Nr. 4/85 April
DM 6,50, sfr 6,50, öS 50, Lit 5900, hfl 7,50

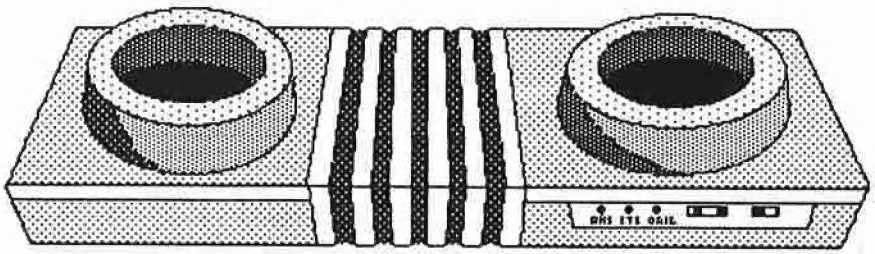
PEEKER

MAGAZIN FÜR APPLE-COMPUTER

Graf-quattro
Menü-Generator
Macros für 65CO2
Modem-Programm
Apples Maus lernt Pascal
CP/M für Einsteiger
Erphi-Controller

Software und Kopierrecht

Hüthig
PUBLIKATION



hib Computerladen
Äußere Bayreuther Straße 72
Postfach 21 01 25

8500 Nürnberg 21

Telefon: 0911/515 939 - Teletex: 2627 - 911 8253

AKUSTIK-KOPPLER
Dataphon s21d,
300 Baud Modem,
nach CCITT V.21 Standard,
m. FTZ-Nr. 18.13.1917.00,
Gebühren- u. anmeldefrei,
V24/RS-232 Standard-
Schnittstelle (25-Pin),
Vollduplexbetrieb,
Answer-, Originate-und
Auto-Modus nur DM 298,00

TELEKOMMUNIKATIONS-KOMPLETT-PAKET
bestehend aus:
1 Dataphon s21d,
1 Anschlußkabel (RS-232-
-Schnittstelle zum
Apple II-Gameport),
1 Terminalprogramm
"HIB MODEM-TRANSFER" nur DM 398,00

BROTHER-Drucker f. alle Apple-Rechner:

HR-5 (Thermodrucker, V.24 oder Centr.-Interf.)	DM	499,00
M-1009 (Matrixdrucker mit V.24 + Centronics)	DM	799,00
HR-15 XL (Typenraddr.)		
mit Centronics-Interface	DM	1698,00
mit V.24-Interface	DM	1748,00
Drucker-Anschlußkabel für APPLE //c oder Macintosh an: HR-5 (V.24), HR-15 (V.24), M-1009	DM	114,00

INFO-COUPON:
Ich interessiere mich besonders für:

- ☐ Telekommunikation
- ☐ Apple II - Interfaces
- ☐ Apple II - Laufwerke
- ☐ BROTHER-Drucker
- ☐ Software für Macintosh

Bitte senden Sie mir ausführliches Informationsmaterial.

ABSENDER:
_____
_____
_____

LAUFWERKE für den Apple II

Sakata-Laufwerk, SFD-155 (Chinon), 1 x 40 Track (163 K-Byte) komplett anschlußf. im Gehäuse	DM	598,00
TOSHIBA 2 x 80 Track	DM	648,00
AUTO-PATCH-CONTROLLER komplett mit Handbuch u. Formatierungs-Diskette	DM	348,00
DISK-DOPPEL-STATION 1,2 M-Byte, bestehend aus: 2 Laufwerke à 2x80 Track im Gehäuse, 1 Auto-Patch-Controller mit Handbuch u. Formatierungs-Diskette, anschlußfertig	nur DM	1898,00
SONDERANGEBOT: TEAC FD-55F	nur DM	598,00

hib Computerladen

INTERFACE-KARTEN für den Apple II

16 K-RAM-Card	DM	129,00
Disk Interface-Card	DM	198,00
Z-80-Card	DM	179,00
80-Zeichen-Card	DM	249,00
Super Serial Card	DM	298,00
128 K RAM-Card	DM	498,00
IC-Test-Karte	DM	398,00

MICROSOFT-Software für den Macintosh:

Basic Version 2.0	DM	598,00
Multiplan deutsch	DM	848,00
Chart (Grafik-Paket)	DM	498,00
Word (prof. Textprogr.)	DM	778,00
File (Datenbank)	DM	778,00
als Komplettpaket	nur DM	3298,00

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

EDITORIAL

Kein Thema wird momentan heftiger diskutiert als das Kopierwesen in der EDV-Branche, denn zur Zeit versucht die Software-Industrie mit allen ihr zur Verfügung stehenden Mitteln (strafrechtlich: Hausdurchsuchung, Beschlagnahme usw.; zivilrechtlich: Abmahnung, Schadenersatz usw.), den Raubkopierern auf den Pelz zu rücken. Deshalb beginnt der vorliegende Peeker mit einem Feature über Software und Kopierrecht, wobei ich aber im Gegensatz zu manchen anderen Darstellungen nicht nur für die Software-Urheber, sondern auch für die Software-Benutzer Partei ergreifen werde, weil teilweise der Versuch unternommen wird, aus dem Urhebergesetz einen für Software geltenden Sonderschutz herauszudeuteln, für den es de jure keine Grundlage gibt. Das EDV-Urheberrecht steht aus zwei Gründen auf sehr wackligen Beinen: Zum einen ist im deutschen Urheberrechtsgesetz das sog. „Computerwerk" (= Computerprogramm) mit keinem Wort erwähnt und damit auch nicht gesondert geregelt. Zum anderen gibt es fast keine Software-Streitfälle, die in Form von Musterprozessen durch alle Instanzen geboxt wurden. Deshalb kann man in Detailfragen nur Mutmaßungen äußern, indem man die Paragraphen „analog" auslegt.

Beispiel: Programmiersprachen

Prof. Dr. H. Hubmann ist der Meinung, daß Computerwerke den Sprachwerken zuzuordnen sind: „Gleichgültig ist auch, welche Sprache gewählt wurde, ob die inländische oder eine fremde, ob eine lebende oder eine tote, ob eine Kunstsprache, wie Esperanto, *oder eine Computersprache, wie Algol usw.* („Urheber- und Verlagsrecht", S. 90). An anderer Stelle heißt es: „Die Methode und Technik des Schaffens sowie die vom Inhalt losgelöste Formgebung als solche sind nicht Gegenstand des Urheberschutzes; eine Sprache wie Esperanto, eine Schrift wie Stenographie oder eine Versform sind nicht geschützt". (S. 86).

Was würde daraus folgen? Wenn Computerwerke Sprachwerke sind, dann sind Computersprachen Sprachen. Da jedoch weder Sprachen wie Englisch oder Deutsch noch Dialekte wie Hessisch oder Schwäbisch Urheberschutz genießen, wären auch weder Computersprachen wie Pascal oder Basic noch Computerdialekte wie Microsoft-Basic oder Wang-Basic geschützt. Und dies würde konkret bedeuten: Aus urheberrechtlicher Sicht würde jedermann ohne Lizenz Applesoft-Basic benutzen dürfen, wie auch jedermann Hessisch benutzen darf.

Man könnte jetzt noch einen Schritt weitergehen und auch die Betriebssysteme, die üblicherweise mit den Programmiersprachen verwoben sind, den Computersprachen zuordnen. Folglich wären dann Disk-Operation-Systems wie CP/M, MS-DOS, Apple-Pascal, DOS 3.3 und ProDOS nicht urheberrechtlich schutzfähig. Wenn dem so wäre, würde dies letztlich einer Systemfirma wie Microsoft, die überwiegend vom Verkauf selbstentwickelter Programmier- und Betriebssysteme lebt, die Existenzgrundlage entziehen, weil jedermann die Microsoft-Sprachen unentgeltlich benutzen dürfte, wie dies ja auch bei Esperanto der Fall ist. Mit dem gegenwärtigen deutschen Urheberrechtsgesetz kommt man also hier keinen Schritt weiter.

Ulrich Stiehl

Hüthig
PUBLIKATION

inhalt

4/85

Impressum

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

Verleger und Herausgeber:
Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften:
Heinz Melcher

Chefredakteur:
Ulrich Stiehl (us) Tel. (0 62 21) 48 93 52
(Bitte nur in redaktionellen Angelegenheiten anrufen)

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

PEEKER

MAGAZIN FÜR APPLE-COMPUTER

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

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

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

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

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Knock, Knock! Who's there?

Software und Kopierrecht

von Ulrich Stiehl

Wenn es in diesen Tagen an der Tür klopft, ist es möglicherweise nicht der Briefträger („The postman always rings twice...", Cain). Denn zur Zeit findet eine groß angelegte Hausdurchsuchungswelle bei mutmaßlichen Raubkopierern statt. Polizisten stürmen die Studentenbuden in Karlsruhe und anderen Informatik-Hochburgen und beschlagnahmen neben den vermuteten Schwarzkopien auch Apple-Hardware als „Raubkopiererwerkzeug". Dabei soll es dem Vernehmen nach zu peinlichen Verwechslungen gekommen sein. So wurden neben den „Händlern" auch die *Käufer* belangt. Ferner kassierte die Polizei in Unkenntnis der Sachlage offenbar teils Originalprogramme sowie Videobänder, Tonbandkassetten u.a.m.
Wir sind der Meinung, daß das Kopierunwesen, das inzwischen groteske Formen angenommen hat, unbedingt eingedämmt werden muß. Es wäre jedoch ratsam, wenn man etwas differenzierter vorginge. Es genügt nicht, daß man nur die Schüler und Studenten anschwärzt und zu den Prügelknaben der Nation macht, denn auch bei den „offiziellen" Händlern wird zuhauf kopiert. Nur spricht man hier nicht von „Schwarzkopien", sondern verbrämend von „Sicherungs- und Arbeitskopien".

Aus diesem aktuellen Anlaß befassen wir uns nachfolgend in extenso mit der „Vervielfältigung zum persönlichen Gebrauch" (§ 53 UrhG) und zeigen auf, was nach dem gegenwärtigen Urheberrechtsgesetz erlaubt und verboten ist. Als Quellen benutzen wir „Hubmann, Urheber- und Verlagsrecht, 4. Aufl., Beck-Verlag" (kurz Hubmann genannt) sowie „Fromm/Nordemann, Urheberrecht, 5. Aufl., Kohlhammer-Verlag" (kurz Nordemann genannt). Die eingestreuten Schaubilder stammen aus meinem „Verlagsbuchhändler", S. 66ff.

Zur Einstimmung in das Thema zunächst einige Fakten zum Kopierunwesen, die unbewiesen in den Raum gestellt werden. Haken Sie einmal ab, was Ihnen bekannt vorkommt!
1. Bibliotheken: „Bibliotheken haben in den letzten Jahren Fotokopierautomaten aufgestellt, mit denen jeder Bibliotheksbenutzter fotokopieren kann, was und wieviel er will." (Nordemann, S. 328)
2. Kopiershops: „Gewerbliche Fotokopieranstalten fragen ebenfalls nicht nach Zweck und Zahl, wenn ihnen Aufträge erteilt werden." (Nordemann, S. 328)
3. Schulen: „In vielen ... Schulen werden ... aus dem Exemplar der Lehrerbücherei oder gar aus einem kostenlosen Ansichtsexemplar ganze Kapitel fotokopiert und im Unterricht verwendet." (Nordemann, S. 328)
4. Computerclubs: Die einschlägigen Treffen der AUGE usw. dienen meist weniger dem Erfahrungs- als vielmehr dem Programmaustausch, wobei die kopiergeschützten Programme mittels „Kopien der kopiergeschützten Kopierprogramme kopiert" werden. („Der Tag geht zu Ende, Johnny Locksmith kommt...")
5. Hardwarehändler: Bei den Händlern sind Kopien von Originalprogrammen oft Zugaben zur käuflich erworbenen Hardware („Kommen Sie mal hinter. Hab' da noch was für Sie...").
6. Kleinanzeigen: In Kleinanzeigen vieler Zeitschriften werden unter Rubriken wie „Verkaufe" und „Tausche" Kopien von Originalprogrammen angeboten. („Wegen Systemwechsel zu verkaufen...").

1. WAS IST EIN COMPUTERWERK?

Das „Gesetz über Urheberrecht und verwandte Schutzrechte" (= Urheberrechtsgesetz = UrhG) vom 9. September 1965 soll die Urheber von Werken der Literatur, Wissenschaft und Kunst in bezug auf ihre Werke bis zu 70 Jahre nach dem Tod des Urhebers schützen (§ 1 UrhG). Das UrhG enumeriert in § 2, Absatz 1 u.a. folgende Werkarten:

- Sprachwerke
- Musikwerke
- Tanzkunstwerke
- Kunstwerke
- Lichtbildwerke
- Filmwerke

„Computerwerke" (= Computerprogramme) werden also nicht gesondert erwähnt. Insoweit ist es fraglich, ob Computerprogramme eine vom gegenwärtigen Gesetz noch nicht gesondert erfaßte spezielle Werkkategorie darstellen oder den wissenschaftlichen Sprachwerken zu subsumieren sind. Momentan wird man wohl das letztere annehmen müssen, so daß die Kopier- und Zitierrecht-Paragraphen, die vorwiegend auf wissenschaftliche Sprachwerke abheben, analog anzuwenden sind. Zumindest dürfte sich der in Druckform oder als Datenträger erschienene *Quellcode* eines Programms wegen der verbalen Kommentare zweifelsfrei als wissenschaftliches Sprachwerk bezeichnen lassen. In Einzelfällen wird man „Computerwerke" auch den Kunstwerken oder Musikwerken zuordnen können, z.B. bei musikalisch untermalten Computergrafikprogrammen.

1.1. Persönliche geistige Schöpfung

Als **Werk** gemäß § 2, Absatz 2 gilt nur die „persönliche geistige Schöpfung":

Persönlich besagt, daß das Werk von einer natürlichen Person erstellt werden muß. Mechanisch, automatisch oder computermäßig hergestellte Produkte sind demzufolge keine urheberrechtlichen Werke. So ist beispielsweise der mit Hilfe eines Assemblers erstellte Quellcode eines Programms ein Werk. Ob jedoch der auf dem Quellcode basierende Objektcode ein Werk darstellt, ist umstritten, denn dieser wird bekanntlich nicht „persönlich", sondern „mechanisch" per Knopfdruck erzeugt. Würde man dagegen den Objektcode direkt in Maschinensprache, d.h. ohne Zuhilfenahme eines Assemblers schreiben, so würde auf alle Fälle ein „persönliches" Werk geschaffen werden. Aus urheberrechtlicher Sicht besteht der Quellcode teils aus Befehlskürzeln, die in einer künstlichen Programmiersprache formuliert sind, sowie teils aus Kommentaren, die in einer natürlichen Sprache geschrieben sind. Der aus dem Quellcode durch Assemblierung oder auch durch Compilierung entstandene Objektcode ist lediglich eine Folge hexadezimaler Zahlen, die vom Mikroprozessor als Maschinenbefehle interpretiert werden. Eine Zahlenfolge als solche, z.B. die dezimale Folge 2, 3, 5, 7, 11, 13...(Primzahlen), ist nicht automatisch ein Computerwerk, wie auch eine Tonfolge, z.B. c, d, e, f, g, a...(Tonleiter), nicht automatisch ein Musikwerk ist. In beiden Fällen fehlt die „persönliche" Note. Der Objektcode ist zwar formal betrachtet eine „Übersetzung", doch zweifelsohne keine Übersetzung im Sinne einer *Bearbeitung* (§ 3 UrhG), wie ja auch die Computerübersetzung, z.B. der Prawda ins Amerikanische, kein Werk ist. So gesehen, ist wahrscheinlich nur der Quellcode schutzfähig, und gerade dieser befindet sich praktisch nie auf den Schwarzkopien der Raubkopierer.

Geistig besagt, daß das Werk im „Gehirn" entstanden sein muß. Ein Lied ist ein Werk, dagegen die Darbietung des Liedes (= Gesang) nur eine *Leistung* (= verwandtes Schutzrecht). Ob der Objektcode, wenn er schon, wie wir meinen, kein Werk darstellt, so doch wenigstens eine „Leistung" ist, ist ebenfalls umstritten. Schon das normale Menschenverstand sagt uns, daß die „Knopfdruck"-Assemblierung (für Richter: Man tippt nur das

Kürzel „ASM". Damit hat sich die Leistung) nicht in die gleiche Kategorie wie die echten Leistungen (Gedichtsvortrag, Gesangsdarbietung usw.) eingeordnet werden darf.
Die Entdeckung eines Naturgesetzes (= *Erfindung*) wird nicht vom Urheber-, sondern vom Patentrecht abgedeckt. Auch mathematische und logische Gesetze sowie *Algorithmen* sind grundsätzlich nicht schutzfähig. Beispielsweise ist der im Jahre 1962 von C.A.R. Hoare entwickelte Quicksort-Algorithmus kein Werk.

Schöpfung besagt, daß etwas *Individuelles* geschaffen werden muß, das über das Alltägliche hinausgeht und somit *Werkcharakter* hat. So fehlt beispielsweise Geschäftsbriefen meist die „Schöpfungshöhe", während Gedichte fast immer „Werkcharakter" haben. Im Gegensatz zum Patentrecht muß eine Schöpfung „neuartig", aber *nicht* neu sein. Während ein Patent grundsätzlich nur dem *Erst*erfinder erteilt wird, wäre im Urheberrecht auch die 175ste Implementierung von Quicksort schutzfähig. Wie verträgt sich dies jedoch mit der obigen Aussage? Der theoretische Quicksort-Algorithmus als solcher ist nicht schutzfähig, wohl aber die individuelle, persönliche und zugleich „neuartige" Implementierung in einer beliebigen Programmiersprache. Je unkomplizierter jedoch ein Algorithmus, desto geringer ist der Spielraum für „Eigenschöpfungen". In der Tat läßt ein korrektes Quicksort-Modul kaum noch Raum für Individualität und „Neuartigkeit". Groteskerweise wäre eine fehlerhafte und eben dadurch „individuelle" Quicksort-Implementierung stets geschützt, während eine korrekte Implementierung eher gemeinfrei (= nicht schutzfähig) ist. Im Gegensatz zur patentmäßigen Erfindung muß ein urheberrechtliches Werk nicht „stimmen". Bekanntlich beruht die gesamte schöngeistige Literatur auf „erdichteten Dingen", die nicht wahr sind im Sinne tatsächlicher Begebenheiten.

1.2. Geschützte „Computerwerke"

Ein „Computerwerk" gilt als geschützt, wenn es erstens eine persönliche geistige Schöpfung darstellt und wenn zweitens die Schutzfrist für das Werk noch nicht abgelaufen ist. Wegen des Algorithmencharakters wird man jedoch vielen „Computerwerken" oder zumindest den algorithmischen Teilen von „Computerwerken" (Sortierroutinen, Rechenroutinen usw.) die Schutzfähigkeit absprechen müssen. Gleiches dürfte wohl für reine „Objektcode-Werke" gelten, d.h. für Programme, von denen nur der Objektcode und nicht zusätzlich der Quellcode veröffentlicht wird.
Nach Ablauf der **Schutzfrist** – d.h. normalerweise 70 Jahre nach dem Tod des Urhebers – wird das Werk gemeinfrei und darf dann von jedermann benutzt werden. Das deutsche Urheberrechtsgesetz schützt selbstverständlich zunächst nur deutsche Staatsangehörige (§ 120). Aufgrund internationaler Urheberrechtsabkommen (insbesondere Welturheberrechtsabkommen und Revidierte Berner Übereinkunft) werden Ausländer im Inland wie Inländer geschützt (**Assimilationsprinzip**, s. Hubmann, S. 21ff.). Wie vielen Lesern sicherlich bekannt ist, sind „Computerwerke" im amerikanischen „Copyright Act" inzwischen gesondert geregelt. Diese Regelungen kommen jedoch aufgrund des Assimilationsprinzips in der Bundesrepublik nicht zur Anwendung, weil sich sonst der amerikanische Urheber in Deutschland besser stellen würde als der deutsche Urheber selbst. Übrigens sei an dieser Stelle erwähnt, daß sich das eingekreiste „c" © aus deutscher Sicht auf das Welturheberrechtsabkommen bezieht (Artikel III: „...bear the symbol © accompanied by the name of the copyright proprietor and the year of first publication..."). In Deutschland selbst sind derartige Formalitäten nicht erforderlich.
In den USA liest man oft von Public-Domain-Software, d.h. vom Autor als gemeinfrei *erklärten* und damit von jedermann benutzbaren Programmen. In Deutschland ist demgegenüber zumindest das sog. *Urheberpersönlichkeitsrecht* (§ 11 UrhG) unveräußerlich, so daß beispielsweise eine Urheber, der einem Computerclub seine Programme kostenlos zur Verfügung gestellt hat, wegen gewandelter Überzeugung seine Werke wieder zurückziehen kann (§ 42 UrhG).

1.3. „Geheimwerke"

Vorab eine begriffliche Klärung: „Schützen" hat im Rechts- und Computerwesen eine unterschiedliche Bedeutung. Ein „geschütztes Urheberwerk" ist eine Schutz genießende persönliche geistige Schöpfung. Ein „geschütztes Computerwerk" (= „Geheimwerk") ist ein kopiergeschütztes Programm. Ein geschütztes Computerwerk ist nicht notwendigerweise ein geschütztes Urheberwerk.
Ein Verlag, der ein Sprachwerk verlegt, ist aufgrund des jeweiligen Landespressegesetzes verpflichtet, 1 Vervielfältigungsstück an die entsprechende Landesbibliothek sowie darüber hinaus 1 weiteres Vervielfältigungsstück an die Deutsche Bibliothek in Frankfurt in *lesbarer* Form abzuliefern (Pflichtabgabegesetz). Folglich kann jeder Urheber anhand des fremden Sprachwerkes prüfen, ob sein eigenes Werk in urheberrechtlichem Konflikt zu dem hinterlegten steht. Dagegen gehört es zu den Abnormitäten der „Computerwerke", daß diese häufig in doppelt unlesbarer Form veröffentlicht werden: Erstens erscheint anstelle des Quellcodes vielfach nur der unlesbare Objektcode. Zweitens erscheint anstelle des ungeschützten Codes vielfach nur der kopiergeschützte und mithin unlesbare Code. „Computerwerke" werden somit zu „Geheimwerken". Sind jedoch „Geheimwerke" überhaupt schutzfähig?

Beispiel 1
Ihr holden Schwäne,
Und trunken von Küssen
Tunkt ihr das Haupt
Ins heilig-nüchterne Wasser

Beispiel 2
You lovely swans,
and drunk with kisses
you dip your heads
into the sacred sober water

Beispiel 3
A0 B0 C1 24 BC BD BE BF
00 80 20 30 14 2A 2A 3C
A0 10 11 FF 7B 2B 2A 20
D0 E0 1F 3A 4C ED FD 60

Beispiel 1 ist ein Auszug eines Gedichtes von Hölderlin („Hälfte des Lebens"). Würde Hölderlin noch leben, so würde es als *Originalwerk* voll unter den Schutz des Urheberrechtsgesetzes fallen. Analoges würde für die englische Übersetzung in *Beispiel 2* gelten, die eine *Bearbeitung* im Sinne von § 3 UrhG in Verbindung mit § 23 UrhG darstellen würde. Wie steht es jedoch mit *Beispiel 3*? Handelt es sich hierbei um eine weitere Übersetzung in eine nur dem Urheber bekannte Geheimsprache oder um ein eigenständiges Produkt in der Form eines „unchiffrierten Geheimwerkes", das den vollen Schutz des Urheberrechts genießt?
Nehmen wir als fiktiven Rechtsfall an, daß ein „Schwarzkopierer" dieses vom „Urheber" veröffentlichte „Werk" (= Beispiel 3), weil er dachte, es sei „ungeschützter Quatsch", ohne Einwilligung kopierte und auf dem „Markt" zu verkaufen versuchte, worauf der Urheber gegen den Schwarz-

PRO METRIC® IST DA!

