : PRINT
470  PRINT "Falsche Eingabe, bitte nochmal!"
480  FOR I = 1 TO 2000: NEXT
490  GOTO 150
```

```
                      1   ****************************************
                      2   *                                      *
                      3   *     TRACK/BLOCK - CONVERTER          *
                      4   *                                      *
                      5   *     von H.G.Hüneke 1984              *
                      6   *                                      *
                      7   * Diese Routine rechnet für das Pro-   *
                      8   * gramm 'PRODOS.INIT', aus dem Buch    *
                      9   * 'APPLE PRODOS für Aufsteiger',      *
                      10  * die entsprechenden Blöcke/Diskette   *
                      11  * aus, um die formatierte Diskette mit *
                      12  * der notwendigen Information in der   *
                      13  * Directory zu versorgen.              *
                      14  *                                      *
                      15  ****************************************
                      16
                      17
                      18  BLOCK    EQU  $8006
                      19
                      20           ORG  $8814
                      21  *
                      22  ****************************************
                      23  *   Sprung in die neue Block-Routine.  *
                      24  ****************************************
                      25
8814: 4C 80 89        26           JMP  CALCBLCK
                      27
8817: C8 FD EE        28           ASC  "Hüneke84"
881A: E5 EB E5 B8 B4
                      29
881F: 00 00 00        30           HEX  00000000   ;Alte Block-Ermittlungsroutine löschen.
8822: 00
8823: 00 00 00        31           HEX  00000000
8826: 00
8827: 00 00 00        32           HEX  000000000000
882A: 00 00 00
882D: 00 00           33  BLOCKLSB HEX  0000
                      34  BLOCKMSB EQU  BLOCKLSB+1
                      35
                      36           ORG  $8980
                      37  ****************************************
                      38  *   Initialisieren des Block-Zählers.  *
                      39  ****************************************
                      40
8980: AD 06 80        41  CALCBLCK LDA  BLOCK      ;Multiplikand mit den
8983: 8D 2D 88        42           STA  BLOCKLSB
8986: A9 00           43           LDA  #0         ;entsprechenden Spuren initialisieren.
8988: 8D 2E 88        44           STA  BLOCKMSB
898B: A2 04           45           LDX  #4         ;Schleifenzähler für die Multiplikation mit 8.
                      46
                      47  ****************************************
                      48  * Berechnen der gesamten Blöcke.       *
                      49  ****************************************
                      50
898D: 0E 2E 88        51  MULTMSB  ASL  BLOCKMSB   ;High-Byte mit 2 multiplizieren.
                      52  *
8990: CA              53  MULTLSB  DEX             ;Zähler für die Multiplikation mit 8.
8991: F0 0B           54           BEQ  MULT8END
8993: 0E 2D 88        55           ASL  BLOCKLSB   ;Low-Byte mit 2 multiplizieren.
8996: 90 F5           56           BCC  MULTMSB    ;Wenn kein übertrag, Multiplikation mit 2.
8998: 2E 2E 88        57           ROL  BLOCKMSB   ;Übertrag ins High-Byte übertragen und verhindern,
899B: 4C 90 89        58           JMP  MULTLSB    ;daß das High-Byte nochmals multipliziert wird.
                      59  *
899E: 60              60  MULT8END RTS
```

Wer ist der Schnellste?

Primzahlen-Preisausschreiben

Bei unserem Primzahlen-Wettbewerb können Sie diese Preise gewinnen:
1. Preis: DM 500,–
2. Preis: DM 250,–
3.–25. Preis: Freiabonnement von „Peeker“ für 1985.

Es wird die Aufgabe gestellt, die Primzahlen im Bereich von 2-8191 nach einem beliebigen Algorithmus in einer beliebigen Programmiersprache auf einem Apple II+/e/c in weniger als 0,45 Sekunden zu ermitteln. Der Schnellste erhält den ersten Preis. Mitmachen kann jeder Leser von „Peeker“, sei er Hobbyist oder Profi. Selbst die Apple-Mannschaft in München darf zeigen, wie schnell sie ist....

1. Sieb des Eratosthenes

Wohl jeder, der auf seinem Apple gelegentlich „aus Spaß an der Freude" programmiert, hat sich schon irgendwann einmal mit der Berechnung von Primzahlen befaßt. Darüber hinaus benutzen viele Computer-Zeitschriften diese Berechnung als „Benchmark", um die Leistungsfähigkeit unterschiedlicher Programmiersprachen zu testen. Deshalb findet man in der einschlägigen Fachliteratur eine Fülle von Algorithmen und Programmvarianten zur Ermittlung der Primzahlen innerhalb eines vorgegebenen Intervalls. Das bekannteste und vielleicht schnellste Verfahren ist das *Sieb des Eratosthenes*, benannt nach dem gleichnamigen griechischen Gelehrten, der um ca. 200 v. Chr. als Mathematiker, Dichter, Astronom und Bibliothekar in Alexandria wirkte. Wie viele geniale Entdeckungen ist das Sieb des Eratosthenes nicht nur extrem effizient, sondern auch ungewöhnlich einfach und deshalb für jeden Nicht-Mathematiker sofort einsichtig:

Primzahlen sind von 0 und 1 verschiedene natürliche Zahlen, die nur durch sich selbst ohne Rest geteilt werden können, also die Zahlen 2, 3, 5, 7, 11 usw. Die Zahl 2 nimmt insofern eine Sonderstellung ein, als sie die erste und damit kleinste Primzahl ist. Ferner ist die Zahl 2 die einzige gerade Primzahl. Nehmen wir an, wir wollten die Primzahlen für das Zahlenintervall 0-9 ermitteln. Da 0 und 1 per definitionem keine Primzahlen sind, brauchen wir uns nur auf das Intervall 2-9 zu beschränken. Die Eratosthenes-Methode geht dann nach folgendem, völlig formalistischem und damit quasi rein mechanischem Verfahren vor:

Schritt 1: Zunächst markieren wir alle Zahlen im Intervall 2-9 (Untergrenze bis Obergrenze) als potentielle Primzahlen, indem wir folgende Flag-Zuordnung vornehmen:

2 3 4 5 6 7 8 9
0 0 0 0 0 0 0 0
0 = Primzahl (potentiell)
1 = keine Primzahl

Im Verlauf der weiteren Erörterung soll gelten, daß Flag 0 = Primzahl und Flag 1 = Nicht-Primzahl bedeutet. Statt dieser willkürlich definierten Flags, die sich speziell für die computermäßige Aussiebung der Primzahlen eignen, könnte man auf dem Papier die später gefundenen Nicht-Primzahlen durchstreichen oder auf andere Weise markieren.