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

Neu von DRV: C-DOS, das 80-Track DOS für B2 und Kompatible, verarbeitet Integer, DOS 3.3 und Prodos-Files DM 149,––
CALVADOS, deutsche Textverarbeitung DM 299,––

TEAC FD55F DM 499,––

TEAC FD 55 B	DM 449,––
FDC4-Controler	DM 179,––
Shugart-Bus-Kabel	DM 45,––
PREH-Commander AK87	DM 279,––
MONITOR 12A, 22 MHZ	DM 355,––
M100-DRUCKER	DM 825,––
Graphik-Par.-Interface	DM 125,––
OPERATOR 2-Tastatur	DM 580,––
Super COM-Tastatur (deutsche Belegung)	DM 369,––
Z 80 B-Karte 6 MHz (AL 5 comp)	DM 799,––
80-Zeichenkarte 4 Zeichensätze	DM 279,––

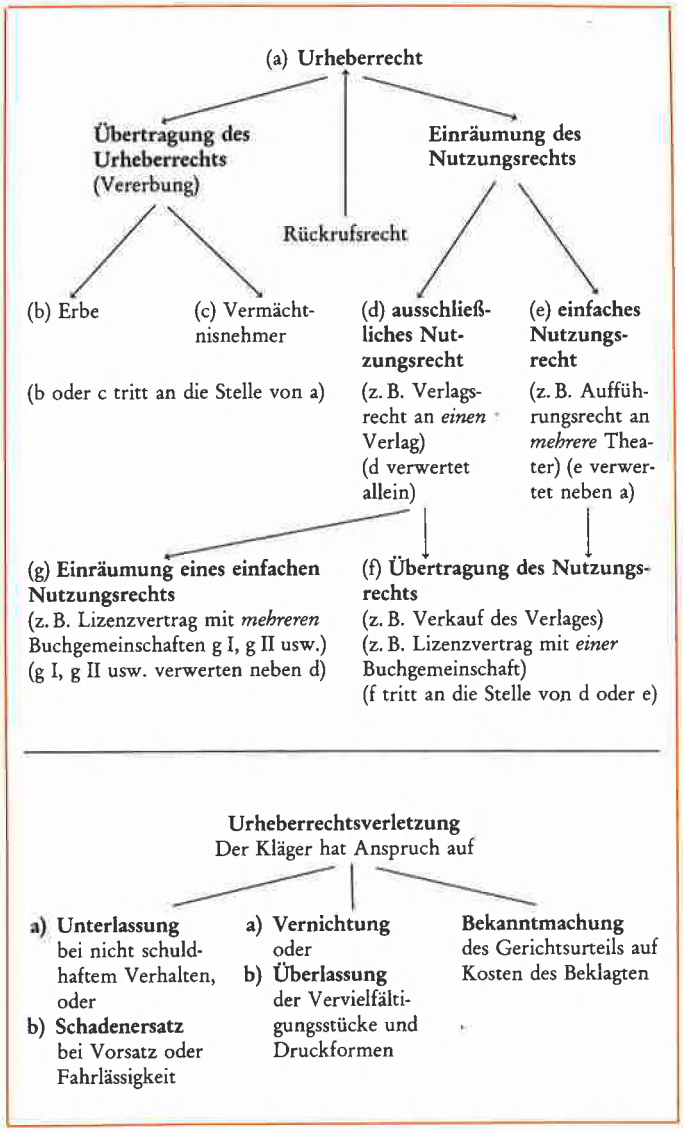
YE-DATA 480 DM 694,––

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

NEU: 15 MB Winchester Slimline inkl. Controler DM 3699,––
Preh Commander AK 106 Starlight-1 21 Funktionstasten DM 359,––

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

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



Urheberrechtsverletzung
Der Kläger hat Anspruch auf

a) Unterlassung
bei nicht schuldhaftem Verhalten, oder
b) Schadenersatz
bei Vorsatz oder Fahrlässigkeit

a) Vernichtung oder
b) Überlassung
der Vervielfältigungsstücke und Druckformen

Bekanntmachung
des Gerichtsurteils auf Kosten des Beklagten

Kopieren zum eigenen Gebrauch aus Schriftwerken
(Bis zu 7 gleiche Kopien, keine Verbreitungsbefugnis)

Persönlicher Gebrauch
a) Natürliche Personen
b) Häuslicher, privater, nichtberuflicher Werkgenuß
c) Werk braucht nicht erschienen zu sein
d) Es können *ganze* Werke kopiert werden
e) Keine Vergütungspflicht

Sonstiger eigener Gebrauch
a) Natürliche oder juristische Personen
b) Kopien kann man durch Kopieranstalten herstellen lassen
c) Eigener Gebrauch (z. B. firmeninterner Gebrauch)
d) Beruflicher/gewerblicher Zweck erforderlich

Wissenschaftlicher Gebrauch
a) Studenten, Privatgelehrte
b) Rein wissenschaftlicher Zweck; nicht gegeben bei Populärwissenschaft (Sachbuch) oder Literatur
c) Es können *ganze* Werke kopiert werden, soweit geboten
d) Werke brauchen nicht erschienen zu sein
e) keine Vergütungspflicht

Nicht-wissenschaftlicher, rein informatorischer Gebrauch
a) Es dürfen nur *Werkteile* oder ganze Zeitschriftenaufsätze kopiert werden
b) Das kopierte Werk muß erschienen sein
c) Falls ein Werk vergriffen ist, darf es auch als ganzes Werk kopiert werden
d) Für gewerbliche Unternehmen (Industrie- und Handelsfirmen, nicht freie akademische Berufe, wie Rechtsanwälte) besteht Vergütungspflicht

Hinweis: Eine Novellierung des Kopierrechts (UrhG § 53, 54) wird voraussichtlich in naher Zukunft durchgeführt werden.

APPLE II PASCAL

Betriebssystem und Sprache

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

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

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

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

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

„Nach Unterlagen von Apple Deutschland hergestellt"

Apple II Betriebssystem,
272 Seiten, DM 49,–

Apple II Sprache,
216 Seiten, DM 39,–

Pascal 1.2 Addendum, etwa 100 Seiten,
DM 36,– (1. Quartal '85)

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

Weiterführende Literatur...

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

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

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

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

Computer für Kinder
(Sally Greenwood Larson)
Ein Buch für Kinder, ihre Lehrer und Eltern.
„Computer für Kinder" richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern dieses Buch bewußt geschrieben wurde.
Unterhaltsam und leicht verständlich.
A4 quer, Fadenheftung, DM 29,80

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

tm 4153

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

In Vorbereitung:
Macintosh Programmier-Handbuch DM 59,–

kopierer Anzeige erstattete. Es kommt zu einer Beschlagnahme von „Schwarzkopien" und anschließend zum Prozeß. Der Schwarzkopierer behauptet, das Werk sei eine „Folge von mechanischen Chiffren", während der Urheber entgegnet, es handele sich um ein „Hölderlin-Übersetzung" in einem unknackbaren Code". Wir würde der Richter den Fall beurteilen, wenn er das „Geheimwerk" *nicht* vorher „knackt"?

Aus dieser Persiflage lernen wir, daß bei „Geheimwerken" niemand zweifelsfrei nachprüfen kann, ob sie schutzfähig sind oder nicht. Folglich kann auch niemand aufgrund des *Urheber*rechtsgesetzes (wohl aber u.U. aufgrund anderer Gesetze) zur Rechenschaft gezogen werden, denn sonst würde der Willkür Tür und Tor geöffnet. Man stelle sich einmal vor, das Urheberrechtsgesetz selbst wäre seinerzeit im Bundesgesetzblatt in chiffrierter Form veröffentlicht worden! Wer hätte sich dann danach richten können? Man stelle sich ferner plastisch vor, *alle* Bücher und Zeitschriften würden in codierter Form erscheinen! Wer würde dann noch von schutzwürdigen „Kulturgütern" sprechen?

Obgleich das Pflichtabgabegesetz mit dem Urheberrechtsgesetz direkt nichts zu tun hat, ist es dringend an der Zeit, daß auch für „Computerwerke" eine Pflichtabgabe als *Quellcode* gesetzlich vorgeschrieben wird, weil dann jedermann bei der Deutschen Bibliothek oder der jeweiligen Landesbibliothek in die „Computerwerke" Einsicht nehmen kann, womit allererst entschieden werden kann, ob ein Werk im Sinne des Urheberrechtsgesetzes vorliegt. *Denn man merke sich: Nicht jedes „Computerwerk" ist automatisch ein urheberrechtlich geschütztes Werk. Vielmehr besteht bei kopiergeschützten Werken stets der Verdacht, daß sie nicht urheberrechtlich geschützt sind.*

Der Grund, warum „Computerwerke" keine „Geheimwerke" sein dürfen, wird klarer, wenn man sich die zwei Seiten eines „Computerwerks" näher ansieht. Die **äußere Seite** ist das Computerprogramm als *Erscheinung und Leistung,* und die **innere Seite** ist das Computerprogramm als *Werk.* Urheberrechtlich schutzfähig ist normalerweise nur die innere Seite. Ein einfaches Beispiel: Auf der kopiergeschützten Demo-Diskette „A Guided Tour of Macintosh" der Firma Apple ist ein kleines Programm, das ein stark vereinfachtes Klavier simuliert, auf dem man mit der „Maus" spielen kann. Die Grafik der Tasten wird man kaum als ein Kunstwerk bezeichnen können. Und da das Programm keine Melodie enthält, liegt auch kein Musikwerk vor. Was der Anwender sieht, also die äußere Seite des Programms, ist damit überhaupt kein Werk, wie ja auch ein echtes Klavier kein Werk im Sinne des Urheberrechts ist. Wie steht es dagegen mit der inneren Seite oder dem Programm selbst? Da die „Guided-Tour"-Diskette kopiergeschützt ist, kann man sich die innere Seite nicht ansehen und damit auch nicht entscheiden, ob eine persönliche geistige Schöpfung vorliegt. Das Klavierdemo könnte theoretisch sogar ein Plagiat sein, das die Firma Apple bei irgendeinem anderen Autor „abgekupfert" hat. In diesem Fall wäre dann der Macintosh-Käufer zugleich Erwerber einer Raubkopie. Da die Firma Apple eine Vorliebe für Abmahnungen hat, hätte sie hier endlich eine Handhabe, um die gesamte eigene Kundschaft abzumahnen... Aber Spaß beiseite: Dieses triviale Beispiel zeigt bereits auf das deutlichste, daß der Käufer eines „Geheimwerkes" nicht überprüfen kann, ob er ein Plagiat erworben hat.

Bei größeren Programmpaketen fallen einem immer wieder ähnliche Leistungen auf: Sortierroutinen, Mergeroutinen, Bildschirmroutinen, Rechenroutinen, Druckroutinen usw. Wenn man solche „Computerwerke" dann „knackt", stellt man fest, daß oft nicht nur die Leistungen dieser Programme ähnlich sind. Man nehme das Sort-Modul des Autors X, das Merge-Modul des Autors Y, den Maskengenerator des Autors Z, kleide das ganze in eine neue äußere Form und mache dann alles so kopiergeschützt, daß man sich Quellenangaben sparen kann – und fertig ist das neue Programmpaket. Manch ein Richter würde vom Ledersessel fallen, wenn er die Quellcodes diverser kopiergeschützter Programmpakete detailliert vergleichen würde. Im Gegensatz zu wissenschaftlichen Werken, die reichlich Zitate aus fremden Werken enthalten (was im übrigen rechtens ist), findet man bei den kommerziellen Computerprogrammen praktisch nie Zitate. Dies kann zweierlei bedeuten: Entweder sind die Programmautoren so genial, daß sie jedes Programm von Grund auf neu schreiben, oder sie wiegen sich in der Hoffnung, daß keiner merkt, bei wem sie „abgekupfert" haben, weil sie ihre Programme nachträglich kopiergeschützt machen.

Ein Beispiel: Nehmen wir an, Sie kaufen das Diskettenkopierprogramm B, das sich mit der Bildschirmmeldung „Insert Disks. Press any key to continue" meldet. Nehmen wir ferner an, daß Sie bereits ein älteres Kopierprogramm A besitzen, das sich mit der gleichen Meldung vorstellt und im übrigen die gleiche Funktion des Diskettenkopierens erfüllt. Da Phrasen in der Art „Press any key to continue" ohnehin keinen Urheberrechtsschutz genießen und da die Kopierleistung des Programms ebenso wie die Kopierleistung eine Photokopiergerätes nicht schutzfähig ist, werden Sie sich fragen, ob Sie nicht einem Plagiator zum Opfer gefallen sind. Schutzfähig wären nämlich nur die Kopier*programme* selbst, genauer gesagt die persönlich-geistig-schöpferische Art und Weise der Implementierung der Programme. Aber davon sehen Sie nichts, wenn diese kopiergeschützt sind.

Wer Banknoten erhält, muß selbst oder durch einen geeigneten Experten nachprüfen lassen können, ob die Scheine „Blüten" sind. Ähnlich muß auch der Erwerber einer Software selbst oder durch einen Experten in Erfahrung bringen können, ob die Software ein Plagiat ist. Dies ist jedoch bei kopiergeschützten Werken per definitionem nicht möglich. Während also Buchautoren ihre Manuskripte, Komponisten ihre Noten und Künstler ihre Gemälde für jedermann nachprüfbar offenbaren, zeigen viele Software-Entwickler nur „Geheimwerke" vor, die mit großer Selbstverständlichkeit als Urheberwerke ausgegeben werden, obwohl sie es wahrscheinlich oft gar nicht sind. Aufgrund dieser unbefriedigenden Rechtslage ergibt sich nur eine sinnvolle Empfehlung: Grundsätzlich sollten nur diejenigen „Computerwerke" Urheberrechtsschutz genießen, deren Quellcodes veröffentlicht oder öffentlich hinterlegt werden.

2. WAS IST EINE WERKKOPIE?

Der § 53 UrhG lautet auszugsweise wie folgt:

Vervielfältigung zum persönlichen Gebrauch:

(1) Zulässig ist, einzelne Vervielfältigungsstücke eines Werkes zum persönlichen Gebrauch herzustellen.

(2) Der zur Vervielfältigung Befugte darf die Vervielfältigungsstücke auch durch einen anderen herstellen lassen;...

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

(4) ...

(5) Ist nach der Art eines Werkes zu erwarten, daß es ... durch Übertragung von einem Bild- oder Tonträger auf einen anderen zum persönlichen Gebrauch vervielfäl-

tigt wird, so hat der Urheber des Werkes gegen den Hersteller von Geräten, die zur Vornahme solcher Vervielfältigungen geeignet sind, einen Anspruch auf Zahlung einer Vergütung für die durch die Veräußerung der Geräte geschaffene Möglichkeit, solche Vervielfältigungen vorzunehmen. ... Der Anspruch kann nur durch eine Verwertungsgesellschaft geltend gemacht werden. Als Vergütung steht jedem Berechtigten ein angemessener Anteil an dem vom Hersteller aus der Veräußerung der Geräte erzielten Erlös zu; die Summe ... darf fünf vom Hundert dieses Veräußerungserlöses nicht übersteigen.

§ 53 UrhG regelt den „persönlichen Gebrauch" und § 54 UrhG den „sonstigen eigenen Gebrauch". „Ersterer ist ein besonders begünstigter Unterfall des letzteren" (Hubmann, S. 158). Aus Platzgründen beschränken wir uns auf den „persönlichen Gebrauch".
Es gibt folgende Arten von „Computerwerk-Exemplaren" (allesamt in der Regel in Diskettenform):

a) Primäroriginal = Quellcode
b) Sekundäroriginal = Objektcode
c) Originalwerkstück = von (b) hergestellte Kopie
d) Primärvervielfältigungsstück = von (c) hergestellte Kopie
e) Sekundärvervielfältigungsstück = von (d) hergestellte Kopie

Die Originalwerkstücke zeichnen sich in der Regel durch ein entsprechendes Etikett auf der Diskette aus. Raubkopien rekrutieren sich aus Primär- und noch häufiger aus den Sekundärvervielfältigungsstücken, doch ist – wie wir gleich sehen werden – keineswegs jedes Vervielfältigungsstück automatisch eine Raubkopie.

2.1. Beispielfälle

Anstelle akstrakter Definitionen erläutern wir das, was nach dem Kopierrecht (un)zulässig ist, anhand von Beispiel(zitat)en:

1. Originale: Mit Originalwerkstücken kann man machen, was man will. Man kann sie „an Verwandte und Freunde weitergeben" (Hubmann, S. 158). „Nach der Erschöpfungstheorie ist es sogar gestattet, Werkstücke weiterzuverkaufen" (Hubmann, S. 158).

2. Einzelexemplare: Es ist „zulässig, *einzelne* Vervielfältigungsstücke eines Werkes ... zum persönlichen Gebrauch herzustellen" (Hubmann, S. 158). Aufgrund des sog. Bremer Schulbuchprozesses, bei dem es darum ging, daß Bremer Schulen ganze Klassensätze vervielfältigt hatten, ist der Begriff „einzelne Stücke" als maximal 7 Exemplare auszulegen (weil eine Schulklasse in der Regel mehr als 7 Schüler umfaßt). Konkret gesprochen heißt dies, daß man z.B. „Appleworks" kaufen und im Anschluß daran maximal 7 Kopien an maximal 7 Freunde verteilen kann.

3. Beliebige Dritte: „Persönlicher Gebrauch liegt vor, wenn durch die Vervielfältigung lediglich dem Hersteller (gemeint: Kopierer) oder dem mit ihm durch ein persönliches Band verbundenen Bekanntenkreis der Werkgenuß vermittelt werden soll" (Hubmann, S. 158). Unzulässig ist es also, wenn für *beliebige* Dritte Kopien hergestellt werden. Wenn mir also mein Freund eine Kopie seines als Originalwerkstück erworbenen „S-C-Macroassemblers" mitbringt, so ist dies (aus der Sicht meines Freundes) zulässig, weil wir durch „ein persönliches Band" verbunden sind. Wenn aber ein Apple-Händler einem Macintosh-Käufer eine Kopie vom neuen „Finder 1.1" in die Hand drückt, dann ist dies (aus der Sicht des Händlers) unzulässig, denn ein Gewerbetreibender handelt per definitionem nie aus „privater Verbundenheit".

4. Kopieranstalt: „Der zur Vervielfältigung Befugte darf die Vervielfältigung auch durch einen anderen, sogar durch einen Gewerbebetrieb, herstellen lassen" (Hubmann, S. 159). Man könnte also eine „Knackerfirma" beauftragen, „Prolok"-geschützte Originalprogramme zu duplizieren.

5. Weiterverbreitung: Gemäß § 53, Absatz 3 dürfen Vervielfältigungsstücke nicht verbreitet, d.h. entgeltlich oder unentgeltlich weitergegeben werden. Der Verkauf oder Tausch von Vervielfältigungsstücken durch Anzeigen in Zeitschriften ist demnach stets verboten, zumal sich eine Anzeige an beliebige Dritte wendet, mit denen man natürlich nicht durch ein „persönliches Band" verbunden sein kann. „Ebenso ist der Verkauf ... mittels Zeitungsanzeige ... unzulässig; auch ein Tausch unter den gleichen Bedingungen würde gegen Absatz 3 verstoßen" (Nordemann, S. 324). (Der Verkauf oder Tausch von *Originalwerkstücken* ist indes stets erlaubt.) Andererseits heißt es bei Nordemann, S. 323: „Ein zur persönlichen Benutzung hergestelltes Werkstück (gemeint: Vervielfältigungsstück) darf man jedoch später veräußern, wenn man es nicht mehr braucht". Das heißt im Extremfall: Wenn die (bis zu 7fache) Vervielfältigung des „Visicalc"-Originals „von vornherein mit der Absicht geschieht, das Vervielfältigungsstück zu verkaufen" (Nordemann, S. 323), so ist dies unzulässig. Braucht man jedoch das Originalwerkstück und die bis zu 7 Vervielfältigungsstücke von „Visicalc" später nicht mehr, so kann man sie nunmehr verkaufen. Nimmt man als weiteren Extremfall an, daß man – ähnlich wie bei Heiratsanzeigen – „auf diesem Wege" ein „persönliches Band" mit zunächst „unpersönlichen Fremden" herstellt, denen man später aus nunmehr „privater Verbundenheit" eines der „Visicalc"-Vervielfältigungsstücke im Wege des „freundschaftlichen Tausches" zukommen läßt, dann wird klar, daß das urheberrechtliche Kopierrecht so viele Löcher wie der Schweizer Käse hat.

2.2. Ein fairer Kompromiß

In § 53, Absatz 5 wurde für Musik- und Videobänder (einschließlich Kassetten und Schallplatten) eine Geräteabgabe eingeführt, weil der Gesetzgeber hier von der Erwartung ausging, daß diese Bild- und Tonträger üblicherweise zum persönlichen Gebrauch vervielfältigt werden. Dies gilt in vollem Umfang auch für Software. Deshalb würde meines Erachtens die ideale Lösung des Kopierdilemmas darin bestehen, daß man für Diskettenlaufwerke (abstrakt: Datenträgervorrichtungen) eine Geräteabgabe und/oder für Leerdisketten (abstrakt: Datenträger) eine Diskettenabgabe einführt. Diese Abgaben sollten – wie bei Tonbandgeräten usw. – 5% des Fabrikabgabepreises nicht übersteigen und durch eine neue Wahrnehmungsgesellschaft (analog zur GEMA) vereinnahmt und dann an die Software-Autoren verteilt werden. (Ähnliches sollte im übrigen auch für Photokopiergeräte gelten). Damit würde dann die Vervielfältigung zum persönlichen Gebrauch nicht mehr diskriminiert oder gar – wie bei der gegenwärtigen „Verhaftungswelle" – kriminalisiert werden. Aus Gründen der Rechtsklarheit bei gewerbsmäßigen Raubkopierern sollte der *Verkauf* von Vervielfältigungsstücken urheberrechtlich grundsätzlich unzulässig sein. Und schließlich sollten nur *Primär*vervielfältigungen (von Originalwerkstücken) zugelassen werden.

Graf-quattro

N.G. Barbieri

Künstlerische Grafik für jedermann

Teil 1: Wie ein Supereditor entsteht

1. DIE GRUNDLAGEN

Es hat alles vor etwa 16.000 Jahren angefangen. Da nagte Flintstone jr. an einem vom Vater mitgebrachten Bisonrippchen und wollte endlich wissen, wie wohl ein ganzer Bison aussehen mag. Flintstone sen. versuchte dies zuerst mit viel Grunzen, Herumspringen und Armewedeln zu erklären, aber der Gesichtsausdruck des Filius war und blieb ziemlich verdattert. Plötzlich hatte Flinstone sen. einen gewaltigen Geistesblitz und machte die beste Erfindung seit dem Feuer. Er besorgte sich Lehm in verschiedenen Farben und malte einen Bison in seiner ganzen Pracht an die Höhlenwand, der heute noch in Altamira zu bewundern ist.

1.1. Ohne Input kein Output

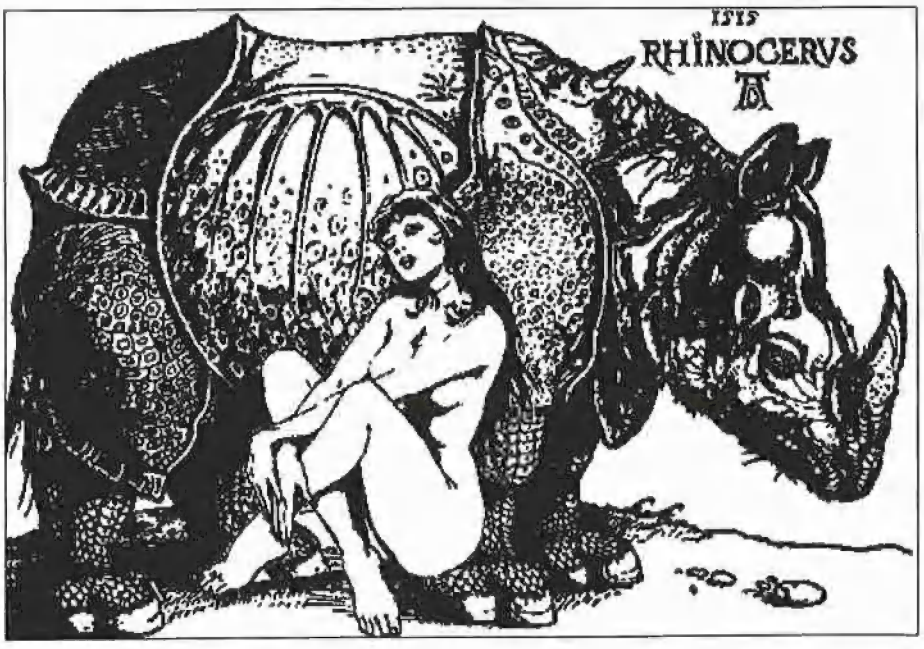
Wie macht man aber Bilder mit dem Computer? Wie kommt Onkel Dagobert auf die Mattscheibe? Auch im Falle der Computergrafik gilt das eiserne Gesetz: Ohne Input kein Output!