Schritt 2: Nunmehr initialisieren wir einen Zähler auf die erste Zahl des Zahlenstrahls, also auf die Untergrenze (= 2).

Schritt 3: Wir prüfen jetzt, ob die momentane Zahl (beim ersten Durchlauf Zahl 2) auf 0 gesetzt ist, d.h. als Flag = 0 hat. Wenn nein, gehen wir direkt zu Schritt 4. Wenn ja, haben wir eine Primzahl gefunden und können damit alle *Vielfachen* der momentanen Primzahl bis zur Obergrenze auf 1 = Nicht-Primzahlen setzen. Im ersten Durchlauf bezüglich der Zahl 2 sind dies die Zahlen 4, 6 und 8.

Schritt 4: Nunmehr erhöhen wir den Zähler um 1 und prüfen, ob der Zähler größer als die Obergrenze ist. Wenn nein, gehen wir zurück zu Schritt 3. Wenn ja, haben wir alle Primzahlen in dem vorgegebenen Intervall ermittelt.

Für den Zahlenstrahl 2-9 wären damit folgende Streichungen bzw. Flagsetzungen erforderlich:

2 3 4 5 6 7 8 9
0 0 1 0 1 0 1 0 = 2
0 0 1 0 1 0 1 1 = 3
0 0 1 0 1 0 1 1 = 5
0 0 1 0 1 0 1 1 = 7

Das obige Siebverfahren läßt sich auf verschiedene Weise verfeinern:

a) Man könnte darauf verzichten, vorab *alle* Zahlen auf 0 = potentielle Primzahlen zu setzen. Statt dessen könnte man gleich mit der Primzahl 2 beginnen, diese selbst auf 0 setzen und sodann alle Vielfachen von 2 auf 1 = Nicht-Primzahlen sowie alle *Nicht*-Vielfachen von 2 auf 0 = potentielle Primzahlen setzen.

b) Wie ersichtlich, finden bei dem Intervall 2-9 nach der Zahl 3 keine weiteren Streichungen mehr statt. Es läßt sich mathematisch beweisen, daß die Streichung nur bis zur Quadratwurzel der Obergrenze durchgeführt werden muß, also hier bis zur Quadratwurzel von 9 = 3. Damit reduziert sich die Anzahl der erneuten Streichungen von bereits gestrichenen Nicht-Primzahlen ganz erheblich.

c) Nehmen wir an, wir hätten gerade die Zahl 2 als Primzahl ermittelt und im Anschluß daran alle Vielfachen von 2 als Nicht-Primzahlen eliminiert. Nunmehr wären wir bei der Zahl 3 angelangt, die wir ebenfalls – da noch nicht gestrichen bzw. auf 1 gesetzt – als Primzahl erkennen. Wir könnten nunmehr alle Vielfachen von 3, d.h. 6 und 9 streichen. Da jedoch alle Vielfachen von 2, d.h. alle geraden Zahlen außer 2, bereits gestrichen sind, brauchen wir bei der Zahl 3 nicht alle geraden Vielfachen von 3, also z.B. nicht Zahl 6, zu streichen, da diese geraden Zahlen ja bereits gestrichen worden sind. Folglich können wir uns bei der Primzahl 3 statt der Streichung von 3 + (1 * 3), 3 + (2 * 3) usw. auf die Streichung von 3 + (1 * 3 * 2), 3 + (2 * 3 * 2) usw. beschränken. Bei dem Intervall 2-9 wäre damit nur die 9 zu markieren, da die 6 bereits zuvor durch 2 markiert worden ist.
Bei der Primzahl 5 könnten wir dasselbe Verfahren wie bei der Primzahl 3 anwenden, also 5 + (1 * 5 * 2) = 15, 5 + (2 * 5 * 2) = 25, 5 + (3 * 5 * 2) = 35, 5 + (4 * 5 * 2) = 45. Wie ersichtlich, wären jedoch die Zahlen 15 und 45 bereits mit der Primzahl 3 als deren Vielfache markiert worden. Folglich könnte man hier mit einem noch verzwickteren Algorithmus, der auf das Wissen der bereits abgehakten Primzah-

len 2 und 3 zurückgreift, die Zahlen 15 und 45 überspringen.

Das nachfolgende Applesoft-Programm berechnet die Primzahlen in dem Intervall 2-8191 nach dem Eratosthenes-Verfahren. Zur genaueren Zeitmessung erfolgen insgesamt 256 Durchläufe. Zu diesem Zweck wird die Speicherstelle 768 ($0300) hochgezählt, da vor jedem Durchlauf alle Variablen gelöscht werden.
In Zeile 110 wird das Programm per Tastendruck gestartet.
In Zeile 130 wird der Speicher gelöscht, dann in Zeile 140 die Quadratwurzel aus 8191 ermittelt (= 90,50414), die zur Vermeidung von Rundungsfehlern auf 91 erhöht wird.
In Zeile 150 wird der Primzahlen-Array P% dimensioniert, womit zugleich alle Werte = Flags auf 0 gesetzt werden. (Schritt 1)
Zeile 170-190 stellen die eigentliche Berechnung dar (Schritt 3). Dabei wird von X = 2 bis 91 hochgezählt (Schritt 2 und 4), wobei immer dann, wenn P%(X) = 0 ist, die Vielfachen von X bis zur Obergrenze H auf 1 = Nicht-Primzahlen gesetzt werden. Der Vergleich „IF X + X < H" = „IF S + S < H" in Zeile 180 könnte theoretisch entfallen.
In Zeile 210 wird nach jedem Durchlauf am Bildschirm ein „P" angezeigt. Nach 256 Durchläufen wird in Zeile 240 ein Piepston zur Zeitmessung ausgegeben. Danach werden zur Kontrolle alle ermittelten Primzahlen sowie am Schluß die Summe aller Primzahlen (insgesamt 1028 Zahlen) gelistet.