Computergrafik kann grob in zwei Kategorien eingeteilt werden: Die eine nennt man **Vektorgrafik**. Hierbei handelt es sich hauptsächlich darum, eine mehr oder weniger komplexe Informationsdatei zu verwalten. Die zweite Kategorie heißt **Rastergrafik**, und in diesem Fall wird nur ein Bitmuster verändert und verwaltet. Diese Beitragsserie wird sich nur mit dieser zweiten Grafikkategorie befassen.

Neben dem Grafiktablett, dem klassischen Eingabegerät der Computergrafik, kann die Videokamera oder ein Digitizer den grafischen Input ganzer Bilder besorgen. Mühseliger ist der Input mittels Paddle, Joystick oder Trackball. Auch kommt hier das „Tablett des armen Mannes", die Maus, in Frage. Die alphanumerische Tastatur, unterstützt von geeigneter Software, ist als Input und Verarbeitungswerkzeug ebenfalls nicht zu verachten.

Der Output auf dem Monitor ist bei der Computergrafik selbstverständlich. Hardcopies (Printouts, Ausdrucke) können mit einem grafikfähigen Drucker hergestellt werden, z.B. mit dem hervorragenden Programm von Jürgen und Dieter Geiß im Peeker Heft 1/84. Glückliche Besitzer eines Plotters können mit einem geeignetem Grafik-Dump-Programm weitaus bessere Ergebnisse erzielen. Foto, Film und Videoband können sich als weitere Medien anbieten.

Wenn wir den Input mit einer Kakaobohne und den Output mit einer fertigen Tafel Schokolade vergleichen, wird sofort klar, daß dazwischen eine ganze Menge passieren muß. Man braucht sozusagen Werkzeuge, um eine Transformation des ursprünglichen Werkstoffes in das fertige Produkt zu erreichen, und zwar die richtigen! Versuchen Sie einmal, eine Kuckucksuhr mit Vorschlaghammer und Rohrzange zu reparieren. Also mußte für unsere Zwecke erst einmal ein Raster-Grafik-Editor erstellt werden. Sie werden hier vielleicht einwenden, daß bereits E-Z Draw, Alphaplot und viele andere Editoren vorhanden sind. Stimmt! Ich kenne sie alle, und doch habe ich es mit diesen Editoren noch nie geschafft, ein künstlerisches Bild in die Welt zu setzen. Was mir vorschwebt, ist der „Totale Editor", die komplette Manipulation einer Grafikseite.



In jedem steckt ein Dürer

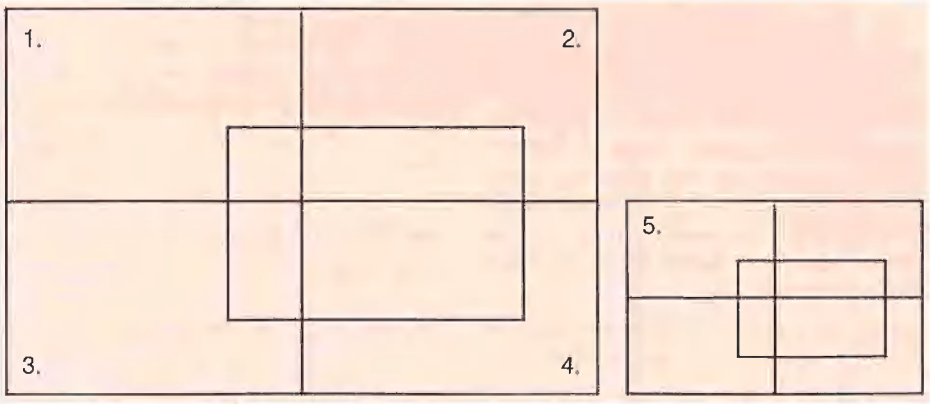


Bild 2: Graf-quattro-Seite und Pilotseite. Was auf der Pilotseite angezeigt ist, wird von Graf-quattro automatisch in die Grafikseiten geholt.

1.2. Programmstruktur

Bevor ich auf die Vorzüge eines solchen Supereditors eingehe, einige Worte zum Gesamtkonzept oder zur „Programmstruktur", wie Informatiker sagen würden. Zuallererst: Ein solches Programm ist so groß, daß der gesamte Speicher des Apple nicht reichen würde. Deshalb muß das Gesamtprogramm in Module aufgeteilt werden. Jedes dieser Module wird bei Bedarf von der Diskette geladen und gezielt eingesetzt. Um die Daten braucht man sich keine Sorgen zu machen, denn bei der Rastergrafik sind es eben nur eine Menge Bits, die auf einer der Grafikseiten gut aufgehoben sind. Jedes Modul ist also ein Teilprogramm, das eine bestimmte Verarbeitung der Grafik erlaubt. Wenn alle notwendigen Module erstellt sind, braucht man nur noch ein „Hauptmenü" zu schreiben.

Jedes Modul wird aus einem Applesoft-Programm und einer Reihe von Assemblerroutinen, die die eigentliche Arbeit schnell und effizient erledigen, bestehen. Jede Assemblerroutine wird so gestaltet, daß sie eine einzige Aufgabe erfüllt und auch einzeln, d.h. unabhängig vom Gesamtprogamm, anwendbar ist. Auf diese Weise kann der Leser die Assemblerroutinen je nach Bedarf in eigene Programme einbauen. Ich werde versuchen, in den Beiträgen jeden einzelnen Programmteil so zu kommentieren, daß auch der Assembler-Anfänger folgen kann.

1.3. Speicherverteilung

Bevor man mit einem Progamm beginnt, sollte man sich immer einen Überblick über den vorhandenen Speicher und dessen Aufteilung in die benötigten Bereiche verschaffen.
Der verfügbare RAM-Speicher wird in vier Bereiche eingeteilt:

- Der untere Teil von $0800 (2048) bis $1FFF (8191) ist für Applesoft reserviert. Keines unserer Applesoft-Progamme darf also größer als 6042 Bytes werden, einschließlich der Variablen und Strings.
- Der mittlere Bereich von $2000 (8192) bis $5FFF (24575) ist von den zwei Grafikseiten belegt. Die erste Grafikseite (HGR) wird unsere Arbeitsseite sein, die zweite Grafikseite (HGR2) benutzen wir als Grafikhilfsseite, sozusagen als Schmierbogen oder Ablegeplatz für Hilfsmittel.
- Der obere Bereich ab $6000 (24576) besteht wiederum aus zwei Teilen. In dem unteren Teil werden die jeweils notwendigen Assemblerroutinen untergebracht. Oberhalb des Assemblerbereichs werden wir, je nach Bedarf, andere Werkzeuge bereithalten, wie z.B. Shape-Tabellen und Fonts für die Beschriftung der Grafikseite.
- Der kleine Bereich von $0300 (768) bis $03CF (975) ist wie geschaffen, um Informationen sicher aufzubewahren, die von einem Modul zum anderen zu übertragen sind. Außerdem werden wir hier Variablen und Pointer für den Assemblerteil unterbringen.
- In der Zeropage $0000 (0) bis $00FF (255) sind noch eine ganze Menge Stellen frei, die das Betriebssystem überhaupt nicht benutzt. Diese werden natürlich auch zum Einsatz kommen, um die Assemblerroutinen schneller und kompakter zu gestalten.

2. DIE MODULE

Kommen wir nun zu den einzelnen Modulen. So wie jede Assemblerroutine auch allein einsetzbar ist, ist jedes Modul praktisch ein in sich geschlossenes Programm. Wer also ein bestimmtes Modul nicht braucht, kann es weglassen.
Die Aufteilung in einzelne Module mit genau definierten Aufgaben hat auch den Vorteil, daß jeder Anfänger das Ganze besser verfolgen kann.

2.1. Page-Editor

Das ist im Grunde genommen nichts Neues. Page-Editoren gibt es inzwischen wie Sand am Meer. Ich werde aber versuchen, etwas extrem Bequemes mit einer Menge zusätzlicher Funktionen zu schaffen. Also nicht nur Punkte, Linien, Vierecke, Boxen, Kreise, Ellipsen, Bogen usw., sondern auch die Möglichkeit, Teile der Grafik zu transponieren, zu verdoppeln, Strukturen zu wiederholen usw. Der Page-Editor kann wie alle anderen Module eingesetzt werden, um das Vierfache der auf der Apple-Grafik verfügbaren Grafikpunkte zu editieren, und zwar mit dem folgenden Modul.

2.2. Graf-quattro

Dieses Modul entstand, weil ich mich geärgert habe. Als ich meinen ersten Apple kaufte (beinahe in der Steinzeit), war es schon ein kleines Wunder, 280 x 192 Punkte als Hires-Grafik zur Verfügung zu haben. Später kamen die anderen und protzten mit ihrer größeren Auflösung. Zuerst war ich neidisch, doch dann habe ich mir gesagt: Was die anderen können, kann auch mein Apple, wenngleich auf Umwegen.
Also, wenn man vier Grafikseiten im Quadrat zusammenklebt, verfügt man über eine Grafik mit 560 x 384 Punkten. Selbstverständlich kann man solch eine Grafik nicht auf einmal im Bildwiederholspeicher unterbringen. Ein beliebiger Ausschnitt davon geht aber schon. Demgegenüber kann mit einem Plotter oder mit einem Drucker der ganze Bereich gedumpt werden. Für den Printerdump sind vielleicht die oben zitierten Autoren so nett, eine modifizierte Version ihres wirklich sehr guten Programms den Peeker-Lesern anzubieten. Ich stelle mir vor, daß erst die Abschnitte eins und zwei (siehe Bild) jeweils in die erste und zweite Grafikseite geladen und dann parallel gedumpt werden. danach folgen die Abschnitte drei und vier, so daß das komplette, zusammenhängende Bild auf Papier gebracht werden kann. Falls eine Language-Card vorhanden ist, könnte man ein gemovtes DOS einsetzen und somit Platz genug für alle

vier Grafikseiten schafften, und zwar von $2000 bis $9FFF.
Zu den vier Grafikseiten wird eine fünfte angelegt, die eine geschrumpfte Version aller vier Seiten enthält und als Kontrollseite dient. Das Graf-quattro-Modul arbeitet folgendermaßen:

Auf der Diskette werden zuerst fünf Grafikseiten angelegt. Die vier Seiten der eigentlichen Grafik können ebensowohl leer sein oder bereits Teile der Grafik enthalten. Mit einem Digitizer könnte man sogar ein Grundmotiv in der Gesamtauflösung als Basis für die Grafik abtasten. Diese Seiten bekommen einen beliebigen Dateinamen, gefolgt von den Zahlen 1 bis 5. Die Aufgabe des Graf-quattro-Moduls ist es, mit Hilfe der fünften Seite, die wir Pilotseite nennen wollen, aus den vier Seiten einen beliebig zu plazierenden Ausschnitt von 280 x 192 Punkten herauszuholen. Danach bringt das Graf-quattro-Modul den editierten Ausschnitt in die vier Grafikseiten zurück. Die Pilotseite wird automatisch entsprechend geändert oder ergänzt. Wer eine RAM-Karte besitzt und als Pseudo-Disk benutzt, kann dies alles sehr schnell während des Editiervorgangs erledigen.

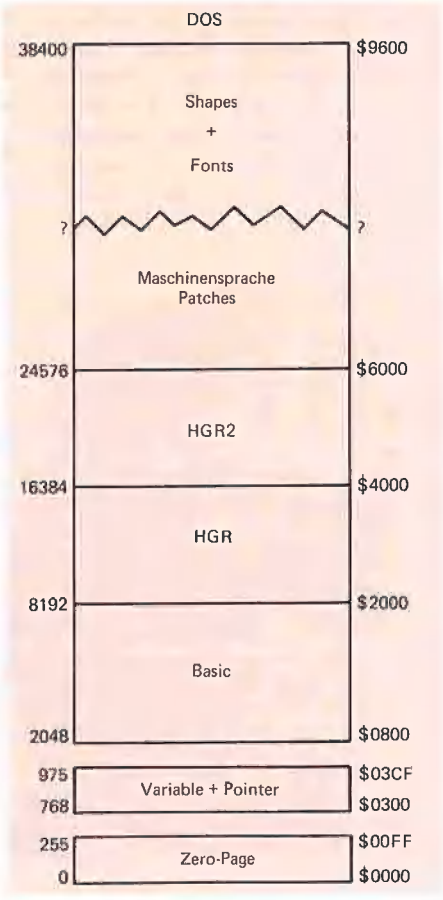


Bild 1: Speicherverteilung bei Graf-quattro

2.2.1. Spezialroutinen

Scrunch: (to scrunch = wörtlich „stauchen"; auch „to crunch") Die Assemblerroutinen dieses Moduls werden schon für das Graf-quattro-Modul eingesetzt. Diese Funktion ist als separates Modul sehr nützlich, um eine Grafikseite linear um die Hälfte zu verkleinern, so daß das Ganze oder Teile davon in anderen Grafiken Verwendung finden können.

Zoom: (to zoom = wörtlich „auseinanderziehen") An sich findet dieses Modul wenig Verwendung, da beim Aufblasen von einem Teil der Grafik die ganze Struktur vergröbert wird, so daß jeder Punkt in ein Blöckchen von vier Punkten verwandelt wird. In einzelnen Fällen könnte ein gezoomter Ausschnitt aber von Nutzen sein.

Retusche mit Lupe: Damit wird eine feine und sehr genaue Punktretusche der Grafikseite ermöglicht. Um die Augen nicht übermäßig zu strapazieren, nutzt das Modul den Zoomeffekt als bewegliche Lupe.
Pinsel: Dieses Modul liefert uns eine Reihe von Hires-Pinseln, vom feinen Retuschepinsel bis zur dicken Quaste, und einen Radiergummi. Damit kann man die Grafikseite frei gestalten oder ergänzen, z.B. kann man sie mit Schattierungen oder Strukturen versehen.

Turtle: Dies ist eine Emulation der Turtlegrafik, die aber mit einigen zusätzlichen Funktionen ausgestattet ist. Man kann nicht nur eine Linie bilden und drehen, sondern sie auch in jeder Richtung verschieben. Die Fähigkeiten des Page-Editors werden so erheblich erweitert.

Spiegeln: Die ganze Seite oder ein beliebiger Ausschnitt davon kann, egal ob links/rechts oder oben/unten, gespiegelt werden. Der Editor wird viel variabler, und Teile der Grafik können für eine Weiterverwendung vorbereitet werden.

Drehen: Ein beliebiger Ausschnitt der Grafikseite bis zu max. 192 × 192 Punkten kann jeweils um 90 Grad gedreht werden. Dies ergänzt das Spiegelmodul, um z.B. Umrandungen oder Ornamentik anzubringen.
Fill: Eine komplizierte Assemblerroutine besorgt das punktgenaue Füllen jeder geschlossenen Fläche, gleichgültig wie sie gestaltet ist.
Kollage: Hier geht es nicht darum, viereckige Grafikausschnitte übereinander zu bringen; das kann schon jeder Macintosh-Rechner. Es ist vielmehr das exakte, konturierte Aufkleben von Grafikteilen auf eine darunterstehende Grafik gemeint! Peeker macht's eben möglich.

Shapes: Das Modul wird alle Geheimnisse der Apple-Shapes enthüllen. Für Grafikanfänger wird dieser Beitrag besonders interessant sein, da sie sich erfahrungsgemäß mit den Apple-Shapes immer schwer tun.

2.3. Editor mit Shapes

Dies ist der normale Shape-Editor.

2.4. Editor mit Autoshapes

Dieses Modul wird eine große Überraschung werden. Einfach einen beliebigen Ausschnitt auf einer Grafikseite definieren, und der Inhalt wird automatisch in ein Shape verwandelt, das dann nicht nur gedreht, sondern auch links/rechts und oben/unten gespiegelt werden kann. Und es kommt noch mehr: Die normalen Apple-Shapes brauchen drei Bits, um einen Vektor zu definieren. Das macht die Sache umständlich und erlaubt normalerweise nur, zwei Vektoren in einem Byte unterzubringen. (Die restlichen zwei Bits sind selten zu gebrauchen, es sei denn, sie beinhalten einen Vektor mit „Move to".) Das Auto-Shape-System, das ich vorhabe, benötigt nur zwei Bits pro Vektor, so daß immer vier Vektoren in einem Byte Platz finden. Wer das nicht glaubt, soll es abwarten. Das System wird außerdem erheblich schnellere Shapes als das Applesoft-Shape-System bieten.

2.5. Font-Editor

Mit diesem Modul kann jeder seine eigenen Schriften entwerfen. Es handelt sich hier nicht um einen Abklatsch von HIGHER TEXT, wo jeder Buchstabe genau dasselbe Blöckchen belegt. Bei diesen Fonts werden die Buchstaben eine beliebige Größe bekommen, und jeder Buchstabe wird in der Breite genau seinen zugewiesenen Platz belegen. Das nennt man variable Schrittbreite (oder drucktechnisch „Dickte", Anm. d. Red.), d.h. das „I" ist schmaler als das „M".

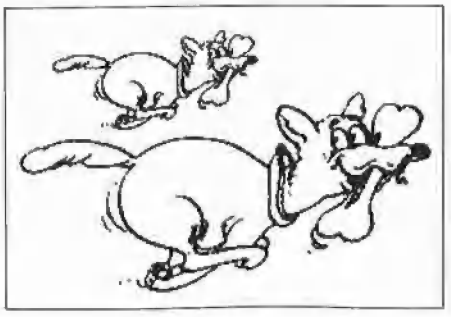


Bild 3: „Fipsi"

pandabooks

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

Ablage Bearbeiten Zeichensatz Große Stil **Hilfsmittel**

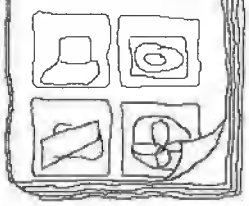
Pandabooks

Apple Grafik? Pandabooks!

Gratis! AppleQuick

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

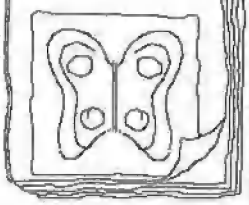
Mikrocomputer Grafik
Roy E. Myers



Endlich anspruchsvolle Computergrafik für BASIC-Programmierer! Mikrocomputer Grafik
- enthält fast 80 lauffertige BASIC-Programme, die die beschriebenen Grafikkonzepte illustrieren
- beschreibt "Hidden Line"- und "Hidden Surface"-Methoden, Skalierung, Rotation und Translation von Grafiken
- bietet eine Einführung in die Animationstechnik

3-89058-000-9 292 S DM 49,--
komplett mit Disk DM 89,--

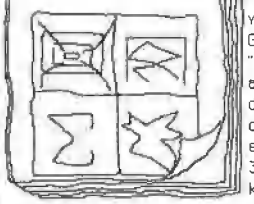
Apple II Raster Grafik
Jeffrey Stanton



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

3-89058-006-8 299 S DM 49,--
komplett mit Disk DM 89,--

Apple Pascal Grafik
Tom Swan



22 Pascal-Programme, mit denen Sie die Grafik-Möglichkeiten Ihres Apple voll ausschöpfen.
"DESIGNER" ermöglicht es Ihnen, eigene Zeichensätze zu entwerfen;
"GREDIT" unterstützt Sie beim Entwerfen kompletter Bildschirm-Grafiken;
"PRINTFOTO" bringt Ihre Entwurfe aufs Papier. Darüberhinaus bietet das Buch eine Fülle fertiger Prozeduren, die Sie zeitsparend in Ihre eigenen Programme einbauen konnen.

3-89058-009-2 280 S DM 49,--
komplett mit Disk DM 89,--

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

Bestellcoupon

Name:
Anschrift:

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

Menge	Titel	Preis
1	AppleQuick	gratis

Sie müssen Kalkulationen erstellen ...

Sie müssen Rechnungen schreiben...

Sie müssen schnell ein Angebot zusammenstellen...

Sie brauchen eine zuverlässige Lagerverwaltung

Sie wollen Geschäftsbriefe schreiben

Dann brauchen Sie

HandMac

Das Programm für das Handwerk.

- das Ihnen diese Arbeiten abnimmt
- das speziell für Sie in Deutschland entwickelt wurde
- das Sie sofort benutzen können, ohne wochenlange Schulung, ohne seitenstarkes Handbuch

HandMac

läuft auf Apple Macintosh und gibt es nur im autorisierten Fachhandel

Information: copy team gmbh
schuhstr. 23
8520 erlangen
09131 – 21383
Auch auf der Hannover-Messe!

2.6. Text-Editor

Unter Verwendung der zu diesem Zweck geschaffenen Fonts kann man die Grafikseite horizontal oder vertikal beschriften. Das ist in groben Zügen das, was die Grafquattro-Serie bieten wird.

3. DIE XPLOT-ROUTINE

Und nun, um den Beitrag nicht so trocken ausgehen zu lassen, hier der erste Patch. Applesoft kann zwar den HPLOT-Befehl in verschiedenen mehr oder weniger brauchbaren Farben ausführen. Das Wichtigste wurde aber außer acht gelassen: der „Exklusiv-oder"-Modus. An sich wäre es eine Kleinigkeit gewesen, ein HCOLOR = 8 einzufügen. Platz genug war vorhanden, auch ohne andere Routinen zu beschneiden, wie z. B. die HFIND ($F5CB), die von Applesoft sowieso nicht benutzt wird. Es hätte genügt, die zwei Patches für DRAW und XDRAW vernünftiger zu gestalten, womit man Platz für XPLOT sowie ein paar andere Kleinigkeiten geschaffen hätte. Aber ROM ist eben ROM, und uns bleibt nichts anderes übrig, als das Fehlende hinzuzufügen. Eine XOR-Linie braucht keine Farbbestimmung, weil sie immer zum vorhandenen Hintergrund negativ dargestellt wird. Es gibt selbstverständlich mehrere Möglichkeiten, dies zu bewerkstelligen, auch effizienter als im Applesoft-Interpreter. Ich habe mich aber für eine modifizierte Applesoft-Routine entschieden, da sie kürzer ist und in bekannter Art und Weise arbeitet.

Die Routine – **XPLOT.ROUTINE** genannt – ist 181 Bytes lang, belegt den Speicherplatz unmittelbar nach der zweiten Grafikseite, also von $6000 (24576) bis einschließlich $60B4 (24756), und kann folgendermaßen angesprochen werden:
Vom Applesoft aus zuerst CALL 24576 und danach HPLOT-Befehle wie üblich bis zum Ende der Programmzeile oder zum nächsten Doppelpunkt. Der CALL fängt sozusagen die darauffolgenden HPLOT- oder HPLOT-TO-Befehle ab und verarbeitet sie selber. Damit kann man in Applesoft wie gewohnt programmieren. Ich könnte auch den Ampersand (&) strapazieren, habe dies aber bewußt nicht getan, da es eben nur ein kurzer Patch ist. Sind genug Patches für ein erstes Modul zusammen, kann man immer noch eine Tabelle mit sämtlichen &-Applesoft-Befehlen basteln.
Das kleine Applesoft-Testprogramm – **XPLOT.DEMO** genannt – illustriert, wie es gemacht wird. Nur einige Worte zur letzten Applesoft-Zeile. Wer es nicht bereits weiß: Dies ist ein Trick, um die Grafikseite mit einer bestimmten Farbe zu füllen. In Speicherstelle 28 wird die sogenannte Color-Mask gepokt, und der CALL füllt die Seite mit der gepokten Farbe. 127 bedeutet hier weiß = 1. Im einzelnen gilt:
schwarz 1 = 0
grün = 42
purpurrot = 85
weiß 1 = 127
orange = 170
blau = 213
weiß 2 = 255
Selbstverständlich können auch andere Werte gepokt werden, die dann aber verschiedene Streifenmuster erzeugen.

Für angehende Assembler-Freaks eine Aufklärung: Grafikpunkte und Linien funktionieren unter Verwendung aller drei A-, X- und Y-Register mit Hilfe einer Routine namens **HPOSN,** die die Aufgabe hat, einen unsichtbaren Cursor auf der Grafikseite zu positionieren. Außerdem wird der zuletzt geplottete Punkt zwischengespeichert, so daß anschließende Linien mit „TO" gerufen werden können. Die drei Register reichen aus, da der X-Koordinatenwert zwar mehr als acht Bits braucht, für den Y-Koordinatenwert 8 Bits aber völlig genügen (max. 191). Merkwürdigerweise gilt für die Cursorpositionierung (HPOSN) und das evtl. folgende HPLOT folgende Registerbelegung:
(1)
Im A-Reg. der Y-Wert
Im X-Reg. der X-Low-Wert
Im Y-Reg. der X-High-Wert
Nach der Positionierung des Cursors oder nach einem ersten HPLOT für eine darauf folgende Linie ist statt dessen folgende Belegung vonnöten:
(2)
Im A-Reg. der X-Low-Wert
Im X-Reg. der X-High-Wert
Im Y-Reg. der Y-Wert
Für den Assemblerprogrammierer bietet die XPLOT-Routine drei Entry-Points, die im Source-Listing mit EP1, EP2 und EP3 markiert sind.
EP1 = $601D ist XPOINT, und die Registerbelegung ist die übliche wie bei (1).
EP2 = $6039 ist XLINE mit derselben Registerbelegung wie XPOINT. Die darauffolgenden Zeilen besorgen den nötigen „Swap".
EP3 = $603F ist XLINE mit der üblichen Applesoft-Registerbelegung, wie unter (2).
Das war es vorläufig.

XPLOT.DEMO

```
1Ø REM XPLOT.DEMO
2Ø HIMEM: 8192:XP = 24576: HGR :
   POKE - 163Ø2,Ø:C = Ø:A = - 1
3Ø PRINT CHR$ (4);"BLOAD XPLOT.ROUTINE"
4Ø X1 = 138:X2 = 14Ø:X3 = X2:X4 = X1:
   Y1 = 94:Y2 = Y1:Y3 = 96:Y4 = Y3
5Ø A = A + 1: IF A = 2 THEN A = Ø: GOSUB 9Ø
6Ø FOR I = 1 TO 45: GOSUB 7Ø:X1 = X1 - 2:
   X2 = X2 + 2:X3 = X3 + 2:X4 = X4 - 2:
   Y1 = Y1 - 2:Y2 = Y2 - 2:Y3 = Y3 + 2:
   Y4 = Y4 + 2: NEXT : GOTO 4Ø
7Ø CALL XP: HPLOT X1,Y1 TO X2,Y2 TO X3,Y3
   TO X4,Y4 TO X1,Y1: RETURN
8Ø REM Das ist ein Trick!
9Ø C = Ø + 127 * (C = Ø): POKE 28,C:
   CALL 62454: RETURN
```

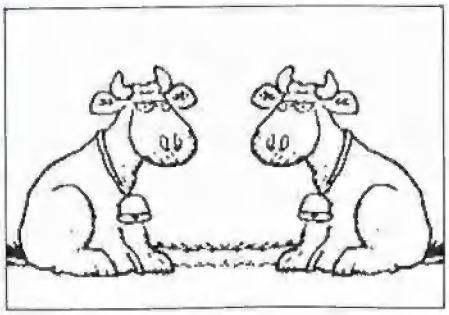


Bild 4: „Die zarteste Versuchung..."

XPLOT.ROUTINE

```
                        1           ORG  $6000
                        2   *
                        3   *       XPLOT.ROUTINE
                        4   *       =============
                        5   *
                        6   * von N.G.Barbieri, 1985
                        7   *
                        8   * Eine modifizierte Version der
                        9   * Applesoft-HPLOT-Routine, um
                        10  * XPLOT zu ermöglichen.
                        11  *
                        12  COLON    EQU  $3A        ;":"
                        13  HPLOT    EQU  $93        ;Token
                        14  TO       EQU  $C1        ;Token
                        15  CHRGOT   EQU  $B7
                        16  SYNCHR   EQU  $DEC0      ;Parse
                        17  HFNS     EQU  $F6B9
                        18  HPOSN    EQU  $F411
                        19  LFTRT    EQU  $F465
                        20  UPDOWN   EQU  $F4D3
                        21  *
                        22  *
6000: A9 3A             23          LDA  #<COLON
6002: 20 C0 DE          24          JSR  SYNCHR     ;Syntax
6005: A9 93             25          LDA  #<HPLOT     ;Check
6007: 20 C0 DE          26          JSR  SYNCHR
                        27  *
600A: 20 B7 00          28          JSR  CHRGOT
600D: C9 C1             29          CMP  #TO
600F: F0 22             30          BEQ  DOLINE
6011: 20 B9 F6          31          JSR  HFNS
6014: 48                32          PHA
6015: 20 B7 00          33          JSR  CHRGOT
6018: C9 C1             34          CMP  #<TO
601A: F0 0B             35          BEQ  XLINE1
601C: 68                36          PLA
                        37  *
                        38  * XPOINT - Reg.-Belegung:
                        39  *
                        40  * A-Reg. = Y-Wert
                        41  * X-Reg. = XL-Wert
                        42  * Y-Reg. = XH-Wert
                        43  *
601D: 20 11 F4          44          JSR  HPOSN      ;EP1
6020: A5 30             45          LDA  $30
6022: 29 7F             46          AND  #$7F       ;jetzt
6024: 4C 60 F4          47          JMP  $F460      ;XPOINT
                        48  *
6027: 68                49  XLINE1  PLA             ;XPLOT
6028: 20 11 F4          50          JSR  HPOSN      ;Beginn
                        51  *
602B: 20 B7 00          52  XLINE2  JSR  CHRGOT     ;XPLOT TO
602E: C9 C1             53          CMP  #TO
6030: F0 01             54          BEQ  DOLINE
6032: 60                55          RTS
                        56  *
6033: 20 C0 DE          57  DOLINE  JSR  SYNCHR
6036: 20 B9 F6          58          JSR  HFNS
                        59  *
                        60  * XLINE 1 - Reg.-Belegung:
                        61  *
                        62  * A-Reg. = Y-Wert
                        63  * X-Reg. = XL-Wert
                        64  * Y-Reg. = XH-Wert
                        65  *
6039: 84 9D             66          STY  $9D        ;EP2
603B: A8                67          TAY
603C: 8A                68          TXA             ;Reg.
603D: A6 9D             69          LDX  $9D        ;Swap
                        70  *
                        71  * XLINE 2 - Reg.-Belegung:
                        72  *
                        73  * A-Reg. = XL-Wert
                        74  * X-Reg. = XH-Wert
                        75  * Y-Reg. = Y-Wert
                        76  *
603F: 48                77          PHA             ;EP3
6040: 38                78          SEC
6041: E5 E0             79          SBC  $E0
6043: 48                80          PHA
6044: 8A                81          TXA
6045: E5 E1             82          SBC  $E1
6047: 85 D3             83          STA  $D3
6049: B0 0A             84          BCS  XLIN2
604B: 68                85          PLA
604C: 49 FF             86          EOR  #$FF
604E: 69 01             87          ADC  #$1
6050: 48                88          PHA
6051: A9 00             89          LDA  #$0
6053: E5 D3             90          SBC  $D3
6055: 85 D1             91  XLIN2   STA  $D1
6057: 85 D5             92          STA  $D5
6059: 68                93          PLA
605A: 85 D0             94          STA  $D0
605C: 85 D4             95          STA  $D4
605E: 68                96          PLA
605F: 85 E0             97          STA  $E0
6061: 86 E1             98          STX  $E1
6063: 98                99          TYA
6064: 18                100         CLC
6065: E5 E2             101         SBC  $E2
6067: 90 04             102         BCC  XLIN3
6069: 49 FF             103         EOR  #$FF
606B: 69 FE             104         ADC  #$FE
606D: 85 D2             105 XLIN3   STA  $D2
606F: 84 E2             106         STY  $E2
6071: 66 D3             107         ROR  $D3
6073: 38                108         SEC
6074: E5 D0             109         SBC  $D0
6076: AA                110         TAX
6077: A9 FF             111         LDA  #$FF
6079: E5 D1             112         SBC  $D1
607B: 85 1D             113         STA  $1D
607D: A4 E5             114         LDY  $E5
607F: B0 05             115         BCS  MOVEX2
6081: 0A                116 MOVEX   ASL
6082: 20 65 F4          117         JSR  LFTRT
6085: 38                118         SEC
6086: A5 D4             119 MOVEX2  LDA  $D4
6088: 65 D2             120         ADC  $D2
608A: 85 D4             121         STA  $D4
608C: A5 D5             122         LDA  $D5
608E: E9 00             123         SBC  #$0
6090: 85 D5             124 HCOUNT  STA  $D5
                        125 *
6092: A5 30             126         LDA  $30        ;Sim
6094: 29 7F             127         AND  #$7F       ;Salabim
6096: 51 26             128         EOR  ($26),Y
6098: 91 26             129         STA  ($26),Y
                        130 *
609A: E8                131         INX
609B: D0 04             132         BNE  XLIN4
609D: E6 1D             133         INC  $1D
609F: F0 8A             134         BEQ  XLINE2
60A1: A5 D3             135 XLIN4   LDA  $D3
60A3: B0 DC             136         BCS  MOVEX
60A5: 20 D3 F4          137         JSR  UPDOWN
60A8: 18                138         CLC
60A9: A5 D4             139         LDA  $D4
60AB: 65 D0             140         ADC  $D0
60AD: 85 D4             141         STA  $D4
60AF: A5 D5             142         LDA  $D5
60B1: 65 D1             143         ADC  $D1
60B3: 50 DB             144         BVC  HCOUNT
```

181 Bytes

von Dr. H. Kersten

Menü-Generator für den Apple II

Der beschriebene Generator soll dem Benutzer die langweilige und zum Teil nervende Arbeit der Herstellung von Textseiten („Menüs") auf dem Bildschirm abnehmen: Der Benutzer schreibt sein Menü „life" auf den Bildschirm, der Generator erzeugt aus dem vorgegebenen Bild ein Basic-Programm, das beim Ablauf gerade dieses Menü liefert. Textpassagen können als INVERSE, NORMAL oder FLASH gekennzeichnet werden, GET- und INPUT-Befehle werden nach Benutzerwunsch ebenfalls eingebaut, eine Bildschirm-Einrahmung ist leicht realisierbar. Die erzeugten Basic-Zeilen werden möglichst kurz gehalten. Zugrunde liegt das Betriebssystem DOS 3.3 und 40-Zeichen/Zeile-Darstellung.

Viele Programme neueren Datums sind menü-gesteuert, d.h. sie liefern dem Benutzer nach dem Start eine Übersicht über die verschiedenen Wahlmöglichkeiten (options) und bieten im allgemeinen hierfür Ersatzwerte (default values) an. Der Benutzer kann, sofern nötig, einzelne Werte ändern. Verlangt ein Programm umfangreiche Eingaben, so sind oft mehrere Menü-Seiten erforderlich.

Für das Herstellen solcher Menüs ist es deshalb notwendig, die erläuternden Texte in sinnvoller Reihenfolge und gut strukturiert auf dem Bildschirm erscheinen zu lassen und anschließend die Benutzerwünsche abzufragen. Die Programmierung solcher Menüs ist zwar nicht schwierig, oft aber eine zeitraubende und aufwendige Routine-Arbeit.

Eine effiziente Lösung dieser Problemstellung ist die Benutzung eines Generator-Programms: Man schreibt per Tastatur das Menü-Bild (alle Texte und sonstigen Zeichen) auf den Schirm, hebt dort, wo es notwendig erscheint, Zeichenketten z.B. mit INVERSE hervor und markiert die Positionen für spätere GET- und INPUT-Befehle. Aus diesen und einigen anderen Informationen erzeugt der Generator ein entsprechendes Basic-Programm.

Vorteile dieser Methode: Man sieht sein Menü „life", Korrekturen jeder Art sind leicht möglich, jegliches Abzählen von Zeilen oder Spalten entfällt, das erzeugte Programm läuft auf Anhieb fehlerfrei und läßt sich problemlos in ein bestehendes Programm integrieren. Sogar aufwendige Menüs erfordern wenig Zeit.

Man kann den Generator auch zur Erzeugung einfacher (aber umfangreicher) Textausgaben und für „Text-Grafik" (‚Bilder' mit normalen Druckzeichen) verwenden.

Arbeiten mit dem Generator-Programm

Nach dem Starten des Programms **MENUE.GENERATOR** werden zunächst folgende Parameter abgefragt:

1. Menü-Größe in Zeilen * Spalten (Ersatzwerte und Maximalwerte 24 * 40)
2. Zeilennummer der ersten zu erzeugenden Basic-Zeile (Ersatzwert 100)
3. Zeileninkrement (Ersatzwert 10)
4. Name, Slot und Drive eines Textfiles, auf dem das erzeugte Basic-Programm abgelegt wird (Ersatzwerte: MENUE/S6/D1).

Will man alle Ersatzwerte übernehmen, so gibt man auf die Frage nach Änderungen N(ein) ein. Im anderen Fall trägt man die gewünschten Daten ein; einzelne Ersatzwerte übernimmt man durch Eingabe von „RETURN". Soweit wie möglich werden die Eingaben auf formale Korrektheit überprüft.

Nachdem für alle Optionen Werte festliegen, wird der Bildschirm frei, der Cursor erscheint oben links, der Editier-Vorgang kann beginnen: Man gibt nun alle Texte und Zeichen ein, die das Menü ausmachen – jeweils an der richtigen Schirmposition. Hierbei kann man folgende Kontrollfunktionen/-tasten benutzen:

Cursor-Positionierung: ESC mit I, J, K, M wie beim Apple-Monitor, Ctrl-H Cursor

links, Ctrl-U Cursor rechts. Überfahrene Zeichen bleiben stets erhalten. Return geht zum Anfang der nächsten Zeile, ohne die Zeichen rechts vom Cursor zu löschen. Es findet kein Scrolling statt: Der Cursor springt vom Schirmende zum Schirmanfang.

Darstellungsart: Nach Ctrl-I werden weitere Eingaben als INVERSE markiert, nach Ctrl-F als FLASH, nach Ctrl-N wieder als NORMAL. Das jeweilige „Ergebnis" sieht man unmittelbar.

Spaltentransfer: Nach Ctrl-T wird in der Zeile des Cursors das Zeichen der 1. Spalte in die letzte Spalte übernommen. Durch Festhalten von Ctrl-T läßt sich somit die komplette 1. Spalte übertragen. Die Darstellungsart wird dabei berücksichtigt.

Eingabe-Befehl: Ctrl-G markiert die Position für einen GET-Befehl, Ctrl-P für einen INPUT-Befehl. (Als Markierungszeichen werden [und] bzw. Ä und Ü ausgegeben.) Diese Befehle werden später in der zeitlichen Reihenfolge durchlaufen, in der sie eingegeben wurden. Die GET-Befehle werden mit Variablen C0$, C1$, ..., die INPUT-Befehle mit V0$, V1$, ...„programmiert". Die Interpretation dieser String-Variablen (vom Benutzer zu programmieren) sollte im Zielprogramm unmittelbar hinter dem Eingabe-Befehl erfolgen. Maximal sind 256 Eingabe-Anweisungen möglich.

Korrekturen werden durch Überschreiben ausgeführt; gelöscht wird also mit Leerzeichen. Ctrl-W löscht den gesamten Schirm und alle Markierungen.

Der **Generiervorgang** wird durch Ctrl-S gestartet (Cursor-Position ist belanglos!). Mit Ctrl-W plus Ctrl-S kann man jederzeit aus dem Programm aussteigen, ohne daß das Disk-Laufwerk angesprochen wird.

Am Ende der Verarbeitung erhält man Informationen über die „verbrauchten" Zeilennummern; hier kann ohne erneutes Editieren eine Generierung mit veränderten Parametern abgerufen werden. Das ist auch für folgenden Fall wichtig: Die Zeilennummern dürfen beim Applesoft-Interpreter den Wert 63999 nicht überschreiten. Letzteres wird vom Menü-Generator im Einzelfall überprüft und ggf. angezeigt.

Soll das erzeugte Menü-Programm in ein schon vorhandenes Programm eingebaut werden, so ist dies mit LOAD... zu laden und danach dann mit EXEC... das Menü-Programm einzulesen. Das Gesamtprogramm wird dann mit SAVE... gespeichert. (Unter Umständen kann in den einzelnen Phasen ein RENUMBER-Programm nützlich sein.)

Zur Funktion des Generators

Im Programm-Listing sind die einzelnen Funktionsblöcke durch REM-Zeilen überschrieben.
In der Editierphase wird je nach Kontrollzeichen der Cursor bewegt oder die jeweilige Funktion (z.B. Spaltentransfer) ausgeführt. Liegt kein Kontrollzeichen vor, so wird das eingegebene Zeichen in eine 24 * 40 Matrix (SC$) eingetragen, der zugehörige Darstellungstyp in der Matrix DM (0 für NORMAL, 1 für INVERSE, 2 für FLASH) vermerkt.
Alle benutzten Zeilen werden in ZM markiert, die höchste Zeilennummer in ML vorgemerkt. Durch diese Maßnahmen wird die Generierung zum Teil erheblich beschleunigt.
Die Programmierung von „GET" und „INPUT" geschieht durch Eintragen dieser Funktionen in die Tabelle IM (Typ 1 = GET / 2 = INPUT, Zeile, Spalte).
Ein Detail-Problem bedarf noch der Erwähnung: Wird in die rechte untere Bildschirmecke ein Zeichen über PRINT ausgegeben, so schiebt der Apple-Monitor den Bildschirm-Inhalt um eine Zeile nach oben (Scrolling), was hier stets unerwünscht ist. Es läßt sich dadurch vermeiden, daß beim Editieren wie auch beim erzeugten Basic-Programm statt PRINT ein POKE 2039,... verwendet wird. Hierdurch wird das Zeichen für diese spezielle Schirmposition direkt in den Bildschirm-Puffer geschrieben.
Nach Ende der Editierphase werden nun die Bildschirm-Matrix SC$ und parallel dazu die Darstellungsmatrix DM Zeichen für Zeichen abgetastet und in PRINT-Befehle umgesetzt – unterbrochen durch Anweisungen wie VTAB, HTAB, NORMAL, FLASH und INVERSE. Ein besonderes Kapitel ist dabei die Darstellung von Leerzeichen, die ja im Menü-Bild im allgemeinen in großer Zahl auftreten. Sie werden vom Generator am Zeilenanfang mit HTAB, in der Zeilenmitte als String oder über die TAB-Funktion realisiert. Die Optimierungsroutine (Zeile 6000-7000) findet stets die kürzeste Möglichkeit.
Schließlich: Treten Anführungszeichen (") im Menü auf, so werden sie in „CHR$ (34)" umgesetzt. Den Abschluß der Generierung bildet die „Programmierung" der Eingabe-Befehle.

Testbeispiel

Bevor man mit einem Test beginnt, sollte in jedem Fall eine Programmsicherung stattfinden. Zur direkten Überwachung des Generator-Outputs kann man vor dem RUN-Kommando noch MON O eingeben. Nach dem Start antwortet man auf die erste Frage (nach Änderung der Ersatzwerte) mit N (= Nein). Danach gibt man etwa ab 1. Zeile folgendes Testbeispiel ein (Funktionen sind in <...> angegeben, <Ctrl-I> z.B. bedeutet dabei gleichzeitiges Betätigen der Ctrl- und der I-Taste).

```
<return>
<return>
NORMAL<return>
<Ctrl-I>INVERSE<return>
<Ctrl-F>FLASH<return>
<return>
<return>
<Ctrl-N>TEST FÜR GET....:<Ctrl-G>
<return>
<return>
TEST FÜR INPUT..:<Ctrl-P><return>
<return>
* TRANSFER-TEST <Ctrl-T><Ctrl-S>
```

Der Generator-Output sollte dann die folgende Form haben:

```
100 TEXT : HOME
110 VTAB 3: HTAB 1: REM **3**
120 NORMAL : PRINT "NORMAL";
130 VTAB 4: HTAB 1: REM **4**
140 INVERSE : PRINT "INVERSE";
150 VTAB 5: HTAB 1: REM **5**
160 FLASH : PRINT "FLASH";
170 VTAB 8: HTAB 1: REM **8**
180 NORMAL : PRINT "TEST FÜR
    GET....:";
190 VTAB 10: HTAB 1: REM **10**
200 NORMAL : PRINT "TEST FÜR
    INPUT..:";
210 VTAB 12: HTAB 1: REM **12**
220 NORMAL : PRINT "* TRANSFER-
    TEST"; TAB( 40);"*";
230 VTAB 8: HTAB 19: GET C0$: PRINT
    C0$;
240 VTAB 10: HTAB 19: INPUT V0$
```

Zur Kontrolle des Disk-Files gibt man der Reihe nach folgende Anweisungen ein: NEW – EXEC MENUE – LIST. Kontrolle des Menü-Bildes: RUN. Ein erster größerer Test könnte darin bestehen, das Start-Menü des Generators selbst „nachzubauen".
Für sehr detaillierte und umfangreiche Menüs kann die Generierzeit im Bereich von 60 Sekunden liegen. Besitzt man einen Basic-Compiler, so kann diese Zeit erheblich (auf unter 20 Sekunden) reduziert werden.
Bei so vielen Menüs bleibt nur noch „Guten Appetit" zu wünschen.