```
100 REM !INTEGER*
110 HOME : PRINT „PRIM-TEST": PRINT
    „STOPPUHR!": PRINT : GET X$
120 POKE 768,0: REM Zähler initialisieren
130 CLEAR : REM Loop
140 S = INT ( SQR (8191)) + 1
150 DIM P%(8191):H = 8191
160 REM Berechnen
170 FOR X = 2 TO S
180 IF P%(X) = 0 THEN IF X + X < H
    THEN FOR Y = X + X TO H
    STEP X:P%(Y) = 1: NEXT
190 NEXT
200 REM 256 Loops
210 PRINT „P";
220 L = PEEK (768):L = L + 1
230 IF L < 256 THEN POKE 768,L:
    GOTO 130
240 PRINT CHR$ (7): REM Piepston
250 REM Anzeigen
260 PRINT :A = 0
270 FOR X = 2 TO H: IF P%(X) = 0 THEN
    PRINT X,:A = A + 1
280 NEXT : PRINT : PRINT „ANZAHL: "A
290 REM 77 Sek. mit 1 MHz 6502
300 REM 10 Sek. mit Tasc 1 MHz 6502
310 REM 2.9 Sek. mit Tasc 3.5 MHz
    65C02C – beim Wettbewerb nicht
    erlaubt!
```

2. Wettbewerbsbedingungen

Allgemeine Aufgabenstellung: Es soll ein Primzahlen-Programm erstellt werden, das alle Primzahlen im Intervall von 2 bis 8191 in weniger als 0,45 Sekunden auf einem normalen, d.h. nicht „frisierten" Apple II+/IIe/IIc ermittelt.

1. Jeder kann sich an dem Preisausschreiben beteiligen (mit Ausnahme der Redaktion von „Peeker"), doch darf jeder nur eine einzige Lösung einsenden, die im übrigen alle 1028 Primzahlen von 2–8191 in weniger als 0,45 Sekunden, d.h. ab 0,44 abwärts, ermitteln muß. Um diese Zeit zu unterschreiten, muß man sich schon einiges einfallen lassen. Der Grund für diese hohe Programmgeschwindigkeit ist darin zu sehen, daß bereits sehr viele Eratosthenes-Varianten publiziert wurden, auf die ein Teilnehmer natürlich zurückgreifen könnte. Unseres Wissens ist jedoch noch niemals eine Variante veröffentlicht worden, die eine halbe Sekunde deutlich unterschreitet. Selbst das u.W. momentan schnellste Eratosthenes-Programm – erschienen in „Apple Assembler – Tips und Tricks" (Hüthig-Verlag) – bringt es nur auf 0,55 Sekunden. Durch Festlegung des Zeitlimits dürfte gewährleistet sein, daß nur selbstentwickelte Lösungen eingereicht werden. Unter Ausnutzung aller Assemblerkniffe läßt sich beispielsweise die sehr seriöse Programmvariante aus dem erwähnten Apple-Assembler-Buch ganz erheblich „tunen", wobei an dieser Stelle verschwiegen sei, welche Zeitverbesserungen realisierbar sind.

2. Die Primzahlen-Programme werden von uns auf einem normalen Apple IIe mit 1 MHz 6502-Prozessor und Disk-II-Laufwerken getestet. Damit scheiden andere und/oder schnellere Prozessoren aus. Erlaubt sind alle Programmiersprachen oder Kombinationen von Programmiersprachen (z.B. compiliertes Applesoft plus reines Assembler usw.) sowie alle denkbaren Programmiertricks (z.B. selbstmodifizierendes Programm usw.). Nicht erlaubt sind Tricks, die mit dem Primzahlenprogramm nichts zu tun haben. Beispielsweise könnte ein cleverer Programmierer auf den Trick verfallen, *vor* dem eigentlichen Programmstart eine Hires-Grafik zu zeigen, die etwa den Bereich $2000–$3FFF (= 8192 Bytes) ausblankt mit der Folge, daß man sich das Löschen des Primzahlen-Arrays sparen würde.

3. Die eingereichte Lösung – bootfähiger Objekt-Code + kommentierter Quell-Code – soll sich auf einer normalen 5 1/4 Zoll Diskette für Disk-II-Laufwerke befinden. Das Etikett ist mit der vollständigen Adresse sowie mit der selbst gestoppten Durchschnittszeit für 1 Durchlauf zu versehen. Alle eingereichten Disketten werden später wieder zurückgesandt.

4. Da nicht jeder über eine Hardware-Uhr verfügt, muß das Programm folgende Zeitmessung per Stoppuhr zulassen: Ein kurzes Menü soll 2 Optionen zulassen: A-Version mit 256 Durchläufen und B-Version mit 1024 Durchläufen. Die A-Version wird per Tastendruck gestartet (Stoppuhr läuft an), ruft dann das eigentliche Primzahlenprogramm 256 mal auf und wird mit einem Piepston beendet (Stoppuhr hält an). Der so ermittelte Wert (z.B. 97,1 oder 77,8) wird auf einen durch 0,5 teilbaren Wert abgerundet (z.B. auf 97,0 oder 77,5), dann durch 256 geteilt und schließlich auf die zweite Stelle nach dem Komma abgehackt (z.B. auf 0,37 oder 0,30). Die Fehlerquote beim Handstoppen von 256 Durchläufen (A-Version) liegt bei ca. +/- 0,5 Sekunden. Dies wären jedoch nur 0,5 : 256 = 0,001953125 = ca. 0,002 Sekunden, also ein praktisch bedeutungsloser Wert. Wir messen zunächst nach der A-Version. Sollte unser Meßwert von Ihrem selbstgemessenen zu Ihren Ungunsten abweichen, stoppen wir nach der B-Version. Nehmen wir an, jemand würde eine weltrekordverdächtige Zeit von 0,20 Sekunden für 1 Durchlauf nennen. Diese sei ermittelt worden durch 205,1, abgerundet auf 205,0, geteilt durch 1024 = 0,200195 = abgehackt auf 0,20 Sekunden. Nehmen wir ferner an, wir selbst hätten den total unwahrscheinlichen Wert von 207,6 – also 2,5 Sekunden mehr! – gestoppt, dann wären dies 207,6, abgerundet auf 207,5, geteilt durch 1024 = 0,202636 = abgehackt auf 0,20 Sekunden, also trotz extrem falscher Messung kein schlechterer Wert.

5. Das Primzahlenprogramm darf im voraus nur zwei Zahlen kennen, nämlich die Untergrenze 2 und die Obergrenze 8191. Es darf z.B. nicht im voraus die Quadratwurzel aus 8191 = 90.50414355 kennen, vielmehr muß diese, falls man das Eratosthenes-Verfahren benutzen will, eigens

errechnet werden. Ob dies durch die entsprechende Applesoft-Interpreter-Routine oder durch eine eigene Routine erfolgt, ist dabei gleichgültig. Das oben beschriebene Sieb des Eratosthenes dient im übrigen nur als Anregung. Jeder beliebige andere mathematische Algorithmus ist zulässig. Ferner ist jedes beliebige Wissen um die Primzahlen zulässig, z.B. daß gerade Zahlen keine Primzahlen sind. Sollte dieses Wissen jedoch die Kenntnis einer oder mehrerer Primzahlen voraussetzen (z.B. der Zahl 2), dann darf von diesem Wissen *erst dann* Gebrauch gemacht werden, wenn diese Primzahl(en) bereits zuvor ohne dieses Wissen ermittelt worden sind.

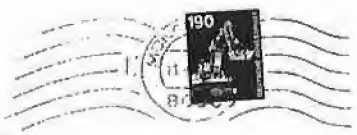
6. Das Primzahlenprogramm darf keinerlei Tabellen enthalten. Ganz plump wäre z.B., wenn alle Primzahlen bereits in verschlüsselter Form vorlägen und mit dem Programm als Array eingeladen würden. Jeder der 256 oder 1024 Durchläufe muß wie ein Ei dem anderen gleichen, d.h. in einem nachfolgenden Durchlauf darf nicht auf Werte zurückgegriffen werden, die in einem vorangehenden Durchlauf ermittelt wurden. Nach jedem Durchlauf müssen sich alle ermittelten Primzahlen und Nicht-Primzahlen im Bereich 2-8191 komplett im Speicher befinden. Ob diese Fließkomma-Flags, Integerflags, Byte-Flags, Nibble-Flags oder Bit-Flags sind, ist demgegenüber gleichgültig. Nehmen wir an, ein Assemblerprogramm würde den Speicherbereich $2000-$3FFF für die 8192 Zahlen reservieren, wobei etwa $2002 als Primzahl 2 auf 0 gesetzt würde ($2002–$2000 = $0002), $2004 auf 1 usw. Dann darf man ab dem zweiten Durchlauf nicht von dem Wissen Gebrauch machen, daß nunmehr bereits alle Primzahlen berechnet im Speicher vorliegen. Vielmehr muß jeder Durchlauf so tun, als enthielte der Speicher Zufallswerte, so daß in jedem Durchgang dieselbe Initialisierungsroutine aufgerufen werden muß.

7. **Einsendeschluß** ist der 15. November 1984. Richten Sie Ihre Lösungen an Dr. Alfred Hüthig Verlag, Redaktion Peeker, Postfach 102869, 6900 Heidelberg 1.

Von Lösungen, die hinsichtlich *aller* zeitkritischen Routinen *total* identisch sind, wird nur die zuerst eingesandte berücksichtigt. Die Programme der Gewinner des 1. und 2. Preises werden im November-Heft von „Peeker" veröffentlicht. Der Rechtsweg ist ausgeschlossen.

Besuchen Sie uns auf der Apple-Expo!

Zur Orgatechnik in Köln hat die Firma Apple eine komplette 5000 qm große Halle angemietet, in der wir mit einem eigenen Zeitschriften-, Bücher- und Software-Stand vertreten sein werden. Die Apple-Expo ist vom 25. bis 29. 10. 1984 täglich von 10.00 – 20.00 Uhr geöffnet und steht unter dem von Apple verkündeten Motto „Versuchen Sie nicht, eine Maschine zu werden". Indessen können wir nur schmunzelnd konstatieren, daß selbst die Firma Apple ihrer eigenen Maxime nicht gerecht wird, wie das bei uns eingegangene Kuvert bezeugt. Es hätte uns nicht gewundert, wenn der inliegende Brief unseren Chefredakteur mit „Sehr geehrter Herr Postfach" angeredet hätte...



Redaktion Peeker
Dr. Alfred Hüthig Verlag
z. H. Herrn Peeker
Chefredaktion

6900 Heidelberg

Apple Computer GmbH
Ingolstädter Str. 20
D-8000 München 45
Tel. (089) 35034-0

Hinweise für Autoren

Wir suchen laufend Beiträge: Erfahrungsberichte über kommerzielle Hardware- und Software-Produkte, Tips und Tricks für Anwender und Programmierer, gut dokumentierte Programme für Anfänger und Fortgeschrittene, hardwaretechnische Beiträge usw. Wir zahlen DM 200,- pro Druckseite, gleichviel ob Text oder Programm, wobei ein Beitrag mehrere Druckseiten oder gar mehrere Aufsatzfolgen umfassen kann. Es lohnt sich also, für uns zu schreiben.
Alle Textbeiträge einschließlich der Programmlistings werden von einem Apple IIe direkt in die Lichtsatzanlage unserer Druckerei übertragen. Im einzelnen sind deshalb folgende Richtlinien zu beachten:

Beiträge müssen sowohl als Papierausdruck wie auch als Diskette eingereicht werden. Zur Zeit können ausschließlich die klassischen 5 1/4 Zoll Disketten für Apple II/IIe/IIc angenommen werden. Berücksichtigt werden grundsätzlich nur Originalaufsätze und Originalprogramme. Es ist jedoch urheberrechtlich zulässig, aus Werken anderer Urheber zu zitieren. So können Sie z.B. Teile aus einem fremden Programm in Ihr eigenes übernehmen. In diesem Fall muß jedoch an geeigneter Stelle ein Zitathinweis erfolgen, etwa in der Form „Zeilen X-Y enthalten den Teil Soundso aus dem Programm ABC des Autors X, erschienen bei Firma Y, Ort und Jahr".
Wie im Verlagswesen üblich, übertragen Sie uns mit den eingereichten Aufsätzen und/oder Programmen das Recht der Vervielfältigung und Verbreitung in jeder Form. Speziell in bezug auf die Programme besagt dies, daß sie von uns nicht nur in Druckform, sondern auch in Diskettenform vertrieben werden, und zwar als Peeker-Sammeldisketten, die in unregelmäßiger Folge erscheinen und eine Auswahl der größeren Programme als Quell- und Objekt-Code enthalten. Versehen Sie Ihre Programme nicht mit dem amerikanischen Copyright-Vermerk, sondern schreiben Sie statt dessen nur „Von Name, Ort und Jahr".