MENUE.GENERATOR

```
1000 REM ---Startwerte/Parameter---
1010 DIM CB$(19),ZM(24),SC$(24,40),DM(24,40),IM(255,3)
1020 DATA  27,73,74,75,77,8,21,13,20,9,14,19,7,16,91,
     93,34,6,23
1030 FOR I = 1 TO 19: READ X:CB$(I) = CHR$ (X): NEXT
1040 NL = 24:NC = 40:SL = 100:IN = 10:FI$ = "MENUE":
     ST = 6:DR = 1:R$ = CB$(8)
1050 OF(0) = 128:OF(1) = - 64:OF(2) = 0
1060 TX$(0) = "  NORMAL ":TX$(1) = "  INVERSE ":
     TX$(2) = "  FLASH "
1070 GOSUB 7000
2000 REM ---Editierung des Menü-Bildes---
2010 TEXT : HOME :LC = 1:CC = 1:ML = 0:DA = 0:EM = 0:
     NORMAL
2020 GET C$
2030 IF EM = 1 THEN 2110
2040 IF C$ = CB$(1) THEN EM = 1: GOTO 2020
2050 IF ASC (C$) > = 32 THEN 2200
2060 FOR I = 6 TO 19
2070 IF C$ > < CB$(I) THEN 2090
2080 ON I - 5 GOTO 2410,2420,2440,2500,2600,2610,3010,
     2700,2710,2200,2200,2200,2620,2900
2090 NEXT I
2100 PRINT CHR$ (7);: GOTO 2020
2110 FOR I = 1 TO 5
2120 IF C$ > < CB$(I) THEN 2140
2130 ON I GOTO 2020,2400,2410,2420,2430
2140 NEXT I
2150 EM = 0: GOTO 2020
2200 SC$(LC,CC) = C$: IF (C$ > < " " OR DA > < 0) AND
     C$ > < "" THEN ZM(LC) = 1
2210 IF ML < LC THEN ML = LC
2220 DM(LC,CC) = DA
2230 IF LC = 24 AND CC = 40 AND C$ > < "" THEN GOSUB
     9300: POKE 2039,AS:CC = CC + 1: GOTO 2250
2240 PRINT C$;:CC = CC + 1
2250 IF CC < = 0 THEN LC = LC - 1:CC = NC
2260 IF CC > NC THEN LC = LC + 1:CC = 1
2270 IF LC < = 0 THEN LC = NL
2280 IF LC > NL THEN LC = 1
2290 IF TM = 1 THEN CC = CC + 1:TM = 0
2300 HTAB CC: VTAB LC: GOTO 2020
2400 LC = LC - 1: GOTO 2250
2410 CC = CC - 1: GOTO 2250
2420 CC = CC + 1: GOTO 2250
2430 LC = LC + 1: GOTO 2250
2440 LC = LC + 1:CC = 1: GOTO 2250
2500 C$ = SC$(LC,1):DA = DM(LC,1): HTAB NC:CC = NC:TM = 1
2510 GOSUB 2800: GOTO 2200
2600 DA = 1: GOTO 2630
2610 DA = 0: GOTO 2630
2620 DA = 2
2630 GOSUB 2800: GOTO 2020
2700 GOSUB 2730:IM(IC,0) = 1:C$ = CB$(15): GOTO 2720
2710 GOSUB 2730:IM(IC,0) = 2:C$ = CB$(16)
2720 IM(IC,1) = LC:IM(IC,2) = CC:IC = IC + 1: GOTO 2240
2730 IF IC < 256 THEN RETURN
2740 PRINT CHR$ (7);:C$ = " ": POP : GOTO 2240
2800 IF DA = 0 THEN NORMAL : RETURN
2810 IF DA = 1 THEN INVERSE : RETURN
2820 FLASH : RETURN
2900 HOME : VTAB 12: HTAB 14: NORMAL : PRINT
     "CLEARING..."
2910 FOR I = 1 TO ML:ZM(I) = 0
2920 FOR J = 1 TO 40:SC$(I,J) = "":DM(I,J) = 0: NEXT J:
     NEXT I:IC = 0: GOTO 2010
3000 REM ---Generierung der Basic Zeilen---
3010 TEXT : HOME : IF ML = 0 AND IC = 0 THEN END
3020 VTAB 14: HTAB 8: FLASH :
     PRINT "B I T T E  W A R T E N ";:IZ(0) = 0:IZ(1) = 0
3030 NORMAL : VTAB 18: HTAB 8:
     PRINT "ZEILE 00 WIRD BEARBEITET"
3040 NORMAL : GOSUB 9500:PK = 0:DA = - 1:LN = SL
3050 PRINT R$;LN;" TEXT:HOME";:LN = LN + IN
3055 IF ML = 0 THEN 4510
3060 FOR I = 1 TO ML:: IF ZM(I) = 0 THEN 3230
3070 Z1 = INT (I / 10):Z2 = I - Z1 * 10: POKE 1245,176 +
     Z1: POKE 1246,176 + Z2
3080 GOSUB 5010: IF N2 < N1 THEN 3230
3090 IF I = 24 AND N2 = 40 THEN N2 = 39:PK = 1:
     IF N1 = 40 THEN 3230
3100 PRINT R$;LN;"VTAB";I;":HTAB";N1;":REM**";I;"**";:
     LN = LN + IN
3110 PC = 0:PM = 0:BC = 0:QC = 0
3120 FOR J = N1 TO N2
3130 IF DA > < DM(I,J) THEN GOSUB 6100:PM = 0:QC = 0:
     DA = DM(I,J): PRINT R$;LN;TX$(DA);:LN = LN + IN
3140 C$ = SC$(I,J)
3150 IF PM = 0 THEN PRINT ": PRINT ";:PM = 1
3160 IF C$ = " " OR C$ = "" THEN BC = BC + 1: GOTO 3200
3170 IF C$ = CB$(17) THEN GOSUB 6100: GOTO 6410
3180 ON PC + 1 GOSUB 6200,6300
3190 PRINT C$;:PC = 1
3200 NEXT J
3210 DL = DA:DA = - 1
3220 GOSUB 6100
3230 NEXT I
4000 REM ---Zeile 24, Spalte 40---
4010 IF PK = 0 THEN 4040
4020 DA = DM(24,40):C$ = SC$(24,40): GOSUB 9300
4030 PRINT R$;LN;" POKE 2039,";AS;:LN = LN + IN
4040 IF DL > < 0 THEN PRINT R$;LN;TX$(0);:LN = LN + IN
4500 REM ---Programmierung der Eingabe-Befehle---
4510 IF IC = 0 THEN 4590
4520 FOR I = 0 TO IC - 1
4530 ID = IM(I,0) - 1
4540 PRINT R$;LN;"VTAB";IM(I,1);":HTAB";IM(I,2);":";
4550 IF ID = 0 THEN PRINT " GET C";IZ(0);"$:
     PRINT C";IZ(0);"$";:: GOTO 4570
4560 PRINT "  INPUT V";IZ(1);"$";
4570 IZ(ID) = IZ(ID) + 1:LN = LN + IN
4580 NEXT I
4590 PRINT : PRINT D$;"CLOSE": GOSUB 8000: END
5000 REM ---Zeile eingrenzen---
5010 N1 = NC:N2 = 1
5020 FOR N = 1 TO NC
5030 S1$ = SC$(I,N)
5040 IF S1$ > < "" AND (S1$ > < " " OR DM(I,N) > < 0)
     THEN N1 = N: GOTO 5060
5050 NEXT N
5060 FOR N = NC TO 1 STEP - 1
5070 S2$ = SC$(I,N)
5080 IF S2$ > < "" AND (S2$ > < " " OR DM(I,N) > < 0)
     THEN N2 = N: GOTO 5100
5090 NEXT N
5100 RETURN
6000 REM ---Abschließen/Optimieren der Strings---
6100 IF PC = 1 THEN 6150
6110 IF BC = 0 THEN 6170
6120 IF BC < = 4 THEN PRINT CB$(17); SPC(BC);CB$(17);";";
6130 IF BC > 4 THEN PRINT "TAB(";J;");";
6140 GOTO 6170
6150 IF BC < = 7 THEN PRINT SPC(BC);CB$(17);";";
6160 IF BC > 7 THEN PRINT CB$(17);";TAB(";J;");";
6170 BC = 0:PC = 0: RETURN
6200 IF BC < = 7 THEN PRINT CB$(17); SPC(BC);
6210 IF BC > 7 THEN PRINT "TAB(";J;");";CB$(17);
6220 BC = 0: RETURN
6300 IF BC < = 10 THEN PRINT SPC(BC);
6310 IF BC > 10 THEN PRINT CB$(17);";TAB(";J;");";
     CB$(17);
6320 BC = 0: RETURN
6400 REM ---Bearbeiten von >"<---
6410 QC = QC + 1: IF QC < 20 THEN PRINT "CHR$(34);";:
     GOTO 3200
6420 QC = 0: PRINT R$;LN;"  PRINT CHR$(34);";:LN = LN + IN
6430 GOTO 3200
7000 HOME : REM ---Menü 1---
7010 INVERSE : PRINT SPC(39);: GOSUB 9100: PRINT SPC(10);
7020 INVERSE : PRINT "MENUE-GENERATOR";: GOSUB 9100
7030 PRINT "   Änderung der Ersatzwerte (J/N)?    ";:
     GOSUB 9100
7040 PRINT "   Ersatzwerte sind :";: GOSUB 9100
7050 PRINT "   Bildschirm/Zeilen  : ";: INVERSE :
     PRINT NL;: GOSUB 9200
7060 PRINT SPC(14);"Spalten : ";: INVERSE : PRINT NC;:
     GOSUB 9100
7070 PRINT "   Basic-Programm/";: GOSUB 9200
7080 PRINT SPC(10);"1.Zeilennr. : ";: INVERSE :
     PRINT SL;: GOSUB 9200
7090 PRINT SPC(10);"Inkrement    : ";: INVERSE :
     PRINT IN;: GOSUB 9100
7100 PRINT "   Menü-(Exec-)File/";: GOSUB 9200
7110 PRINT SPC(16);"Name  : ";: INVERSE : PRINT FI$;:
     GOSUB 9100
7120 PRINT SPC(16);"Slot  :   ";: INVERSE : PRINT ST;:
     GOSUB 9200
7130 PRINT SPC(16);"Drive :   ";: INVERSE : PRINT DR;:
     GOSUB 9100
7140 INVERSE : PRINT SPC(39);: NORMAL
7150 VTAB 5: HTAB 37: GET C0$: PRINT C0$;
7160 IF C0$ = "N" THEN RETURN
```

MICROMINT

MICROMINT VOLLTREFFER

LASAR 16
– IBM Comp. 64 K, Contr. Bootrom
TEAC FD 55 B
Netzteil 15 A **3.800;**

LASAR ZE
– Apple comp. 64 K +
12 K ROM +
6502 + Z 80 A **1.365;**

Außerdem volles Rückgaberecht innerhalb 14 Tagen ohne Begründung.

	Apple	IBM
● Mehrzweckgehäuse lt. Abb.	209,–	209,–
● Schaltnetzteile Apple 5 A/IBM 15 A	115,–	350,–
● Profitastatur dtsch. LASAR 2000	365,–	365,–
● Interface ab	95,–	400,–
● Drucker Epson komp.!!!	898,–	898,–

Prompte Belieferung von 100 m² Lagerfläche. Kostenlose Tiefstpreishändlerliste noch heute schriftlich anfordern – großes Angebot an IBM-Comp.

Generalimporteur MICROMINT Computer GmbH
Hochdahler Straße 151, 4006 Erkrath 2
Telex 8 589 305 mcm

☎ 02104/33024

ccp datentechnik

640 KByte-Drives für Apple II (//e)
- $5^1/_4$- od. $3^1/_2$-Zoll-Format (Teac FD55/35-F)
- Umschaltbar auf 40 Track (Apple kompatibel)
- Patch für DOS 3.3, UCSD-Pascal, CP/M 2.2, CP/M 2.23 (60K), PRODOS, AP22, ALS CP/M+
- Umfangreiches Manual
- Anschlußfertige Auslieferung
- Ehring-Contr.-Kabel u. 2 Drive (1.2 MB) . . . . **DM 1775,–**

Festplatten für Apple II (//e)
- $5^1/_4$ Zoll-Format (Slimline)
- Booten direkt von der Festplatte
- in DOS 3.3, UCSD-Pascal, PRODOS und CP/M 2.2
- läuft auch mit unseren 160 Track-Drives
- Copy-Programme im Lieferumfang
- Umfangreiches Manual
- Festplatte (5 – 11,2 MB netto) incl. Controller, Netzteil und Patchsoftware . . . . . ab **DM 3250,–**

Akustikkoppler für Apple II (//e)
- Serielle und Modem auf einer Karte
- Hörerverbindung wird mit Kabel herausgeführt
- 300 u. 600 Baud voll- u. halbduplex
- Originate und Answermode
- CCITT-Norm (V21) u. Bell 103 (DFÜ weltweit)
- ASCII-Express, Modem 7 usw. lauffähig
- Leerplatine incl. ROM, Bausatz oder Fertiggerät
- Fertiggerät + Manual . . . . . **DM 398,–**

Alles für Ihren Apple

Info bei:

ccp datentechnik
Herderstraße 12 – 2000 HH 76
Telefon 0 40/22 56 76

ZUSATZ-KARTEN:

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

Floppy-Controller

FDC 4 für alle Laufwerke . . . . . 199,– Bausatz wie links 179,–
Leerplatine wie oben incl. Prom u. Eprom 130,–

Autopatch-Controller (Erphi Controller)

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

Joy Stick De Luxe . . . . . 59,– **Netzteil 5A** 149,–

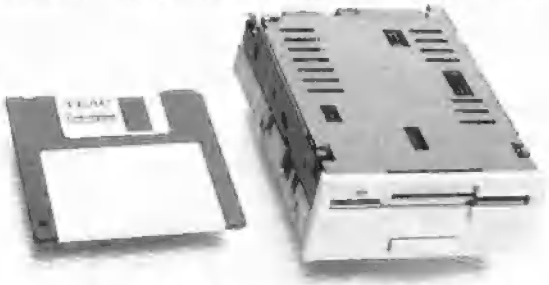
Halbschalen-Gehäuse für 2 Slimline-Laufwerke 65,–
Gehäuse für 1 5¼" Slimline Laufwerk 39,–
Gehäuse für 2 5¼" Slimline Laufwerke mit Platz für ein Netzteil 159,–
IBM®-Gehäuse 229,–
Floppy-Kabel 34pol. für 2 Laufwerke mit Shugart-Bus 35,–

Preh Commander Keyboards

Wir bieten Ihnen die **Preh-Qualität** auch für Apple. AK 88 spez. mit Gehäuse, Anschlußkabel, Zehner-Tastenfeld, dt. Zeichensatz, Sondertasten für Ctrl-Codes und Rechenfunktionen **339,–**

TEAC 3½" Laufwerk FD 35 F 579,–

Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur **579,–**



Nochmalige Preissenkung bei TEAC:

TEAC FD 55 A 1 x 40 Track	498,–	**TEAC FD 55 B** 2 x 40 Track	579,–
TEAC FD 55 E 1 x 80 Track	535,–	**TEAC FD 55 F** 2 x 80 Track	638,–

Philipps Slimline Floppy X3134 A, 2x80 Track solange Vorrat . . . . . nur **638,–**
SONY 3½" Laufwerk . . . . . nur **799,–**
Apple®-kompatibles Laufwerk incl. Gehäuse + Kabel **599,–**

EPSON DRUCKER

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

Patch-Diskette für **SONY 3½" Laufwerke** – ermöglicht die Anpassung an II/IIe und kompatible Computer . . . . . **80,–** Manual vorab 15,-DM (wird beim Kauf der Patch-Diskette angerechnet).
Disketten Döb&Böd SS/DD 10St. **62,–** **dto.** für 5¼" Laufwerke **69,–**
Akustikkoppler AK 300 mit FTZ-Nr. incl. Netzteil . . . . . **549,–** **10 Disketten** f.Sony 3½" Laufw. . **150,–**

Die Microfloppy mit Zukunft:

Speicherkapazität: 2 x 1 MByte formatiert: 2 x 640 kByte. Anschlußfertig mit PROM-residenter Patchsoftware für CP/M 2.2, Apple DOS 3.3, Diversi-DOS 2-C, 4-C (DD MOVER), Apple Pascal 1.1, Pascal 1.2, Pro-DOS 1.0.1, 1.1, 1.1.1 zum Preis von **1640,–**
Low Power Version 1740,–

10 MB Winchester 4490,–

incl. Controler, Software und Gehäuse **4490,–**

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

Ueding electronics

Holtewiese 2
5750 Menden 1

DFÜ 02373/66877
Tel. 02373/63159

```
7170 IF C0$ > < "J" THEN PRINT CHR$ (7);: GOTO 7150
7180 POKE 32,1: POKE 33,38
7190 VT = 9:UL = 1:V0 = NL:0L = 24: GOSUB 9400:NL = V0
7200 VT = 10:UL = 1:V0 = NC:0L = 40: GOSUB 9400:NC = V0
7210 VT = 13:UL = 1:V0 = SL:0L = 63999: GOSUB 9400:SL = V0
7220 VT = 14:UL = 1:V0 = IN:0L = 63998: GOSUB 9400:IN = V0
7230 VTAB 18: HTAB 38: PRINT "*";
7240 VTAB 18: HTAB 7: INPUT V4$
7250 L = LEN (V4$): IF L > 30 THEN PRINT CHR$ (7);:
     GOTO 7000
7260 IF L = 0 THEN 7290
7270 L = ASC ( LEFT$ (V4$,1)): IF L < 65 OR L > 90 THEN
     PRINT CHR$ (7);: GOTO 7000
7280 FI$ = V4$
7290 VT = 19:UL = 4:V0 = ST:0L = 6: GOSUB 9400:ST = V0
7300 VT = 20:UL = 1:V0 = DR:0L = 4: GOSUB 9400:DR = V0
7310 RETURN
8000 HOME : REM ---Menü 2---
8010 INVERSE : PRINT SPC(39);: GOSUB 9100:
     PRINT SPC(11);
8020 INVERSE : PRINT "MENUE-GENERATOR";
8030 GOSUB 9100: PRINT "  File-Name:";
8040 GOSUB 9100: HTAB 4: PRINT FI$;",S";ST;",D";DR;
8050 GOSUB 9100: PRINT "  Zeilennummern: ";SL;" bis ";
     LN - IN;: GOSUB 9200
8060 IF LN - IN > 63999 THEN NORMAL : PRINT TAB(19);:
     FLASH : PRINT "NICHT ZULAESSIG";
8070 GOSUB 9200: PRINT "  Generierung wiederholen(J/N)? ";
8080 GOSUB 9200: GOSUB 9100: PRINT "    1.Zeilennummer :";
8090 INVERSE : PRINT SL;
```

```
8100 GOSUB 9200: PRINT "    Inkrement      : ";: INVERSE :
     PRINT IN;: GOSUB 9100
8110 INVERSE : PRINT SPC(39);: NORMAL
8120 VTAB 11: HTAB 34: GET C0$: PRINT C0$;
8130 IF C0$ = "N" THEN VTAB 20: RETURN
8140 IF C0$ > < "J" THEN PRINT CHR$ (7);: GOTO 8120
8150 POKE 33,39
8160 VT = 14:UL = 1:V0 = SL:0L = 63999: GOSUB 9400:SL = V0
8170 VT = 15:UL = 1:V0 = IN:0L = 63998: GOSUB 9400:IN = V0
8180 POP : GOTO 3010
9000 REM ---Subroutinen---
9100 NORMAL : PRINT TAB(40);: INVERSE : PRINT "  ";:
     NORMAL : PRINT SPC(38);: INVERSE : PRINT "  ";:
     NORMAL : RETURN
9200 NORMAL : PRINT TAB(40);: INVERSE : PRINT "  ";:
     NORMAL : RETURN
9300 XX = OF(DA): IF DA = 1 AND ASC (C$) < = 64 THEN XX = 0
9310 AS = ASC (C$) + XX: RETURN
9400 VTAB VT: HTAB 30: INPUT V$: IF V$ = "" THEN RETURN
9410 V1 = VAL (V$): IF V1 < UL OR V1 > 0L THEN
     PRINT  CHR$ (7);: GOTO 9400
9420 V0 = V1: RETURN
9500 PRINT :D$ = CHR$ (4)
9510 PRINT D$;"OPEN";FI$;",S";ST;",D";DR
9520 PRINT D$;"UNLOCK";FI$
9530 PRINT D$;"DELETE";FI$
9540 PRINT D$;"OPEN";FI$
9550 PRINT D$;"WRITE";FI$
9560 RETURN
```

Peeker-Sammeldisketten

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

Disk# 1
(Heft 1+2, 1984)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

T.VIDEXT
VIDEXT.LISA.SOURCE
VIDEXT

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

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

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

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

Disk #4
(Heft 3-4, 1985)

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

MULTIPRECISION (3/85, S. 32)

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

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

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

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

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

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

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

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

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

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

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

peeker Börse

Verkauf Software

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

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

Profess. Daten-, Text-, **Grafikverarbeitung,** universell einsetzbar, DOS 3.3 Disc: 159 DM, Info gratis, Demodisc und Gesamtangebot: 10 DM G. Basse, Vahlenhorst 9, 29 Oldenburg

Profi-Rechnungsprogramm (99 Pos.) DM 89,–. Info: Florian Kopitzki, Ameisenbergstr. 57 c, 7000 Stuttgart 1

APPLE II-Programmgenerator für Eingabemasken bzw. Felder 59,– Dateiverwaltung m. Editor, Grafik-Listenpaket 65,–, Bstllg. gg. Vorkasse, Info –,80: Fahrenberger, Rittershausstr. 4, 5300 Bonn 1

Für Ihre Kinder/Schüler (8-12 J.)! Disk. über **Grundrechnungsarten.** Praxiserprobt! DM 59,–. Lehrer G. Eiler, Hof 16, A-6951 Lingenau

Ankauf Software

GraForth Animation Guide, Trans-Forth ges., Chiffre P 1000

Suche **Patch/Treiber** für Profile (Apple-Festplatte) unter CP/M Chiffre P 1001

Verkauf Hardware

Zu wenig **SLOTS?** Slot Expander liefert 8 weitere Slots, per Software umschaltbar, Platinenset (2), durchkontaktiert, incl. Beschreibung: DM 180,–, Info: E. Tausendpfund, Gonsenheimer Spieß 18, 65 Mainz.

Sprachausgabe-Interface *** Info: Zotter, Triberg 15, 7030 Böblingen

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

IBM-Gehäuse für Apple DM 180,– Controller (1×35 – 2×80) DM 200,– beides neuw., Tel. 08191/5817 ab 18h

Verkaufe: **2 BASF Floppys** 6106 im AKA Doppelgehäuse DM 600,–; Tel. 0551/91363

Verkaufe **APPLE**-Kompa. mit Zubehör, Preis VB, Tel. 02384/3909 nach 19.00 Uhr

68000-KARTE IBS APZO mit ASM-SW. 990,– DM, Klaus Schmidt, Dörnerhof, 4100 Duisburg 1, (0203) 341729

Farbgraphik-Interface für NEC 8023 bzw. Itoh 8510 SCP nur 250,– DM, Tel. 02422/74 20

APPLE II comp. mit 64 k 6502 und Z80 CPU im IBM Gehäuse und separater Tastatur nur DM 1398,– noch heute Liste anfordern über weitere Hardware bei
B. Kraus, Rohrersmühlstr. 21, 8540 Schwabach

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

Disketten, 51/4", ssldd, 10 Stück 37,– DM, Tel. 0911/460681

Ankauf Hardware

Suche **Patch/Treiber** für Profile (Apple-Festplatte) unter CP/M Chiffre P 1001

SUCHE II e/c/+ oder MAC und Drucker und Software (auch einzeln), Tel. 0421/701771

Erscheinungs- und Anzeigenschlußtermine für peeker

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

Inserentenverzeichnis peeker 4/85

Für Ihre Unterlagen

Abonnement bestellt

am: _____

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

peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg 1

PEEKER
MAGAZIN FÜR APPLE-COMPUTER

Abo-Karte

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

Ich wünsche jährliche Berechnung

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

Bank/PschA _____

Bankleitzahl _____ Kto.-Nr. _____

Datum _____ Unterschrift _____

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER
MAGAZIN FÜR APPLE-COMPUTER

Buch-Shop

Bitte senden Sie mir gegen Rechnung folgende Bücher:

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

Datum _____ Unterschrift _____

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER
MAGAZIN FÜR APPLE-COMPUTER

Software-Karte

Bitte senden Sie mir
gegen Rechnung folgende Apple-Programme:

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

Datum _____ Unterschrift _____

Abo-Karte

Name

Firma

Abteilung

Straße

PLZ/Ort

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

Datum

Unterschrift

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

POSTKARTE

peeker
Leserservice

Postfach 10 28 69

6900 Heidelberg 1

Buch-Shop

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

peeker
Versandbuchhandlung

Postfach 10 28 69

6900 Heidelberg 1

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

peeker
Softwareabteilung

Postfach 10 28 69

6900 Heidelberg 1

Die Fachzeitschrift, die umfassend aus allen Teilbereichen des CAE berichtet

CAE-Journal für Leser, die ständig informiert sein müssen über:

- neue Entwicklungen und Trends
- neue Technologien
- neue Methoden und Verfahren
- neue Applikationen
- Wirtschaftliches und Aktuelles aus allen Bereichen des Computereinsatzes in der Technik

Hüthig
PUBLIKATION

Coupon

Sie möchten das CAE-Journal kennenlernen? Dann machen Sie von unserem nebenstehenden Kennenlern-Angebot Gebrauch

Ja, ich möchte **CAE**-Journal kennenlernen und prüfen. Sie schicken mir die neueste Ausgabe kostenlos zu. Wenn ich 20 Tage nach Erhalt meines Probeheftes nichts von mir hören lasse, liefern Sie bitte die Zeitschrift bis auf Widerruf im Abonnement zum Jahresbezugspreis von DM 60,– inkl. Mwst., zzgl. Versandspesen für 6 Ausgaben 1985. Kündigungen sind bis 8 Wochen vor Jahresende schriftlich an den Verlag zu richten

Name, Vorname

Firma

Abtlg.