1. Textbeiträge

a) Die Aufsatztexte sollten im Idealfall mit einem DOS-3.3-Textprogramm erstellt werden, das normale ASCII-Textfiles erzeugt, z.B. Applewriter II/IIe usw. Wordstar-Dateien im CP/M-Format sind ebenfalls möglich, doch müssen in diesem Fall die Texte endlos (ohne Silbentrennung) erfaßt werden.
b) Die Texte dürfen keinerlei Steuerzeichen für Matrixdrucker usw. enthalten. Überschriften dürfen nicht unterstrichen werden. Vielmehr brauchen lediglich im ausgedruckten Manuskript die Überschriften sowie darüber hinaus im Text selbst Hervorhebungen mit einem Farbstift o.ä. markiert zu werden. Die entsprechende Kodierung für die Satzanlage wird dann von uns übernommen.
c) Der Text muß grundsätzlich endlos erfaßt werden, d.h. ein Return darf nur am Absatzende und niemals am Zeilenende stehen, da der Satzrechner sonst keinen automatischen Umbruch mehr vornehmen kann. Vermeiden Sie darüber hinaus Einzüge zu Beginn eines Absatzes. Auch dies besorgt nämlich die Satzanlage automatisch.
d) Ausgangspunkt unserer Kodierung ist der deutsche ASCII-Zeichensatz, also mit Umlauten, ß usw. anstelle von eckigen und geschweiften Klammern. Wenn Sie – z.B. beim alten Apple II – keine Umlaute usw. eingeben können, rufen Sie uns an. Bei Texten, die *sowohl* die deutschen *als auch* die amerikanischen ASCII-Sonderzeichen enthalten sollen, z.B. bei Pascal-Aufsätzen, müssen die letzteren kodiert werden. Kodierung bedeutet, daß eine Buchstabenkombination, z.B. „§p" für griechisches π durch unser Transmitter-Programm in den entsprechenden Kode der Satzanlage konvertiert wird.
e) Vermeiden Sie unnötige Einrückungen und Kurztabellen. Ferner mischen Sie bitte nicht Aufsatztexte mit Programmtexten, da letztere gesondert in die Satzanlage übertragen werden. Größere Tabellen, die von uns konventionell gesetzt werden, sind auf gesonderten Textfiles zu speichern. Es genügt, wenn Sie senkrechte Trennlinien mit Kuli o.ä. andeuten. Schaubilder usw. sind stets willkommen. Hierzu genügen Bleistiftskizzen als Vorlagen für unsere Reinzeichner.
f) Achten Sie bitte auf Orthographie und Interpunktion. Deutsch ist immer noch eine wichtigere Sprache als Basic und Assembler!

2. Programmlistings

Beachten Sie, daß wir im Gegensatz zu anderen Computer-Zeitschriften Programmlistings nicht von den meist sehr häßlichen Printouts abphotographieren, sondern wie die Textbeiträge von Diskette in die Satzanlage übertragen, wobei eine besondere Schreibmaschinen-Lichtsatzschrift verwendet wird. Nur bei exotischen Programmiersprachen wählen wir z. Zt. noch den konventionellen Weg.

2.1. Basic-Programme

a) Applesoft-Programme usw. liefern Sie bitte als normale Programm-Files, die von uns für die Satzanlage aufbereitet werden.
b) Zeilennummern sollten stets dieselbe Stellenzahl haben, z.B. nur 3stellige Zeilennummern 100, 110, 120 o.ä., damit diese Zahlen von uns korrekt freigestellt werden können (RENUMBER anwenden).
c) REM's sollten der besseren Lesbarkeit halber möglichst in Groß-Klein-Buchstaben geschrieben werden.
d) Menü-Texte, REM's usw. müssen unbedingt in Deutsch sein. Englischsprachige Programme werden nicht angenommen. Englische Computer-Fachausdrücke sind demgegenüber selbstverständlich zulässig.

2.2. Assembler-Programme

a) Assembler-Programme sind als Source-Code und als Objekt-Code auf Diskette einzureichen. Wir bevorzugen den Assembler namens Merlin (= Big Mac) von G. Bredon. Verwendbar sind ferner Lisa 2.5, Toolkit-Assembler, ProDOS-Assembler sowie jeder andere Assembler, bei dem die Möglichkeit besteht, anstelle auf Papier auf Diskette zu assemblieren.
b) Der Source-Code sollte möglichst im 40-Z/Z-Modus geschrieben werden, damit die assemblierten Listings in der Zeitschrift 2spaltig gesetzt werden können. In der Kommentarspalte des Source-Codes sollten nur ganz kurze Bemerkungen stehen. Längere Kommentare schreibe man *über* die Befehle in reine Kommentarzeilen, wobei auch diese 40 Zeichen pro Zeile nicht überschreiten sollten.
c) Vermeiden Sie Makros und seltene Pseudo-Op-Codes, damit der Quell-Code ohne großen Aufwand in einen anderen Assembler konvertiert werden kann.

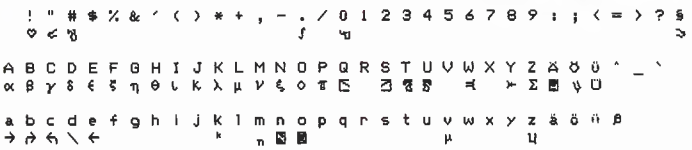
2.3. Sonstige Programme

Programme in selteneren Programmiersprachen sind willkommen, erfordern jedoch eine vorherige Absprache wegen der Konvertierungsmöglichkeit. Ggf. muß ein Schönschreibdrucker-Printout erstellt werden.

Individuelle Zeichensätze für Matrixdrucker

Der DMP-Charger der Firma Norbert Hunstig in Münster ist ein Zeichensatz-Generator-Programm, das in Pascal geschrieben wurde und zur Erstellung eigener Zeichensätze für den Apple Dot Matrix Printer, den Apple Imagewriter und den Epson FX 80 dient. Die neuen Zeichen sind Buchstaben, keine Grafik. Sie werden vom DMP Charger geladen und können danach von jedem beliebigen Programm (z. B. Applewriter) benutzt werden. Die Umschaltung erfolgt jeweils durch eine ESC-Sequenz. Es können bis zu 95 Zeichen umfassende Zeichensätze erstellt werden.

Die einzelnen Zeichen werden zum Editieren in 10facher Vergrößerung auf dem Bildschirm dargestellt. Dabei hat man jederzeit den Überblick über das zu bearbeitende Zeichen, d. h. man sieht auch, wie das jeweilige Zeichen auf dem Drucker aussieht. Die einzelnen Zeichen sind auch nach Erstellung eines Zeichensatzes korrigierbar. Ein Zeichensatz-Drucker ist nebenstehend abgebildet.



Bezugsquelle: Norbert Hunstig, Labor für Nachrichtentechnik, Nottulner Landweg 81, 4400 Münster, Tel. 02534/7449, Telex 892496

AFC-Tastatur mit Maus für Apple

Will der Benutzer bereits bestehende Apple-Programme mit Maus oder Rollball bedienen, so ist er gezwungen, ein zusätzliches Interface in seinen Rechner zu integrieren und vor allem seine Software anzupassen.
Das Ziel, ohne jegliche Anpassung, weder auf Hard- noch auf Softwareebene, jedem Benutzer ein bequemes, zeitgerechtes Eingabemedium zur Verfügung zu stellen, wurde durch die Verknüpfung intelligenter Tastatur und Maus erreicht. An die Tastaturen ALPHAKEY und OPERATOR der Firma AFC Computer, Köln, können Rollball oder Maus direkt angeschlossen werden (siehe Abbildung). Die Tastaturen stellen hier das intelligente Interface für beide Eingabemedien dar.
In jedem Programm wird die aktuelle Cursorposition mit den Cursortasten der Rechnertastatur bewegt. Wird nun eine an die erwähnten Tastaturen angeschlossene Maus zum direkten Cursorpositionieren bewegt, so werden die Maussignale von der Tastatur in wiederholte Cursorbefehle umgewandelt.
Ein an diese Tastaturen angeschlossener Rollball verhält sich ähnlich wie eine Maus, der einzige Unterschied liegt im wahrsten Sinne des Wortes auf oder in der Hand, da der Rollball stationär auf dem Tisch liegt und somit platzsparend dem Benutzer den gewünschten Komfort bietet.
Neben der vereinfachten Möglichkeit, den Cursor zu bewegen, können mit den zusätzlichen Tasten auf der Maus oder dem Rollball häufig benötigte Befehle abgerufen werden, ohne die Hand auf die Tastatur zurückführen zu müssen. Bei der Bearbeitung von Texten bietet sich z. B. das Umschalten auf Einfügen oder Überschreiben sowie das gezielte Löschen von Buchstaben oder Textteilen an.
Bezugsquelle: AFC Computer GmbH, Salmstr. 20, 5000 Köln 91, Tel. 0221/838000, Telex 8873254

Copy Killer

Der Copy Killer der Firma Mattes Computersysteme (früher Hamasoft) in Albstadt ist ein professionelles Kopierschutzsystem für den Apple II. Mit dem Copy Killer ist es möglich, normale 5 1/4-Zoll-Disketten vor „Raubkopierern" zu schützen. Zunächst wird eine Slave-Diskette angelegt. Auf diese Slave-Diskette wird dann die Original-Diskette kopiert.
Die so kopierte Diskette ist kopiergeschützt und läßt sich auch mit den bekannten Bit-Kopierprogrammen wie Nibbles Away, Locksmith usw. nicht mehr kopieren.
Besonderheit: Beim Kopieren auf eine Slave-Diskette kann eine Boot-Meldung mit eingegeben werden. Diese Boot-Meldung wird beim Booten der Diskette angezeigt. So kann z. B. jede Diskette individuell mit einer Serien-Nummer versehen werden.
Wir haben das Programm ausführlich getestet. Es besteht übrigens aus 2 in Applesoft geschriebenen Teilmodulen, die durch ein gemeinsames Hello-Programm aufgerufen werden und erstens der Initialisierung einer Leerdiskette im geschützten DOS 3.3-Format (Option CREATE) sowie zweitens der Übertragung der ungeschützten Programmvorlage auf die Leerdiskette dienen (Option TRANSFER). Beispielsweise ließ sich eine derart präparierte Programmdiskette mit Locksmith-Version 5.0 nicht mehr kopieren. U.a. arbeitet das Schutzverfahren mit stark geändertem Timing. Ferner sind auch die Sektoren-Erkennungs-Bytes geändert bzw. fehlen ganz, so daß von einem weitestgehend zuverlässigen Schutzverfahren gesprochen werden kann.
Als Nachteile des Copy Killers sind im einzelnen zu nennen, daß erstens nur DOS 3.3-Disketten, also z. B. keine ProDOS- oder Pascal-Disketten geschützt werden können, und daß zweitens ein Umschalten zwischen geschützter Programmdiskette und ungeschützter Datendiskette offenbar nicht möglich ist, so daß Datendisketten nicht konventionell mit z. B. COPYA gesichert werden können.

Bezugsquelle: Mattes Computersysteme (Hamasoft), Theophil-Wurm-Str. 2, Tel. 07432/13316, Telex 763317

Sony-Laufwerke von Ueding

Die Firma Ueding electronics, Holtewiese 2, 5750 Menden 1, liefert seit kurzem Sony-Laufwerke (siehe Abbildung) mit Patch-Diskette für DOS 3.3, CP/M 2.2, Pascal 1.1. Für ProDOS sind Patch-Programme in Vorbereitung. Die Laufwerke sind für den Apple II gedacht. Ein ausführlicher Testbericht ist für das November-Heft von „Peeker" geplant.

65C02 + Accelerator: Wer liefert was?