Straße

PLZ/Ort

Datum Unterschrift

Wichtig: Wenn ich **CAE**-Journal nicht regelmäßig lesen möchte, schicke ich innerhalb von 20 Tagen eine kurze Mitteilung an den Verlag und alles ist für mich erledigt

Datum Unterschrift

Coupon ausgefüllt einsenden an:

CAE-Journal, Leserservice
Postfach 10 28 69
D-6900 Heidelberg 1

Makros für die neuen Befehle des 65C02

von Dipl.-Oec. Edgar Meyzis

Nachfolgend wird beschrieben, wie Makrobefehle für Assembler des Typs MERLIN und BIG MAC entwickelt werden können. Im „Peeker", Heft 1/1984 wurden die zusätzlichen Befehle der CPU 65C02 und ein entsprechender Disassembler vorgestellt. Die Darstellung knüpft daran an und bietet eine Möglichkeit, den erweiterten Befehlssatz der CPU mit Hilfe von Macros komfortabel zu nutzen. Inzwischen gibt es einige Assembler, die die neuen 65C02-Opcodes automatisch unterstützen, und zwar „Merlin.Pro" (z.Zt. noch nicht in Deutschland erhältlich) oder „SC-Macroassembler" (über Fa. Karl-Heinz Weiß, 2940 Wilhelmshaven, Am Wiesenhof 17, erhältlich). Da sich jedoch nicht jeder wegen der neuen Opcodes einen neuen Assembler kaufen will, sind die nachfolgenden Macrobefehle eine nützliche Ergänzung.

1. Struktur von Makrobefehlen

Bevor auf die Makrobefehle für den 65C02 eingegangen wird, ist es zweckmäßig, sich kurz mit ihrer Grundstruktur und der Implementierung auseinanderzusetzen. Die erste und die letzte Zeile eines Makrobefehls ist an folgende Form gebunden:

```
LABEL MAC

        EOM bzw. <<<
```

Die erste Zeile enthält den Namen (LABEL) des Befehls und den Hinweis (MAC), daß die Definition eines Makros folgt. Mit „EOM" (End of Macro) bzw. „<<<" ist eine Definition abzuschließen. Makrobefehle werden in einem Assemblerprogramm wie folgt aufgerufen:

```
        PMC LABEL bzw. >>> LABEL
```

Der Aufruf wird durch den Assembler an der Buchstabenfolge „PMC" (Put Macro) bzw. an den drei Rechtspfeilen (Größer-Zeichen) erkannt. Als LABEL können deshalb durchaus Zeichenfolgen verwendet werden, die – allein eingesetzt – für den Assembler eine spezifische Bedeutung haben (z.B. Mnemonics für Operation Codes). Sind bei Aufruf eines Makrobefehls Parameter zu übergeben, so ist dem LABEL eine Parameterliste anzufügen:

```
>>> LABEL.Parameter;...;Parameter
```

Es können bis zu acht Parameter übergeben werden. Mit ihrer Spezifizierung wird zugleich eine entsprechende Anzahl lokaler Variablen deklariert. Sie nehmen die Werte der übergebenen Parameter in der Reihenfolge ihrer Nennung an. Die Bezeichnungen der Variablen sind vorgegeben:

```
]1....]8     Amerikanischer Zeichensatz
Ü1.....Ü8    Deutscher Zeichensatz
```

Die Parameter, die in die Variablen eingehen, müssen vor Aufruf der jeweiligen Makrobefehle definiert sein, um uneingeschränkt korrekte Ergebnisse zu erhalten. Deshalb ist es im allgemeinen unzweckmäßig, in der Parameterliste LABELs aufzuführen, die erst später im Programm folgen. Es besteht jedoch eine wichtige Ausnahme: Standardanweisungen an den Assembler für die Erzeugung von Objektcode (z.B. LDA LABEL) werden auch dann stets genau ausgeführt, wenn das LABEL sich auf eine spätere Stelle im Programm bezieht. Von dieser Möglichkeit wurde bei der Erarbeitung der Makrobefehle für den 65C02 mehrfach Gebrauch gemacht.
Bei der Definition von Makrobefehlen ist es häufig erforderlich, die folgende Kontrollstruktur einzusetzen, um eine konditionale Assemblierung zu erreichen:

```
DO Ausdruck
ELSE
FIN
```

Nimmt der Ausdruck einen Wert ungleich null an, so wird der von DO und ELSE eingeschlossene Programmteil ausgeführt. Ist hingegen der Ausdruck null, so werden die Instruktionen zwischen ELSE und FIN abgearbeitet.
Den allgemeinen Teil abschließend bleibt noch zu erwähnen, daß Makrobefehle den Anfang eines Quellprogramms bilden müssen. Sie sind in ihrer Gesamtheit von „DO 0" und „FIN" einzuschließen, um zu vermeiden, daß für sie bei der Assemblierung Objektcode erzeugt wird.

2. Anforderungen an 65C02-Macrobefehle

Wir können uns nun der konkreten Aufgabenstellung zuwenden, d.h. der Definition von Makrobefehlen, die die Operation Codes für den erweiterten Befehlssatz des 65C02 erzeugen. An die zu erarbeitenden Makros sind folgende Anforderungen zu

stellen:
- als LABEL möglichst standardisierte Mnemonics verwenden
- übergebene Parameter formal auf Richtigkeit prüfen
- ggf. Fehlermeldungen generieren und Assemblierung abbrechen
- Bezugnahme auf folgende Programmteile unter Angabe von LABEL ermöglichen.

Die Forderungen werden von den vorliegenden Makrobefehlen erfüllt. Dazu ist es teilweise erforderlich, Fehler zu simulieren, um Meldungen zu erzeugen, die die tatsächliche Situation richtig beschreiben.

3. Die Makrobefehle für den 65C02

Das beigefügte Listing enthält die Makrobefehle in alphabetischer Reihenfolge. Der Definition der Befehle liegen fünf Muster zugrunde, die sich hauptsächlich durch ihren Grad an Komplexität unterscheiden. Je ein Befehl aus den sich so bildenden fünf Gruppen wurde exemplarisch ausführlich kommentiert, um das Vorgehen zu verdeutlichen. Bei Einarbeitung in die Aufgabenstellung wäre es von Vorteil, mit der Analyse der einfachsten Befehle zu beginnen und den Beispielen zu folgen.

Beispiel 1: DECREMENT ACCUMULATOR

Es bietet sich an, z.B. die Definition des Makrobefehls „DEA" zu betrachten: Das Makro fügt in den Objektcode lediglich das Byte „3A", den Operation Code für „DEA" ein. Dazu werden keine Variablen benötigt; die Parameterliste bleibt somit leer.

Beispiel 2: BRANCH ALWAYS

Als typisches Beispiel für das nächste Muster dient der Befehl „BRA", der wie folgt aufzurufen ist:

>>> BRA.ADDR bzw. >>> BRA.LABEL

Als Parameter ist die Adresse durch ihren Zahlenwert oder symbolisch anzugeben (z.B. $8050 oder ZIEL). Zunächst wird ein bedingter Sprung zu der Adresse simuliert, die die Variable „]1" enthält. Dabei wird die maximal mögliche Sprungdistanz beachtet. Falls erforderlich, wird eine entsprechende Fehlermeldung erzeugt. Durch Simulation eines bedingten Sprunges ist es möglich, zu Programmteilen zu verzweigen, die erst später folgen.

Beispiel 3: ADD WITH CARRY INDIRECT

Das dritte Muster wird durch den Befehl „>>> ADCI.ZP-ADDR" repräsentiert. Als Parameter ist eine Adresse auf der Zero-Page zu übergeben. Durch die Instruktion „ADC (]1,X)" wird die Einhaltung dieser Bedingung geprüft und zugleich die Adresse im Objektcode gesetzt. Es ist anschließend nur noch erforderlich, den Operation Code für die indirekte Addition einzusetzen. Der Makrobefehl weicht von dem standardisierten Mnemonic ab, um das Lesen des Quellprogramms zu erleichtern.

Beispiel 4: STORE ZERO WITH INDEX

Das vierte Grundmuster ist in dem Befehl „>>> STZX.ADDR" zu erkennen. Vor der Generierung von Objektcode wird mit der Variablen „]ANFG" die aktuelle Adresse des Programmzeigers festgehalten. Anschließend wird ein Speicherzugriff (STA]1,X) simuliert und die Länge des Codes (]BYTES) festgestellt. Je nach Adresse kann es sich nur um einen Zweibyte- oder um einen Dreibytebefehl handeln. Dieser Sachverhalt bestimmt die folgende bedingte Assemblierung. Ergibt der Ausdruck „]BYTES-2" den Wert null, so handelt es sich um einen Zweibytebefehl; somit ist der Operation Code „74" zu setzen. Liegt hingegen die übergebene Adresse nicht auf der Zero-Page, so ist der Code „9E" einzufügen.

Beispiel 5: BRANCH ON BIT RESET

Der Befehl „>>> BBR.0-7;ZP-ADDR; ZIEL" spiegelt das fünfte und letzte Muster wider. Der erste Parameter darf nur im Wertebereich von 0 bis 7 liegen. Dies wird durch Auswertung des Ausdrucks „]1&FFF8" geprüft, indem die Variable „]1" durch die logische Operation „AND" mit der Konstanten „$FFF8" verknüpft wird. Liegt der Wert der Variablen außerhalb der angegebenen Grenzen, so wird eine Fehlermeldung generiert. Aus dem ersten Parameter ist der Operation Code entsprechend des zu testenden Bits zu gewinnen. Dazu wird das untere Halbbyte in das obere durch eine Multiplikation mit 16 (= $10) „geschoben"; anschließend wird das untere Halbbyte durch die logische Operation „OR" (.) mit „$F" gefüllt (BBR1 EQU]1*$10.$0F).

Es soll nicht unerwähnt bleiben, daß die Anwendung der Makrobefehle auch einen kleinen Haken hat. Die Übersichtlichkeit der Assemblerlistings leidet darunter, daß vorläufige Operation Codes gesetzt und anschließend substituiert werden. Der Zugewinn an Komfort bei der Programmierung gleicht diesen Mangel jedoch aus.

```
1   ************************************
2   *                                  *
3   *     MACROS zur Unterstützung     *
4   *    der neuen Befehle des 65C02   *
5   *    von Edgar Meyzis, Nov. 1984   *
6   *    (für Big Mac bzw. Merlin)     *
7   ************************************
8   *
9           DO  Ø
10  *
11  * ADC INDIRECT    OP CODE 72
12  *
13  * Syntax: >>> ADCI.ZP-ADDR o.
14  *         >>> ADCI.ZP-LABEL
15  *
16  ADCI    MAC
17  *
18  * ADDR u. vorläufigen OP CODE setzen;
19  * Fehlermeldung, wenn ADDR n. auf ZP
20  *
21          ADC  (Ü1,X)
22  *
23  * Zeiger f. Code zurück auf ADC-Byte
24  *
25          DS   -2
26  *
27  * vorläufigen OP CODE ersetzen
28  *
29          HEX  72          ; ADC IND
30  *
31  * Zeiger vorsetzen, hinter ADDR
32  *
33          DS   1
34          <<<
35  *
36  * AND INDIRECT       OP CODE 32
37  *
38  * Syntax: >>> ANDI.ZP-ADDR o.
39  *         >>> ANDI.ZP-LABEL
40  *
41  ANDI    MAC
42  *
43          AND  (Ü1,X)      ; AND IND
44          DS   -2
45          HEX  32
46          DS   1
47          <<<
48  *
49  * BBR [#(Ø-7)]     OP CODES ØF-7F
50  *
51  * Syntax: >>> BBR.Ø-7;ZP-ADDR;ADDR
52  *         >>> BBR.Ø-7;ZP-LABEL;LABEL
53  *
54  BBR     MAC
55  *
56  * Byte für OP CODE reservieren
57  *
58          DS   1
59  *
60  * Sprungdistanz prüfen und setzen,
61  * ggf. BAD BRANCH melden
62  *
63          BNE  Ü3
64  *
65  * Zeiger zurück auf Byte f. OP CODE
66  *
67          DS   -3
68  *
69  * Ist der erste Operand kleiner 8 ?
70  *
71          DO   Ü1&$FFF8
72  *
73  * nein, somit BAD OPERAND simulieren
74  *
75          JMP  $#Ø
76          ELSE
77  *
78  * liegt ADDR auf der ZPG? Ggf. Fehler
79  * melden; falls o.k., ADDR setzen
80  *
81          LDA  (Ü2,X)
82  *
83  * Zeiger auf Byte für OP CODE setzen
84  *
85          DS   -2
86  *
87  * OP CODE berechnen; dazu unteres
```

```
88   * Halbbyte i. d. obere schieben und
89   * anschl. unt. Halbbyte $F setzen
90   *
91   BBR1     EQU  Ü1*$1Ø.$ØF
92   *
93   * OP CODE setzen
94   *
95            DFB  BBR1          ; BBR
96            FIN
97            DS   1
98            <<<
99   *
1ØØ  * BBS [#(Ø-7)]      OP CODES 8F-FF
1Ø1  *
1Ø2  * Syntax: >>> BBS.Ø-7;ZP-ADDR;ADDR
1Ø3  *          >>> BBS.Ø-7;ZP-LABEL;LABEL
1Ø4  *
1Ø5  BBS      MAC
1Ø6           DS   1
1Ø7           BNE  Ü3
1Ø8           DS   -3
1Ø9           DO   Ü1&$FFF8
11Ø           JMP  $#Ø
111           ELSE
112           LDA  (Ü2,X)
113           DS   -2
114  BBS1     EQU  8+Ü1*$1Ø.$ØF
115           DFB  BBS1          ; BBS
116           FIN
117           DS   1
118           <<<
119  *
12Ø  * BIT IMMEDIATE   OP CODE 89
121  *
122  * Syntax: >>> BIT.$NUMBER
123  *          >>> BIT.LABEL
124  *
125  BIT      MAC
126           DO   Ü1&$FFØØ
127           JMP  $#Ø
128           ELSE
129           LDA  Ü1
13Ø           DS   -2
131           HEX  89          ; BIT IMM.
132           DS   1
133           FIN
134           <<<
135  *
136  * BIT  ABS,X / ZPG,X  OP CODES 3C, 34
137  *
138  * Syntax: >>> BITX.ADDR o.
139  *          >>> BITX.LABEL
14Ø  *
141  BITX     MAC
142  ÜANFG    EQU  *
143           STA  Ü1,X
144  ÜBYTES   EQU  *-ÜANFG
145           DS   -ÜBYTES
146           DO   ÜBYTES-2
147           HEX  3C          ; BIT ABS,X
148           ELSE
149           HEX  34          ; BIT ZPG,X
15Ø           FIN
151           DS   ÜBYTES-1
152           <<<
153  *
154  BRA      OP   CODE 8Ø
155  *
156  * Syntax: >>> BRA.ADDR
157  *          >>> BRA.LABEL
158  *
159  BRA      MAC
16Ø  *
161  * vorläufigen OP CODE für bedingten
162  * Sprung einfügen;
163  * Sprungdistanz prüfen und ggf.
164  * BAD BRANCH melden; Distanz setzen
165  *
166           BNE  Ü1
167  *
168  * Zeiger zurück auf vorl. OP CODE
169  *
17Ø           DS   -2
171  *
172  * OP CODE für BRA setzen
173  *
174           HEX  8Ø          ; BRA
```

```
175  *
176  * Zeiger vorsetzen, hinter Distanz
177  *
178           DS   1
179           <<<
18Ø  *
181  * CMP INDIRECT   OP CODE D2
182  *
183  * Syntax: >>> CMPI.ZP-ADDR o.
184  *          >>> CMPI.ZP-LABEL
185  *
186  CMPI     MAC
187           CMP  (Ü1,X)
188           DS   -2
189           HEX  D2          ; CMP IND
19Ø           DS   1
191           <<<
192  *
193  * DEA (DECREMENT ACCU )  OP CODE 3A
194  *
195  * Syntax: >>> DEA
196  *
197  DEA      MAC
198  *
199  * OP CODE für DEA setzen
2ØØ  *
2Ø1           HEX  3A          ; DEA
2Ø2           <<<
2Ø3  *
2Ø4  * EOR INDIRECT   OP CODE 52
2Ø5  *
2Ø6  * Syntax: >>> EORI.ZP-ADDR o.
2Ø7  *          >>> EORI.ZP-LABEL
2Ø8  *
2Ø9  EORI     MAC
21Ø           EOR  (Ü1,X)
211           DS   -2
212           HEX  52          ; EOR IND
213           DS   1
214           <<<
215  *
216  * INA ( INCREMENT ACCU ) OP CODE 1A
217  *
218  * Syntax: >>> INA
219  *
22Ø  INA      MAC
221           HEX  1A
222           <<<
223  *
224  * JMP INDIRECT,X  OP CODE 7C
225  *
226  * Syntax: >>> JMPIX.ADDR o.
227  *          >>> JMPIX.LABEL
228  *
229  * ADDR muß nicht auf ZPG liegen !
23Ø  *
231  JMPIX    MAC
232           HEX  7C          ; JMP IND,X
233           DA   Ü1
234           <<<
235  *
236  * LDA INDIRECT   OP CODE B2
237  *
238  * Syntax: >>> LDAI.ZP-ADDR o.
239  *          >>> LDAI.ZP-LABEL
24Ø  *
241  LDAI     MAC
242           LDA  (Ü1,X)
243           DS   -2
244           HEX  B2          ; LDA IND
245           DS   1
246           <<<
247  *
248  * ORA INDIRECT   OP CODE 12
249  *
25Ø  * Syntax: >>> ORAI.ZP-ADDR o.
251  *          >>> ORAI.ZP-LABEL
252  *
253  ORAI     MAC
254           ORA  (Ü1,X)
255           DS   -2
256           HEX  12          ; ORA IND
257           DS   1
258           <<<
259  *
26Ø  * PHX    OP CODE DA
261  *
```

```
262  * Syntax: >>> PHX
263  *
264  PHX      MAC
265           HEX  DA          ; PHX
266           <<<
267  *
268  * PHY    OP CODE 5A
269  *
27Ø  * Syntax: >>> PHY
271  *
272  PHY      MAC
273           HEX  5A          ; PHY
274           <<<
275  *
276  * PLX    OP CODE FA
277  *
278  * Syntax: >>> PLX
279  *
28Ø  PLX      MAC
281           HEX  FA          ; PLX
282           <<<
283  *
284  * PLY    OP CODE 7A
285  *
286  * Syntax: >>> PLY
287  *
288  PLY      MAC
289           HEX  7A          ; PLY
29Ø           <<<
291  *
292  * RMB [#(Ø-7)]      OP CODES Ø7-77
293  *
294  * Syntax: >>> RMB.Ø-7;ZP-ADDR
295  *          >>> RMB.Ø-7;ZP-LABEL
296  *
297  RMB      MAC
298           EOR  (Ü2,X)
299           DS   -1
3ØØ           DO   Ü1&$FFF8
3Ø1           JMP  $#Ø
3Ø2           ELSE
3Ø3  RMB1     EQU  Ü1*$1Ø.$ØF
3Ø4           DFB  RMB1          ; RMB
3Ø5           FIN
3Ø6           DS   1
3Ø7           <<<
3Ø8  *
3Ø9  * SBC INDIRECT   OP CODE F2
31Ø  * Syntax: >>> SBCI.ZP-ADDR o.
311  *          >>> SBCI.ZP-LABEL
312  *
313  SBCI     MAC
314           SBC  (Ü1,X)
315           DS   -2
316           HEX  F2          ; SBC IND
317           DS   1
318           <<<
319  *
32Ø  * SMB [#(Ø-7)]      OP CODES 87-F7
321  *
322  * Syntax: >>> SMB.Ø-7;ZP-ADDR
323  *          >>> SMB.Ø-7;ZP-LABEL
324  *
325  SMB      MAC
326           EOR  (Ü2,X)
327           DS   -1
328           DO   Ü1&$FFF8
329           JMP  $#Ø
33Ø           ELSE
331  SMB1     EQU  8+Ü1*$1Ø.$ØF
332           DFB  SMB1          ; SMB
333           FIN
334           DS   1
335           <<<
336  *
337  * STA INDIRECT   OP CODE 92
338  * Syntax: >>> STAI.ZP-ADDR o.
339  *          >>> STAI.ZP-LABEL
34Ø  *
341  STAI     MAC
342           STA  (Ü1,X)
343           DS   -2
344           HEX  92          ; STA IND
345           DS   1
346           <<<
347  *
348  * STZ  ABS / ZPG    OP CODES 64,9C
```

```
*
* Syntax: >>> STZ.ADDR o.
*          >>> STZ.LABEL
*
STZ       MAC
ÜANFG     EQU  *
          STA  Ü1
ÜBYTES    EQU  *-ÜANFG
          DS   -ÜBYTES
          DO   ÜBYTES-2
          HEX  9C          ; STZ ABS
          ELSE
          HEX  64          ; STZ ZPG
          FIN
          DS   ÜBYTES-1
          <<<
*
* STZ  ABS,X / ZPG,X  OP CODES 74, 9E
*
* Syntax: >>> STZX.ADDR o.
*          >>> STZX.LABEL
*
STZX      MAC
*
* aktuellen Zeigerstand merken
*
ÜANFG     EQU  *
*
* Speicherzugriff simulieren, um über
* vorläufigen OP CODE ein Byte zu re-
* servieren und die ADDR zu setzen
*
          STA  Ü1,X
*
* Länge der Instruktion berechnen
*
ÜBYTES    EQU  *-ÜANFG
*
* Zeiger auf Byte mit vorl. OP CODE
* zurück
*
          DS   -ÜBYTES
*
* Ist es ein Zweibytebefehl?
*
          DO   ÜBYTES-2
*
* nein, Dreibytebefehl; entspr.
*       OP CODE setzen
*
          HEX  9E          ; ABS,X
          ELSE
*
* ja, Zweibytebefehl; entspr.
*       OP CODE setzen
*
          HEX  74          ; ZPG,X
          FIN
*
* Zeiger vorsetzen, hinter ADDR
          DS   ÜBYTES-1
          <<<
*
* TRB  ABS / ZPG    OP CODES 14,1C
*
* Syntax: >>> TRB.ADDR o.
*          >>> TRB.LABEL
*
TRB       MAC
ÜANFG     EQU  *
          STA  Ü1
ÜBYTES    EQU  *-ÜANFG
          DS   -ÜBYTES
          DO   ÜBYTES-2
          HEX  1C          ; TRB ABS
          ELSE
          HEX  14          ; TRB ZPG
          FIN
          DS   ÜBYTES-1
          <<<
*
* TSB  ABS / ZPG    OP CODES Ø4,ØC
*
* Syntax: >>> TSB.ADDR o.
*          >>> TSB.LABEL
*
TSB       MAC
ÜANFG     EQU  *
          STA  Ü1
ÜBYTES    EQU  *-ÜANFG
          DS   -ÜBYTES
          DO   ÜBYTES-2
          HEX  ØC          ; TSB ABS
          ELSE
          HEX  Ø4          ; TSB ZPG
          FIN
          DS   ÜBYTES-1
          <<<
          FIN
```

Hinweis: Da viele eine Apple mit deutschem Zeichensatz benutzen, haben wir hier das Merlin-Macro-Symbol] = Ü nicht kodiert. Anm. d. Red.

SCREEN80

80-Z/Z-Bildschirm-Dump von U. Stiehl

Druckt (nach PR#3) den Bildschirminhalt der 8Ø-Zeichenkarte von IIc/IIe auf Image-Writer sowie auf Epson MX-8Ø und Epson FX-8Ø, falls letztere den Befehl Ctrl-I-N zum Abstellen des Bildschirmechos kennen. SCREEN8Ø belegt den Speicher $Ø3ØØ-$Ø3AD. Der Quellcode befindet sich auf der Peeker-Sammeldisk.

Dump aufrufen mit CALL 768
(ohne vorheriges PR#1)

Mögliche Modifikationen:
POKE 771, S (= Drucker-Slot 1-7; normal 1)
POKE 772, E (= Erste Zeile 1-22; normal 1)
POKE 773, L (= Letzte Zeile 2-24; normal 24)

```
1Ø DATA 76,1Ø,3,1,1,24,Ø,Ø,Ø,Ø
11 DATA 165,54,141,8,3,165,55,141,9,3
12 DATA 173,3,3,32,149,254,16Ø,Ø,14Ø,7
13 DATA 3,185,45,3,24Ø,17,32,171,3,172
14 DATA 7,3,2ØØ,2Ø8,239,  137,2Ø6,Ø,174,174:
   REM 137, 2Ø6, Ø = Ctrl-I + N + Endmarker
   REM 174 ... Füller für längeren Befehl
15 DATA 174,174,174,  173,4,3,56,233,1,141
16 DATA 6,3,32,193,251,16Ø,Ø,169,16Ø,141
17 DATA 85,192,2Ø9,4Ø,2Ø8,14,141,84,192,2Ø9
18 DATA 4Ø,2Ø8,7,2ØØ,192,4Ø,2Ø8,237,24Ø,42
19 DATA 16Ø,Ø,14Ø,7,3,173,6,3,32,193
2Ø DATA 251,172,7,3,141,85,192,32,154,3
21 DATA 172,7,3,141,84,192,32,154,3,172
22 DATA 7,3,2ØØ,192,4Ø,2Ø8,221,169,141,32
23 DATA 164,3,238,6,3,173,6,3,2Ø5,5
24 DATA 3,144,175,173,8,3,133,54,173,9
25 DATA 3,133,55,96,177,4Ø,9,128,2Ø1,16Ø
26 DATA 176,2,1Ø5,64,141,84,192,32,171,3
27 DATA 96,1Ø8,54,Ø
28 RESTORE
29 FOR X = 768 TO 941: READ Y: POKE X,Y: NEXT
```

Terminal-Programm für den Apple II

von Andreas Lecreux

Inzwischen gibt es in fast jeder Großstadt mindestens eine **Mailboxstation**. Um auch mit dem Apple Zugang zu diesen Diensten zu erhalten, benötigt man nicht nur einen Akustikkoppler oder ein Modem, sondern auch ein entsprechendes Programm. Ein solches Programm will ich Ihnen hier vorstellen. Als Voraussetzung benötigt man einen Apple II mit dem Betriebssystem DOS und eine serielle Schnittstelle in Slot 2. Bei Verwendung eines anderen Slots als Slot 2 oder eines anderen Bausteines als des 6850 müssen die entsprechenden Werte in dem Programm modifiziert werden. Das Programm kennt drei Betriebsarten.

1. Dialog-Modus

In dieser Betriebsart ist der Apple ein asynchron arbeitendes Terminal. Jeder Tastendruck wird an die Gegenstelle gesendet. Ausnahme: **Ctrl-B** zeigt die aktuelle Empfangsspeicheradresse an und **ESC** beendet den Modus durch Rücksprung in das Applesoft-Programm. Jedes empfangene Zeichen wird auf dem Bildschirm dargestellt und im internen Speicher, dem Empfangsspeicher, abgelegt. Es findet eine automatische LF-Unterdrükkung (Zeilenvorschub Ctrl-J) statt, um unnötige Leerzeilen auf dem Bildschirm zu vermeiden. Das erste LF nach einem CR (neue Zeile Ctrl-M) wird ignoriert.
Der Empfangsspeicher kann als Binärfile abgespeichert und später weiterverarbeitet werden. Man kann sich also nachträglich den ganzen Dialog ansehen und interessante Stellen oder Menüs ausdrucken, um so beim nächsten Mal Zeit zu sparen. Es ist also nicht unbedingt nötig, darauf zu achten, ob man **Download** aktiviert hat. Wenn der Empfangsspeicher voll ist, beendet das Programm selbsttätig den Dialog-Modus und gibt einen entsprechenden Hinweis auf dem Bildschirm. Bei *jedem* Verlassen des Dialogs sendet das Programm **XOFF** (Ctrl-S), um ein Timeout in der Gegenstelle zu verlängern. Viele Mailboxen nehmen darauf Rücksicht. Wird der Dialog-Modus aktiviert, wird **XON** (Ctrl-Q) gesendet, um der Gegenstelle die erneute Bereitschaft zu signalisieren.

2. Upload-Modus

In dieser Betriebsart werden ASCII-Dateien gesendet. Wird keine Textdatei auf der Diskette gefunden, so versucht das Programm eine Binärdatei in den Speicher zu laden. Da in diesem Fall der Empfangsspeicher überschrieben wird, sollte er vorher gespeichert werden. Aus Sicherheitsgründen wird dies von dem Programm noch einmal abgefragt. Dann wird der Ausgabevektor auf das Terminalprogramm gesetzt und die Datei zeichenweise gesendet. Ein Abbruch ist mittels **ESC** jederzeit möglich. In der Betriebsart Vollduplex findet eine automatische Korrektur statt, indem das Empfangsecho mit dem gesendeten Zeichen verglichen wird. Bei Nicht-Übereinstimmung wird Ctrl-H (BS) gesendet und die Übertragung des Zeichens wiederholt.

3. Download-Modus

In dieser Betriebsart können Dateien empfangen werden. Jedes empfangene Zeichen wird im Empfangspuffer abgelegt. Auch hier findet eine automatische LF-Unterdrückung wie beim Dialog-Modus statt.
Im Gegensatz zum Dialog-Modus ist hier die Tastatur bis auf die Tasten **ESC** und

Ctrl-B gesperrt. Außerdem wird in der Betriebsart Vollduplex jedes empfangene Zeichen geechot. Damit wird der Gegenstelle die Möglichkeit gegeben, die Übertragung auf Korrektheit zu kontrollieren. Nach Beendigung des Modus wird die aktuelle Adresse des letzten abgespeicherten Zeichens angezeigt.

In den Betriebsarten 2 und 3 beherrscht das Programm das **XON/XOFF-Protokoll**. Weitere Steuerzeichen, die während der Übertragung benutzt werden, sind:

Ctrl-B – STX Es beginnt eine neue Datei, lösche den Empfangsspeicher
Ctrl-C – ETX Die Datei ist zu Ende
Ctrl-H – BS Lösche letztes Zeichen
Ctrl-Q – XON Fahre mit dem Senden fort
Ctrl-S – XOFF Beende kurzzeitig die Übertragung
Ctrl-X – CAN Gegenstelle wünscht Abbruch der Übertragung.

Nach Beendigung einer Übertragung wird ein Fehlercode auf dem Bildschirm angezeigt. Dieser setzt sich folgendermaßen zusammen. Code: „BF" steht für Betriebsart und Fehlercode.

Betriebsart:
0 – Empfangen
2 – Download
4 – Warten auf XON
8 – Senden

Fehlercode:
0 – Kein Fehler
1 – ESC-Operator-Eingriff
2 – Ctrl-X empfangen (CAN)
3 – Timeout erreicht

Bei den Funktionen UPLOAD und DOWNLOAD ist zu beachten, daß die Gegenstelle den gleichen Dialog-Modus (das gleiche Übertragungsprotokoll) erfüllt. Nicht alle Mailboxsysteme senden Kontrollzeichen als Echo zurück. Die Fehlerabfrage ist in diesem Fall aus dem Programm herauszunehmen. Zur Zeit gibt es leider noch keine bindende Vereinbarung. Im Zweifelsfall muß in Halbduplex ohne automatische Korrektur übertragen werden.

Alle zeitkritischen Programmteile sind in Assembler geschrieben. Das Menü wurde der Einfachheit halber in Applesoft realisiert. So kann das Hauptprogramm problemlos an individuelle Wünsche angepaßt werden. Das Applesoft-Programm gliedert sich in 4 Teile:

1000-1500 Hauptprogramm
1500-2000 Initialisierung
2000-8000 Unterprogramme (lt. Menü)
9000-9999 Sonstige Unterprogramme.
Interessant dürfte das Line-Input-Unterprogramm (ab 9080) sein. Es wird nicht der String mittels GET-Befehlen zusammengesetzt, sondern die Monitorroutine GETLN aufgerufen und der String durch PEEK aus dem Eingabespeicher ab $0200 ausgelesen. Diese Methode dauert zwar etwas länger, arbeitet aber auch beim Lesen von Textdateien einwandfrei. Die Pause während des Auslesens stört auch nicht weiter, da die Gegenstelle nach einem CR meistens etwas Zeit benötigt, um den empfangenen Satz abzuspeichern. Das Programm gibt folgende Befehle an das serielle Schnittstellen-Treiberprogramm, eingeleitet durch Ctrl-I:

TD – schalte RTS an
TS – schalte RTS ab
Tx – selektiere Übertragungsgeschwindigkeit
U99N – arbeite ohne automatisches CR
C – schalte Echo auf Bildschirm ein (wenn vorhanden, nur bei Simplex benötigt)
O – schalte Echo auf Bildschirm aus

Der Empfangsspeicher beginnt bei $2500 und kann maximal bis zur DOS-Grenze $9600 anwachsen. Ab $9500 erscheint aber schon die Meldung: „PUFFER ZU GROSS". Der Speicher muß nun gesichert oder gelöscht werden. Wenn das Applesoft-Programm erweitert werden soll, muß man unter Umständen das Assembler-Programm verschieben, um neuen Platz zu erhalten. Das Assembler-Programm grenzt unmittelbar an das Applesoft-Programm. Es beginnt mit einer **Sprungtabelle** und einer **Parametertabelle**. Die Hauptteile des Programms können also geändert werden, *ohne* daß das Applesoft-Programm an neue Adressen angepaßt werden muß. Das Assembler-Programm besteht aus den Teilen Dialog, Upload, Download, sende Byte, empfange Byte. Am Ende des Programms befindet sich eine Tabelle, die alle Steuercodes enthält. Durch Austausch dieser Tabelle können auch andere Steuercodes ausgewertet werden, ohne daß das Programm neu assembliert werden muß. Das Assembler-Programm wurde auf größtmögliche Universalität ausgelegt. Es ist auch ohne weiteres möglich, mit den gleichen Routinen und einem anderen Applesoft-Programm ein Mailboxsystem zu schreiben. Viel Spaß bei der Datenübertragung.

TERMINAL

```
1000 HIMEM: 8192: GOTO 1500
1010 CALL DI: PRINT: PRINT
1020 PRINT "1 - Tempo              2 - Bildschirm"
1021 PRINT "3 - sichere Daten      4 - lösche Schirm"
1022 PRINT "5 - lösche Daten       6 - sende Text"
1023 PRINT "7 - empfange Daten     8 - Test sendung"
1024 PRINT "9 - Sender Echo      ESC - Eingabe Text"
1025 PRINT "A - Voll/Halb duplex   B - Catalog Disk"
1027 PRINT "E - Programm Ende      F - Neu Initial."
1030 PRINT
1060 GET FU$:FU = VAL (FU$)
1070 PRINT : PRINT D$"PR#"SE: PRINT : PRINT D$"PR#"SC
1080 ON FU GOSUB 2100,2200,2300,2400,2500,2600,3600,
     2800,2700
1090 IF ASC (FU$) = 27 THEN 3300
1100 IF FU$ = "E" THEN END
1105 IF ASC (FU$) > 96 THEN FU$ = CHR$ ( ASC (FU$) - 32):
     REM Klein -> Groß
1110 IF ASC (FU$) < 65 OR ASC (FU$) > 96 THEN 1010
1120 ON ASC (FU$) - 64 GOSUB 3000,3100,1190,1190,1190,
     2900
1130 GOTO 1010
1190 RETURN : REM Pseudo GOSUB
1500 H$ = CHR$ (26) + "0"
1510 I$ = CHR$ (13) + CHR$ (9)
1520 SC = 0
1530 SE = 2
1540 D$ = CHR$ (13) + CHR$ (4): REM DOS
1545 TI$ = "   TERMINAL SIMULATOR AKTIV V:1.3"
1550 PRINT H$: HOME
1560 INVERSE : PRINT "TERMINAL SIMULATOR": NORMAL : PRINT
1570 J = 1
1580 FOR I = J TO 6
1590 PRINT : PRINT
1600 ON I GOSUB 2900,2100,2500,2700,3000,2200
1610 NEXT
1620 PRINT D$"PR#"SC
1650 GOTO 1010
2100 REM Speed
2105 PRINT "SPEED ZUR ZEIT = "SP
2110 PRINT "SPEED ( 1-19200 2-1200 3-300) ";: GET SP$
2115 PRINT SP$
2120 SP = VAL (SP$): IF SP < 1 OR SP > 3 THEN 2110
2125 PRINT D$"PR#"SE: PRINT I$"T"SP - 1: PRINT D$"PR#"SC
2130 RETURN
2200 REM Screen Slot
2205 PRINT "SCREEN ZUR ZEIT = "SC
2210 PRINT "SCREEN = SLOT 0 ODER 3 ";: GET SC$:
     SC = VAL (SC$)
2215 PRINT SC$
2220 IF SC < > 0 AND SC < > 3 THEN 2210
2225 IF SC = 0 THEN PRINT : PRINT CHR$ (26)"1": PRINT
2240 PRINT D$"PR#"SC: PRINT D$"IN#"SC
2245 IF SC = 0 THEN POKE 0H,253: POKE 0L,240: GOTO 2260:
     REM JSR $FDF0
2250 POKE 0L,07
2255 POKE 0H,12 * 16 + SC
2260 GOTO 2400
2300 REM Puffer schreiben
2305 GOSUB 9000
2310 A$ = "$2500"
2315 L = 1 + PEEK (BH) * 256 + PEEK (BL) -
     (2 * 4096 + 5 * 256)
2320 L$ = STR$ (L)
2325 GOSUB 9066
2330 GOTO 2515
2400 REM Bildschirm löschen
2405 PRINT H$: HOME
2410 PRINT TI$
2415 PRINT "-------------------------------------"
2420 RETURN
2500 REM Puffer löschen
2505 PRINT "SICHER OB PUFFER GELÖSCHT WIRD?";:
     GET FU$: PRINT FU$
2510 IF FU$ = "N" OR FU$ = "n" THEN RETURN
2515 POKE BL,0: POKE BH,37
2520 INVERSE : PRINT "PUFFER AUF 0": NORMAL
2525 RETURN
2600 REM sende Datei
2601 POP : REM wegen ON ERR GOTO
2602 IF DO = 2 THEN PRINT D$;"PR#";SE: PRINT I$;"TD":
     PRINT D$;"PR#";SC
2604 GOSUB 9000: REM Input Name
2606 N1$ = N$:N$ = "TEST": GOSUB 9040: GOSUB 9072
2607 N$ = N1$
2608 ONERR GOTO 2650
2610 GOSUB 9060: REM Datei lesen
2612 ONERR GOTO 2680
2614 CALL UP(DO)
2616 PRINT CHR$ (13);: IF PEEK (ER) = 0 THEN
     PRINT CHR$ (2): REM STX
2618 IF PEEK (ER) < > 0 GOTO 2680:
     REM STX nicht angenommen
2620 GOSUB 9080: REM  LINE INPUT
2622 FOR I = 1 TO LEN (LI$): PRINT MID$ (LI$,I,1);
2624 IF PEEK (ER) = 0 THEN NEXT : GOTO 2620
2630 GOTO 2680: REM Datei Ende
2650 PRINT "Kein Textfile gefunden! Empfangspuffer wird
     überschrieben. Vorher sichern? ";
2652 GET FU$: IF FU$ = "Y" OR FU$ = "y" OR FU$ = "J" OR
     FU$ = "j" THEN PRINT FU$:N1$ = N$:S1 = S:D1 = D:
     GOSUB 2300:N$ = N1$:D = D1:S = S1: GOTO 2656
2654 IF FU$ < > "N" AND FU$ < > "n" THEN GOTO 2652
2656 PRINT : ONERR GOTO 2660
2658 A$ = "$2500": GOSUB 9069: GOTO 2662: REM Bin LOAD
2660 PRINT "DATEI FEHLER CODE:"; PEEK (222): GOTO 2690
2662 L = PEEK (43617) * 256 + PEEK (43616) + 9472:
     REM Datei-Länge
2664 CALL UP(DO)
2666 PRINT CHR$ (13);: IF PEEK (ER) = 0 THEN
     PRINT CHR$ (2): REM STX
2668 FOR I = 9472 TO L: PRINT CHR$ ( PEEK (I));:
     IF PEEK (ER) = 0 THEN NEXT
2680 E = PEEK (ER)
2682 PRINT CHR$ (13);: IF PEEK (ER) = 0 THEN
     PRINT CHR$ (3): REM ETX
2684 POKE ER,E
2686 CALL UE
2690 POKE 216,0:D$ = CHR$ (13) + CHR$ (4): GOSUB 9046
2692 IF DO = 2 THEN GOSUB 3045
2694 GOTO 1010: END
2700 PRINT "ECHO ZUR ZEIT ";
2705 IF MC = 2 THEN PRINT "AKTIV"
2710 IF MC = 1 THEN PRINT "AUS"
2715 PRINT "AKTIVIEREN = J ";: GET FU$
2720 PRINT FU$
2725 IF FU$ = "N" OR FU$ = "n" THEN MC = 1: GOTO 2735
2730 MC = 2
2735 PRINT D$"PR#"SE: IF MC = 1 THEN PRINT I$"0":
     POKE EC,0: REM COUT off
2740 IF MC = 2 THEN PRINT I$"C": POKE EC,128: REM COUT on
2745 PRINT D$"PR#"SC
2750 RETURN
2800 REM Testsendung
2805 IF DO = 2 THEN PRINT D$;"PR#";SE: PRINT I$;"TD"
2815 CALL UP(DO)
2820 PRINT : IF PEEK (ER) < > 0 THEN GOTO 2840
2825 FOR I = 32 TO 127: PRINT CHR$ (I);
2830 IF PEEK (ER) < > 0 THEN GOTO 2840
2835 NEXT : PRINT
2840 CALL UE
2845 IF DO = 2 THEN GOSUB 3045
2847 RETURN
2900 REM INIT
2902 IF DI = 0 THEN DIM UP(2),DO(2)
2904 PRINT D$"PR#"SE: PRINT I$"U99N": PRINT D$"PR#"SC
2906 DI = 2 * 4096: REM Dialog Entry
2908 IF UP(1) = 0 AND PEEK (DI) = 76 AND
     PEEK (DI + 3) = 76 AND PEEK (DI + 6) = 76 GOTO 2912
2910 PRINT D$"BLOAD TERMINAL.B,A$2000"
2912 UP(1) = DI + 3: REM UPLOAD Duplex
2913 DO(1) = DI + 12: REM DOWNLOAD
2914 UP(2) = DI + 6: REM UPLOAD Simplex
2915 DO(2) = DI + 15: REM DOWNLOAD
2916 UE = DI + 9: REM UPLOADend
2920 TX = DI + 18: REM TRANSMIT
2922 RX = DI + 21: REM RECEIVE
2924 BL = PEEK (DI + 25) * 256 + PEEK (DI + 24)
2926 BH = BL + 1
2928 0L = PEEK (DI + 27) * 256 + PEEK (DI + 26)
2930 0H = 0L + 1
2932 TB = DI + 28: REM SENDBYTE
2934 RB = DI + 29: REM EMPFBYTE
2936 ER = DI + 30: REM ERRORCODE
2937 EC = DI + 31: REM ECHO CODE
2938 RETURN
3000 REM Betriebsart voll/halb DOUPLEX
3005 PRINT
3010 PRINT "BETRIEBSART = ";
3015 IF DO = 1 THEN PRINT "VOLL";
3020 IF DO = 2 THEN PRINT "HALB";
3025 PRINT " DOUPLEX"
```

```
3030 PRINT "VOLL = 1, HALB = 2 ";
3035 GET FU$:DO = VAL (FU$): IF DO < 1 OR DO > 2
     THEN 3035
3040 PRINT DO
3045 PRINT D$"PR#"SE
3050 IF DO = 1 THEN PRINT I$"TD"
3055 IF DO = 2 THEN PRINT I$"TS"
3060 PRINT D$"PR#"SC
3065 RETURN
3100 REM CATALOG Disk
3105 GOSUB 9004
3110 PRINT H$: HOME
3115 PRINT D$"CATALOG,S"S",D"D
3120 RETURN
3300 REM sende String
3310 PRINT "TEXT:";: GOSUB 9080
3320 IF LI$ = "" GOTO 1010
3330 IF DO = 2 THEN PRINT D$;"PR#";SE: PRINT I$;"TD"
3340 CALL UP(DO)
3350 FOR I = 1 TO LEN (LI$)
3360 PRINT MID$ (LI$,I,1);
3370 IF PEEK (ER) < > 0 THEN GOTO 3390
3380 NEXT : PRINT
3390 CALL UE
3410 IF DO = 2 THEN GOSUB 3045
3420 GOTO 1010
3600 REM DOWNLOAD
3610 CALL DO(DO)
3620 IF PEEK (ER) = 0 AND PEEK (TB) = 147 THEN
     GOSUB 2300: GOTO 3610
3690 RETURN
9000 REM Eingabe-Standart
9002 INPUT "FILENAME ";N$
9004 PRINT "SLOT ";: GET S$:S = VAL (S$)
9006 IF (S < 1) OR (S > 7) THEN 9004
9008 PRINT S" DRIVE ";: GET DR$
9010 D = VAL (DR$): IF (D < 1) OR (D > 2) THEN
     GET DR$: GOTO 9010
9012 PRINT D: RETURN
9014 REM
9016 REM Übergabe-Konstanten
9018 REM
9020 REM N$ - Dateiname
9022 REM N1$; N2$; Namen bei Renamefunktion
9024 REM A$; L$; Anfang und Länge in HEX bei
     BIN-Funktionen
9026 REM S  - Slot
9028 REM D  - Drive
9030 REM V  - Volume
9032 REM R  - Record
9034 REM B  - Byte
9040 REM OPEN S
9042 PRINT CHR$ (4);"OPEN";N$;",S";S;",D";D;",V";V
9044 RETURN
9046 REM CLOSE S
9048 PRINT
9050 PRINT CHR$ (4);"CLOSE";N$
9052 RETURN
9054 REM WRITE S
9056 PRINT CHR$ (4);"WRITE";N$
9058 RETURN
9060 REM READ S
9062 PRINT CHR$ (4);"READ";N$
9064 RETURN
9066 REM Bin SAVE
9067 PRINT CHR$ (4);"BSAVE";N$;",A";A$;",L";L$;",S";
     S;",D";D
9068 RETURN
9069 REM Bin LOAD
9070 PRINT CHR$ (4);"BLOAD";N$;",A";A$;",S";S;",D";D
9071 RETURN
9072 REM DELETE
9074 PRINT CHR$ (4);"DELETE";N$;",S";S;",D";D
9076 RETURN
9080 REM LINE INPUT LI$
9082 REM LI$ - Übergabe String
9084 REM LI  - Zählervariable
9086 POKE 51,000
9088 CALL 64874
9090 LI$ = "":LI = 511
9092 LI = LI + 1: IF PEEK (LI) < > 141 THEN LI$ = LI$ +
     CHR$ ( PEEK (LI)): GOTO 9092
9094 LI$ = LI$ + CHR$ (13): RETURN
```