Der 65C02-Prozessor sowie die Accelerator-Karte, insbesondere die neue für den Apple IIe, sind z. Z. in Deutschland noch schwer erhältlich. Aufgrund einer Händlerumfrage nennen wir Ihnen in der Reihenfolge der Postleitzahlen nachfolgend Apple-Händler, die den 65C02 (1) und/oder die Accelerator-Karte (2) **ab Lager** liefern können:
– pandasoft Dr.-Ing. Eden, Uhlandstr. 195, D-1000 Berlin 12, (2)
– intermicro software & computer GmbH, Winterhuder Weg 48, D-2000 Hamburg 76, (1), (2)
– Hans Schröder Computersysteme GmbH, Föhrenstr. 19, 2800 Bremen 1, (1)
– UNITRONIC Vertriebs GmbH, Manskestr. 29, D-3160 Lehrte, (1)
– weidemann electronic, Postfach 48, D-5455 Rengsdorf, (1)
– computermarkt R. Sattler, Braunstr. 4, D-6120 Michelstadt, (2)
– Schöpp GmbH Telecommunication, Bahnstr. 79, D-6140 Bensheim, (2)
– GSG Geräte-System-Ges. mbH, Am Weissen Stein 1/3, D-6330 Wetzlar 26, (2)
– ORGASOFT GmbH Organisationsberratung + Software, Werner-von-Siemens-Str. 3, D-7730 VS-Villingen, (2)
– computerfachgeschäft GmbH, Maximiliansplatz 22, D-8000 München 2, (2)
– Mikrocomputer Vertriebsgesellschaft mbH MKV, Landwehrstr. 75, D-8000 München 2, (2)
– Hard- u. Software-Büro Dipl.-Kfm. Richard Zeitler, Holzfällerstr. 4, D-8400 Regensburg, (2)
– bürosysteme Heinz Ott GmbH, Steigbeetstr. 11, D-8500 Nürnberg 50, (2)
– target electronic Ing. O. Aistleitner, Reichstr. 123 a, A-6800 Feldkirch, (2)

Bidirektionaler Datenverkehr zwischen Apple und Brother-Typenradschreibmaschine

Die elektronische Typenradschreibmaschine Brother CE-50 mit speziellem bidirektionalem Interface für Apple II/IIe bietet die Firma interkom electronic in Isernhagen an.

Durch die Entwicklung maßgeschneiderter Hard- und Software für das Interface konnten Möglichkeiten geschaffen werden, die sonst nur Systeme deutlich höherer Preisklassen aufzeigen. So wird eine Vielzahl von zusätzlichen Sonderfunktionen erreicht, z. B. automatisches Unterstreichen, hoch- und tiefstehende Zahlen, Vorwärts- und Rückwärtstransport usw.

Durch die Möglichkeit des bidirektionalen Betriebs wird die Typenradschreibmaschine nicht nur zum Korrespondenzdrucker, sondern auch zur Eingabetastatur für den Apple.

Das System unterstützt alle gngigen Textverarbeitungsprogramme, die auf Apple- oder Apple-kompatiblen Rechnern laufen, außerdem CP/M, Pascal, Fortran und Applesoft (siehe Abbildung).

Bezugsquelle: intercom electronic, Kock & Mreches GmbH, Am Heisterholz 5, 3004 Isernhagen 4, Tel. 05139/87393, Telex 923827

Apple DOS 3.3

von Ulrich Stiehl
2. Aufl. 1984, 203 S., kart., DM 28,-
ISBN 3-7785-1049-5
Hüthig Verlag, Heidelberg

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.

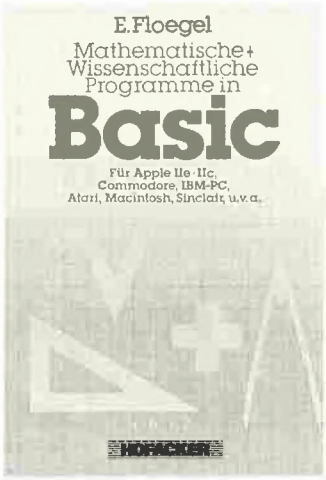
Aus dem Inhalt:

DOS für Benutzer von Anwenderprogrammen – DOS für Applesoft-Programmierer – DOS für Assembler-Programmierer – Catalog, TSL und VTOC – Fehlermeldungen – Vermischte Tips, Tricks und Patches – GETLN, RDKEY und COUT: Input-Output-Vektoren – DOS-Vektoren ab $03D0 – RTWS (Read-Write-Track-Sector)

Mathematische + Wissenschaftliche Programme in Basic

von Ekkehard Flögel
1984, 142 S., zahlr. Abb., kart., DM 29,80
ISBN 3-88963-102-9
Hofacker, Holzkirchen

Das vorliegende Buch enthält eine Sammlung von mathematischen Programmen, die größtenteils Lösungen praktischer Probleme darstellen. Die heutigen Heimcomputer machen es möglich, mathematische Aufgaben auf einfache Art zu lösen, die noch vor wenigen Jahren nur sehr schwerfällig zu lösen waren.
Die Auswahl der Programme ist natürlich nicht vollständig. Auf Statistikprogramme ist in diesem Buch verzichtet worden, da diese in „Statistik in BASIC" (ebenfalls Hofacker-Verlag) bereits enthalten sind.
Die Beispiele reichen von der einfachen Zinseszins- und Rentenrechnung bis zur Lösung eines Randwertproblems. Viel zu wenig ist bekannt, daß auch Rechnungen mit Polynomen, die heute vielfach noch ohne Computer durchgeführt werden, auch mit einem Mikrocomputer realisierbar sind.



Aus dem Inhalt:

Elementare Mathematik – Funktionen und Polynome – Komplexe Zahlen, Vektoren, Matrizen – Lineare Gleichungssysteme – Ausgleichrechnung – Numerische Integration von Differentialgleichungen – Program Evaluation and Review Technique (PERT) – Lösung des Dirchiletproblems durch Differenzverfahren – Lineare Regression – Lineare Planungsrechnung – Fourier Analyse – Algebra

Statistik in Basic

von Ekkehard Flögel
1984, 214 S., zahlr. Abb. und Tabellen, kart., DM 39,-
ISBN 3-88963-188-6
Hofacker, Holzkirchen

In den letzten Jahren ist es durch wachsende Verbreitung der Heimcomputer immer einfacher geworden, Daten zu sammeln und auszuwerten. Das vorliegende Buch enthält eine Sammlung von Statistik-Programmen, die in BASIC geschrieben sind, und zwar überwiegend in Applesoft sowie darüber hinaus in IBM-PC-BASIC.
Es wird gezeigt, wie Balkendiagramme gezeichnet, statistische Kennzahlen ermittelt oder statistische Prüfverfahren angewendet werden.

Zu jedem Programm ist ein Beispiel angegeben. Hier wurde vor allem mit Musteraufgaben aus der Literatur gerechnet, für die Ergebnisse vorlagen, die ohne Verwendung eines Rechners ermittelt wurden.

Die Theorie zu den Beispielen ist kurz gehalten. Dieses Buch soll kein Lehrbuch für Statistiker sein, sondern zeigen, wie statistische Berechnungen auf einfache Art in BASIC formuliert werden.

Aus dem Inhalt:

Darstellung der Daten – Grafische Darstellung von Daten – Statistische Kennzahlen – Wahrscheinlichkeitsrechnung – Markov-Analyse – Diskrete statistische Verteilungen – Die Normal Verteilung – Statistische Prüfverfahren – Parameterfreie Testverfahren – Varianzanalyse – Lineare Regression

– Die mehrfache lineare Regression – Autokorrelation – Nichtlineare Regression – Vorhersagen und Trendberechnungen – Statistik und Glücksspiel

Apple II leicht gemacht

von Joseph Kascmer
1984, 185 S., zahlr. Abb., kart., DM 28,-
ISBN 3-88745-031-0
Sybex-Verlag, Düsseldorf

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 drükken", sondern zumindest einige Details 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

ProDOS-Editor 1.0

Applesoft-Editor unter ProDOS-Betriebssystem

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1024-X

Mit diesem neuen Editor – übrigens der bislang einzige deutsche ProDOS-Editor – wird dem Applesoft-Programmierer ein Werkzeug zur effektiven Programmierung unter dem Betriebssystem ProDOS gegeben, denn die früheren Editoren sind allesamt unter ProDOS nicht mehr lauffähig.

Unter anderem sind folgende Features implementiert worden:

- Zeilenorientierter Editor mit jedem erdenklichen Redigierkomfort (Insert, Delete, Tab, Restore, freie Cursorbewegung in allen vier Richtungen, Eingabe von Ctrl-Buchstaben in Applesoft-Zeilen usw.)
- Renumber (Zeilen-Umnumerierung)
- Xreference (sortierte Variablenliste)
- Suchen von Tokens, Strings und Variablen
- dezimale und hexadezimale Umrechnung
- Ausführung von Monitorbefehlen aus dem Editor heraus
- Listen des Applesoft-Programms in speicherinterner Form als Hex-Dump
- Suchen von Hex-Folgen, Adressen oder Speicherstellen im gesamten RAM-Bereich einschließlich der Language-Card
- frei definierbare Tastatur-Macrobefehle

Der Applesoft-Editor liegt in einem von ProDOS geschützten Bereich und läßt sich per Tastendruck vorübergehend abschalten und ebenso einfach wieder aktivieren.

Gerätevoraussetzung: Apple II+, IIe oder IIc

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

Vorschau

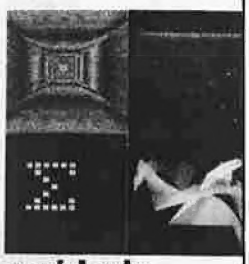
Im nächsten Heft, das Ende November erscheint, lesen Sie u. a.

- Wie man die Grafik verdoppelt
  Teil 2: Double Hires
- Ampersand macht's möglich
  INSTRING-Befehl in Applesoft
- Block-Tracer für ProDOS
- ProDOS für Anfänger, Teil 1
- IF-THEN-ELSE-Befehl
  simuliert in Applesoft
- Prometric
  Ein Apple-Kompatibler mit 65C02
- Hardware-Test: 3,5-Zoll-Laufwerke

Sie haben einen Apple...

wir haben die Bücher...

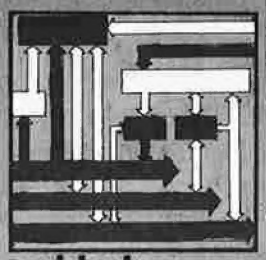
Apple Pascal Grafik
Tom Swan



22 PASCAL-Programme, mit denen Sie die Grafik-Möglichkeiten Ihres Apples voll ausschöpfen:
DESIGNER ermöglicht es Ihnen, eigene Zeichensatze zu entwerfen;
GREDIT unterstützt Sie beim Entwerfen und Abspeichern kompletter Bildschirmgrafiken;
PRINTFOTO bringt Ihre Entwürfe aufs Papier.
Daruberhinaus bietet das Buch eine Fülle fertiger Prozeduren, die Sie zeitsparend in Ihre eigenen Programme einbauen konnen.

ISBN: 3-89058-009-2 ..... DM 49,--
komplett mit Disk DM 89,--

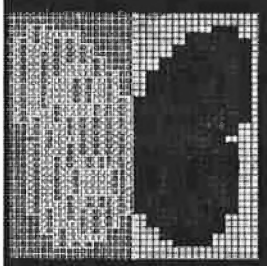
Apple II Schaltpläne
Winston D. Gayler



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

ISBN: 3-89058-012-2 ..... DM 64,--

Apple II Raster Grafik
Jeffrey Stanton



Die Qualität kommerzieller Arcadespiele laßt sich mit APPLESOFT BASIC alleine nicht erreichen. Jeffrey Stanton fuhrt in die Eigenarten der hochauflosenden Apple-Grafik ein und präsentiert schließlich eine Reihe extrem schneller Assembler-Routinen, mit denen Sie viele Effekte bekannter Spiele selbst programmieren können. Gute BASIC-Kenntnisse werden vorausgesetzt, eine Einführung in Assembler-Programmierung wird gegeben.

ISBN: 3-89058-006-8 ..... DM 49,--
komplett mit Disk DM 89,--

Apple II Assembler-Programmierung
Roger Wagner

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

ISBN: 3-89058-003-3 ..... DM 48,--
komplett mit Disk DM 89,--

Mikrocomputer Grafik
Roy E. Myers

Endlich anspruchsvolle Computergrafik für BASIC-Programmierer!
Mikrocomputer Grafik
- enthält fast 80 lauffertige BASIC-Programme, die die beschriebenen Grafikkonzepte illustrieren
- ist für den Anfänger geschrieben und erfordert nur elementare Mathematikkenntnisse
- enthält Techniken zur Erzeugung dreidimensionaler Grafiken in zwei Dimensionen, Hidden Line- und Hidden Surface-Methoden und Skalierung, Rotation und Translation von Grafiken
- bietet eine Einführung in die Animationstechnik

ISBN: 3-89058-000-9 ..... DM 49,--
komplett mit Disk DM 89,--

*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: .....

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 102869 · 6900 Heidelberg schicken.

Buchtitel

Name

Straße

Ort

Unterschrift