TERMINAL.B

```
                        1   ****************************************
                        2   *                                      *
                        3   *  TERMINAL.B von A. Lecreux 1985      *
                        4   *                                      *
                        5   ****************************************
                        6
                        7   DATEN    EQU  $C085+$20  ;Slot 2 seriell
                        8   STATUS   EQU  DATEN-1    ;Status seriell
                        9   KBD      EQU  $C000      ;Tastatur
                       10   KBDSTB   EQU  KBD+$10    ;Strobe
                       11   SEROUT   EQU  $C207      ;Output Slot 2
                       12   SERIN    EQU  $C205      ;Input Slot 2
                       13   COUT     EQU  $FDED      ;Output 40 Zei.
                       14   PRBYTE   EQU  $FDDA
                       15   DOSIO    EQU  $3EA
                       16   MEMH     EQU  $25        ;Memory H.Byte
                       17   MEML     EQU  $0         ;Memory L.Byte
                       18   TDI      EQU  $1         ;TIMEOUT Dialog
                       19   TUPL     EQU  $5         ;Time UPLOAD Echo
                       20   TOUT     EQU  $20        ;TIMEOUT normal
                       22
                       23            ORG  $2000
                       24
2000: 4C 24 20         25            JMP  DIMODE     ;Dialog
2003: 4C 4A 21         26            JMP  UPDOUP     ;UPLOAD Duplex
2006: 4C 3A 21         27            JMP  UPLOAD     ;UPLOAD Simplex
2009: 4C 95 21         28            JMP  ENDUP      ;Ende UPLOAD
200C: 4C D7 21         29            JMP  DOWNLOAD   ;DOWNLOAD Duplex
200F: 4C EC 21         30            JMP  DOWNS      ;DOWNLOAD Simplex
2012: 4C D6 22         31            JMP  TXMODE     ;sende Byte
2015: 4C 8F 23         32            JMP  RXMODE     ;empfange Byte
                       33
2018: 6A 20            34            DA   BUFFER     ;Puffer-Adresse
201A: 21 20            35            DA   OUT1       ;Output Screen
                       36
201C: 00               37   TXBYTE   DFB  0          ;Puffer
201D: 00               38   RXBYTE   DFB  0
                       39
201E: 00               40   ERRBYTE  DFB  0          ;1-ESC
                       41   *                        ;2-CAN
                       42   *                        ;3-TIMEOUT
                       43
201F: 00               44   ECHOFLAG DFB  0          ;Bildschirm Echo
                       45
                       46
2020: 4C F0 FD         47   COUT0    JMP  COUT+3
                       48   OUT1     EQU  *-2
                       49   OUT2     EQU  *-1
                       50
2023: 00               51   STATE    DFB  0          ;Progr.-Status
                       52
                       53   * B=Basic, L=DOWNLOAD, U=UPLOAD,
                       54   * D=DIALOG, S=Puffer sichern.
                       55
                       56   ****************************************
                       57   *
                       58   *        DIALOG-MODUS
                       59   *
                       60   ****************************************
                       61
2024: A9 44            62   DIMODE   LDA  #'D'       ;DIALOG
2026: 8D 23 20         63            STA  STATE
                       64
2029: AD 07 24         65   ENTRY    LDA  GOON       ;sende Ctrl-Q
202C: 20 D6 22         66            JSR  TXMODE
                       67
202F: 20 78 20         68   START    JSR  RECEIVE
2032: AD 00 C0         69            LDA  KBD
2035: 10 F8            70            BPL  START      ;keine Taste
2037: 2C 10 C0         71            BIT  KBDSTB
203A: C9 9B            72            CMP  #$9B       ;ESC?
203C: F0 20            73            BEQ  BASIC
203E: C9 82            74            CMP  #$82       ;Ctrl-B?
2040: F0 16            75            BEQ  BUFF0      ;Puffer anzeigen
2042: A2 00            76            LDX  #0
2044: 8E 1D 20         77            STX  RXBYTE
2047: 20 D6 22         78            JSR  TXMODE     ;sende Taste
204A: AD 1D 20         79            LDA  RXBYTE     ;etwas empfangen
204D: F0 E0            80            BEQ  START      ;nein
204F: 20 69 20         81            JSR  STORE      ;speichern
2052: 20 A9 20         82            JSR  RC0A       ;Puffer-Test
2055: 4C 2F 20         83            JMP  START
                       84
2058: 20 02 21         85   BUFF0    JSR  BUFF
205B: 4C 2F 20         86            JMP  START
                       87
```

```
205E: A9 42     88   BASIC    LDA  #'B'       ;Basic Ende
2060: 8D 23 20  89            STA  STATE
2063: AD 09 24  90            LDA  WAIT
2066: 4C D6 22  91            JMP  TXMODE
                92
2069: 8D        93   STORE    DFB  $8D        ;STA $2500
206A: 00 25     94   BUFFER   DFB  MEML,MEMH
206C: EE 6A 20  95            INC  BUFFER
206F: D0 03     96            BNE  TESTBUF
2071: EE 6B 20  97            INC  BUFFER+1
2074: AD 6B 20  98   TESTBUF  LDA  BUFFER+1
2077: 60        99            RTS
                100
2078: AD A4 C0  101  RECEIVE  LDA  STATUS
207B: 2C EF 23  102           BIT  RXREADY
207E: F0 31     103           BEQ  RETURN     ;nichts empfangen
                104  *        JSR  SERIN      ;lies Byte
                105  *        JSR  RXMODE
2080: AD A5 C0  106           LDA  DATEN
2083: 29 7F     107           AND  #$7F
2085: CD 00 24  108           CMP  LF
2088: D0 12     109           BNE  RC09       ;kein LF
208A: AD F5 23  110           LDA  LBYTE      ;davor CR
208D: CD 03 24  111           CMP  CR
2090: D0 07     112           BNE  RC08       ;nein
2092: AD 00 24  113           LDA  LF
2095: 8D F5 23  114           STA  LBYTE
2098: 60        115           RTS
                116
2099: AD 00 24  117  RC08     LDA  LF
209C: 8D F5 23  118  RC09     STA  LBYTE
209F: 09 80     119           ORA  #$80
20A1: 48        120           PHA
20A2: 20 ED FD  121           JSR  COUT
20A5: 68        122           PLA
20A6: 20 69 20  123           JSR  STORE
20A9: C9 95     124  RC0A     CMP  #$95
20AB: F0 05     125           BEQ  BUFFH      ;Puffer zu groß
20AD: C9 96     126           CMP  #$96
20AF: F0 20     127           BEQ  LOOP2A
20B1: 60        128  RETURN   RTS
                129
20B2: 20 02 21  130  BUFFH    JSR  BUFF
20B5: A9 00     131           LDA  #0
20B7: 8D 1D 20  132           STA  RXBYTE
20BA: AD 09 24  133           LDA  WAIT       ;sende Ctrl-S
20BD: 20 D6 22  134           JSR  TXMODE
20C0: B0 0F     135           BCS  LOOP2-2
20C2: AD 1D 20  136           LDA  RXBYTE
20C5: D0 A2     137           BNE  STORE      ;noch etwas empfangen
20C7: A9 01     138           LDA  #TDI
20C9: 8D F3 23  139           STA  TIME2
20CC: 20 8F 23  140           JSR  RXMODE     ;empfange
20CF: 90 98     141           BCC  STORE      ;etwas empfangen
                142
20D1: A2 00     143  LOOP2A   LDX  #0         ;Ende DIALOG
20D3: BD EA 20  144  LOOP2    LDA  MSG2,X
20D6: C9 00     145           CMP  #0
20D8: F0 08     146           BEQ  LOOP2E
20DA: 09 80     147           ORA  #$80
20DC: 20 ED FD  148           JSR  COUT
20DF: E8        149           INX
20E0: D0 F1     150           BNE  LOOP2
20E2: 68        151  LOOP2E   PLA
20E3: 68        152           PLA
20E4: A9 53     153           LDA  #'S'       ;sichern
20E6: 8D 23 20  154           STA  STATE
20E9: 60        155           RTS             ;nach Basic
                156
20EA: 8D 87     157  MSG2     DFB  $8D,$87
20EC: 2A 20 42  158           ASC  '* PUFFER ZU GROSS *'
20EF: 55 46 46 45 52 20 5A 55
20F7: 20 47 52 4F 53 53 20 2A
20FF: 87 8D 00  159           DFB  $87,$8D,0
                160
2102: A2 00     161  BUFF     LDX  #0         ;Puffer Ausdruck
2104: BD 25 21  162  LOOP1    LDA  MSG1,X
2107: C9 00     163           CMP  #0
2109: F0 09     164           BEQ  LOOP1E
210B: 09 80     165           ORA  #$80
210D: 20 ED FD  166           JSR  COUT
2110: E8        167           INX
2111: 4C 04 21  168           JMP  LOOP1
2114: AD 6B 20  169  LOOP1E   LDA  BUFFER+1
2117: 20 DA FD  170           JSR  PRBYTE
211A: AD 6A 20  171           LDA  BUFFER
211D: 20 DA FD  172           JSR  PRBYTE
```

```
2120: A9 8D     173           LDA  #$8D
2122: 4C ED FD  174           JMP  COUT       ;incl. RTS
                175
2125: 8D        176  MSG1     DFB  $8D
2126: 2A 20 42  177           ASC  '* PUFFER ZUR ZEIT: '
2129: 55 46 46 45 52 20 5A 55
2131: 52 20 5A 45 49 54 3A 20
2139: 00        178           DFB  0
                179
                180
                181  ****************************************
                182  *
                183  *        UPLOAD-MODUS
                184  *
                185  ****************************************
                186
213A: A9 D6     187  UPLOAD   LDA  #TXMODE    ;neuer Vektor
213C: 85 36     188           STA  $36
213E: A9 22     189           LDA  #TXMODE/256
2140: 85 37     190           STA  $37
2142: A9 55     191  UPL0     LDA  #'U'       ;UPLOAD
2144: 8D 23 20  192           STA  STATE
2147: 4C EA 03  193           JMP  DOSIO
                194
214A: A9 55     195  UPDOUP   LDA  #UPDOUP1   ;neuer Vektor
214C: 85 36     196           STA  $36
214E: A9 21     197           LDA  #UPDOUP1/256
2150: 85 37     198           STA  $37
2152: 4C 42 21  199           JMP  UPL0
                200
2155: 20 D6 22  201  UPDOUP1  JSR  TXMODE
2158: B0 37     202           BCS  BACK
215A: A9 05     203           LDA  #TUPL      ;TIMEOUT
215C: 8D F3 23  204           STA  TIME2
215F: 20 8F 23  205           JSR  RXMODE     ;Echo empf.
2162: B0 27     206           BCS  BACK2
2164: AD 1C 20  207           LDA  TXBYTE
2167: CD 1D 20  208           CMP  RXBYTE     ;korrekt?
216A: F0 22     209           BEQ  BACK3      ;richtig
216C: A9 01     210           LDA  #TDI
216E: 8D F3 23  211           STA  TIME2
2171: 20 8F 23  212           JSR  RXMODE     ;wegen SYN 2*
2174: AD 1C 20  213           LDA  TXBYTE
2177: CD 1D 20  214           CMP  RXBYTE
217A: F0 12     215           BEQ  BACK3
217C: 48        216           PHA
217D: AD FE 23  217           LDA  BS         ;Korrektur
                218  *        JSR  UPDOUP1    ;senden
2180: 20 D6 22  219           JSR  TXMODE
2183: B0 04     220           BCS  BACK0
2185: 68        221           PLA
2186: 4C 55 21  222           JMP  UPDOUP1    ;Retry
                223
2189: 68        224  BACK0    PLA
218A: 60        225           RTS
                226
218B: AD 1E 20  227  BACK2    LDA  ERRBYTE
218E: 20 20 20  228  BACK3    JSR  COUT0
2191: AD 1C 20  229  BACK     LDA  TXBYTE
2194: 60        230           RTS
                231
2195: AD 21 20  232  ENDUP    LDA  OUT1       ;alten Vektor
2198: 85 36     233           STA  $36
219A: AD 22 20  234           LDA  OUT2
219D: 85 37     235           STA  $37
219F: 20 EA 03  236           JSR  DOSIO
21A2: A2 00     237           LDX  #0
21A4: BD C1 21  238  ENDUP1   LDA  ENDUP3,X   ;Text ausgeben
21A7: F0 08     239           BEQ  ENDUP2
21A9: 09 80     240           ORA  #$80
21AB: 20 ED FD  241           JSR  COUT
21AE: E8        242           INX
21AF: D0 F3     243           BNE  ENDUP1
21B1: AD 1E 20  244  ENDUP2   LDA  ERRBYTE
21B4: 20 DA FD  245           JSR  PRBYTE
21B7: A9 42     246           LDA  #'B'       ;Basic
21B9: 8D 23 20  247           STA  STATE
21BC: A9 8D     248           LDA  #$8D
21BE: 4C ED FD  249           JMP  COUT
                250
21C1: 0D        251  ENDUP3   DFB  13
21C2: 55 50 4C  252           ASC  'UPLOAD BEENDET CODE:'
21C5: 4F 41 44 20 42 45 45 4E
21CD: 44 45 54 20 43 4F 44 45
21D5: 3A
21D6: 00        253           DFB  0
                254
                255
```

```
                    256  ****************************************
                    257  *
                    258  *         DOWNLOAD-MODUS
                    259  *
                    260  ****************************************
                    261
21D7: 20 2E 22      262  DOWNLOAD JSR  DW0         ;starte
21DA: B0 25         263           BCS  DOWNS2
21DC: 20 3E 22      264  DOWN1    JSR  DW1         ;empf. & speich.
21DF: B0 20         265           BCS  DOWNS2
21E1: AD 1D 20      266           LDA  RXBYTE
21E4: 48            267           PHA
21E5: 20 ED FD      268           JSR  COUT
21E8: 68            269           PLA
21E9: 4C DC 21      270           JMP  DOWN1       ;sende Echo
                    271
21EC: 20 2E 22      272  DOWNS    JSR  DW0         ;starte
21EF: B0 10         273           BCS  DOWNS2
21F1: 20 3E 22      274           JSR  DW1         ;empf. & speich.
21F4: B0 0B         275           BCS  DOWNS2
21F6: AD 1D 20      276  DOWNS1   LDA  RXBYTE     ;Schleife Start
21F9: 20 ED FD      277           JSR  COUT
21FC: 20 43 22      278           JSR  DW2
21FF: 90 F5         279           BCC  DOWNS1
2201: 4C 9C 22      280  DOWNS2   JMP  ENDW
                    281
2204: A9 22         282  DWCAN    LDA  #$22
2206: 8D 1E 20      283           STA  ERRBYTE
2209: 38            284           SEC
220A: 60            285           RTS
                    286
220B: CE 6A 20      287  DWBS     DEC  BUFFER
220E: 10 11         288           BPL  DWBS0       ;Puffer - 1
2210: AD 6A 20      289           LDA  BUFFER
2213: C9 FF         290           CMP  #$FF        ;Überlauf
2215: D0 0A         291           BNE  DWBS0       ;nein
2217: AD 6B 20      292           LDA  BUFFER+1
221A: C9 25         293           CMP  #MEMH       ;Puffer = Anfang
221C: F0 05         294           BEQ  DWBS1       ;ja
221E: CE 6B 20      295           DEC  BUFFER+1
2221: 18            296  DWBS0    CLC
2222: 60            297           RTS
                    298
                    299  DWBS1    EQU  *           ;Puffer = 1
                    300
2223: A9 00         301  DWNEW    LDA  #MEML       ;Reset Counter
2225: 8D 6A 20      302           STA  BUFFER
2228: A9 25         303           LDA  #MEMH
222A: 8D 6B 20      304           STA  BUFFER+1
222D: 60            305           RTS
                    306
222E: A9 4C         307  DW0      LDA  #'L'        ;DOWNLOAD
2230: 8D 23 20      308           STA  STATE
2233: AD 07 24      309           LDA  GOON        ;sende Start
2236: 20 D6 22      310           JSR  TXMODE
2239: B0 60         311           BCS  DW4
223B: A9 0D         312           LDA  #$0D        ;-CR-
223D: 60            313           RTS
                    314
223E: 20 D6 22      315  DW1      JSR  TXMODE      ;sende Echo
2241: B0 58         316           BCS  DW4
2243: A9 20         317  DW2      LDA  #TOUT
2245: 8D F3 23      318           STA  TIME2
2248: 20 8F 23      319           JSR  RXMODE
224B: B0 4E         320           BCS  DW4         ;Error
224D: 29 7F         321           AND  #$7F
224F: CD 0E 24      322           CMP  CAN
2252: F0 B0         323           BEQ  DWCAN       ;Abbruch
2254: CD F9 23      324           CMP  ETX
2257: F0 41         325           BEQ  DW3         ;Text Ende
2259: CD FA 23      326           CMP  EOT
225C: F0 3C         327           BEQ  DW3         ;Übertragung Ende
225E: CD 0D 24      328           CMP  ETB
2261: F0 37         329           BEQ  DW3         ;Übertragung Ende
2263: CD FE 23      330           CMP  BS
2266: F0 A3         331           BEQ  DWBS        ;Korrektur
2268: CD F8 23      332           CMP  STX0
226B: F0 B6         333           BEQ  DWNEW       ;Start Übertr.
226D: CD F7 23      334           CMP  SOH
2270: F0 B1         335           BEQ  DWNEW       ;Start Übertr.
                    336
2272: CD 00 24      337           CMP  LF
2275: D0 12         338           BNE  DW29        ;kein LF
2277: AD F5 23      339           LDA  LBYTE       ;davor CR
227A: CD 03 24      340           CMP  CR
227D: D0 07         341           BNE  DW28        ;nein
227F: AD 00 24      342           LDA  LF
```

```
2282: 8D F5 23      343           STA  LBYTE
2285: 60            344           RTS
                    345
2286: AD 00 24      346  DW28     LDA  LF
2289: 8D F5 23      347  DW29     STA  LBYTE
228C: 09 80         348           ORA  #$80
228E: 20 69 20      349           JSR  STORE
2291: C9 95         350           CMP  #$95
2293: D0 03         351           BNE  DW2A
2295: 4C B2 20      352           JMP  BUFFH       ;Puffer zu groß
                    353
2298: 18            354  DW2A     CLC
2299: 60            355           RTS
                    356
229A: 38            357  DW3      SEC
229B: 60            358  DW4      RTS
                    359
229C: A2 00         360  ENDW     LDX  #0
229E: BD BE 22      361  ENDW1    LDA  ENDW3,X     ;Text ausgeben
22A1: F0 08         362           BEQ  ENDW2
22A3: 09 80         363           ORA  #$80
22A5: 20 ED FD      364           JSR  COUT
22A8: E8            365           INX
22A9: D0 F3         366           BNE  ENDW1
22AB: AD 1E 20      367  ENDW2    LDA  ERRBYTE
22AE: 20 DA FD      368           JSR  PRBYTE
22B1: A9 42         369           LDA  #'B'        ;Basic Ende
22B3: 8D 23 20      370           STA  STATE
22B6: A9 8D         371           LDA  #$8D
22B8: 20 ED FD      372           JSR  COUT
22BB: 4C 02 21      373           JMP  BUFF        ;Puffer anzeigen
                    374
22BE: 0D            375  ENDW3    DFB  13
22BF: 44 4F 57      376           ASC  'DOWNLOAD BEENDET CODE:'
22C2: 4E 4C 4F 41 44 20 42 45
22CA: 45 4E 44 45 54 20 43 4F
22D2: 44 45 3A
22D5: 00            377           DFB  0
                    378
                    379  ****************************************
                    380  *
                    381  *         TRANSMIT-MODUS
                    382  *
                    383  ****************************************
                    384
22D6: 09 80         385  TXMODE   ORA  #$80
22D8: 8D 1C 20      386           STA  TXBYTE      ;rette Byte
22DB: A9 00         387           LDA  #0
22DD: 8D F1 23      388           STA  TIME0
22E0: 8D F2 23      389           STA  TIME1
22E3: 8D 1E 20      390           STA  ERRBYTE
                    391
22E6: AD A4 C0      392           LDA  STATUS
22E9: 2C EF 23      393  TM1      BIT  RXREADY     ;empfangen
22EC: F0 1B         394           BEQ  TM2
                    395  *        JSR  SERIN       ;lies Byte
22EE: AD A5 C0      396           LDA  DATEN
22F1: 29 7F         397           AND  #$7F
22F3: CD 09 24      398           CMP  WAIT        ;warten
22F6: F0 43         399           BEQ  WAITSUB
22F8: CD 0E 24      400           CMP  CAN         ;Abbruch
22FB: D0 04         401           BNE  NOUT
22FD: A9 82         402           LDA  #$82
22FF: D0 19         403           BNE  TM3
                    404
2301: 09 80         405  NOUT     ORA  #$80
2303: 8D 1D 20      406           STA  RXBYTE
2306: 20 20 20      407           JSR  COUT0       ;;;;;;;;;;
                    408
2309: 2C 00 C0      409  TM2      BIT  KBD         ;Taste gedrückt
230C: 10 11         410           BPL  TM0
230E: 2C 10 C0      411           BIT  KBDSTB
2311: AD 00 C0      412           LDA  KBD
2314: C9 1B         413           CMP  #27         ;ESC?
2316: D0 07         414           BNE  TM0
2318: A9 81         415           LDA  #$81
231A: 8D 1E 20      416  TM3      STA  ERRBYTE     ;merke
231D: 38            417           SEC  ABBRUCH
231E: 60            418           RTS
                    419
231F: AD A4 C0      420  TM0      LDA  STATUS
2322: 2C F0 23      421           BIT  TXREADY     ;Sender leer
2325: F0 C2         422           BEQ  TM1
2327: AD 1C 20      423           LDA  TXBYTE      ;hole Byte
232A: 29 7F         424           AND  #$7F
                    425  *        JSR  SEROUT      ;sende Byte
232C: 8D A5 C0      426           STA  DATEN
```

```
232F: 2C 1F 2Ø  427           BIT  ECHOFLAG
2332: 1Ø Ø5     428           BPL  TMØ1        ;nein
2334: Ø9 8Ø     429           ORA  #$8Ø
2336: 2Ø 2Ø 2Ø  43Ø           JSR  COUTØ
2339: 18        431  TMØ1     CLC              ;alles ok
233A: 6Ø        432           RTS
                433
                434  *-----------------------------------
                435
233B: A9 ØØ     436  WAITSUB  LDA  #Ø
233D: 8D F1 23  437           STA  TIMEØ
234Ø: 8D F2 23  438           STA  TIME1
2343: 8D 1E 2Ø  439           STA  ERRBYTE
2346: A9 2Ø     44Ø           LDA  #TOUT
2348: 8D F3 23  441           STA  TIME2
                442
234B: AD A4 CØ  443  WSØ      LDA  STATUS
234E: 2C EF 23  444           BIT  RXREADY     ;empfangen
2351: FØ 13     445           BEQ  WS1
                446  *        JSR  SERIN       ;lies Byte
2353: AD A5 CØ  447           LDA  DATEN
2356: 29 7F     448           AND  #$7F
2358: CD Ø7 24  449           CMP  GOON        ;weiter
235B: FØ C2     45Ø           BEQ  TMØ
235D: CD ØE 24  451           CMP  CAN         ;Abbruch
236Ø: DØ 9F     452           BNE  NOUT
2362: A9 42     453           LDA  #$42
2364: DØ 11     454           BNE  WS2         ;merke Byte
                455
2366: 2C ØØ CØ  456  WS1      BIT  KBD         ;Taste gedrückt
2369: 1Ø 11     457           BPL  WS3
236B: 2C 1Ø CØ  458           BIT  KBDSTB
236E: AD ØØ CØ  459           LDA  KBD
2371: C9 1B     46Ø           CMP  #27         ;ESC?
2373: DØ Ø7     461           BNE  WS3
2375: A9 41     462           LDA  #$41
2377: 8D 1E 2Ø  463  WS2      STA  ERRBYTE     ;merke
237A: 38        464           SEC              ;Abbruch
237B: 6Ø        465           RTS
                466
237C: EE F1 23  467  WS3      INC  TIMEØ
237F: DØ CA     468           BNE  WSØ
2381: EE F2 23  469           INC  TIME1
2384: DØ C5     47Ø           BNE  WSØ
2386: CE F3 23  471           DEC  TIME2
2389: DØ CØ     472           BNE  WSØ
238B: A9 43     473           LDA  #$43
238D: DØ E8     474           BNE  WS2         ;TIMEOUT
                475
                476  ****************************************
                477  *
                478  *        RECEIVE-MODUS
                479  *
                48Ø  ****************************************
                481
                482  RXMODE   EQU  *
238F: A9 ØØ     483           LDA  #Ø
2391: 8D F1 23  484           STA  TIMEØ
2394: 8D F2 23  485           STA  TIME1
2397: 8D 1E 2Ø  486           STA  ERRBYTE
239A: AD F3 23  487           LDA  TIME2
239D: 8D F4 23  488           STA  TIME3       ;TIMEOUT retten
                489
23AØ: AD A4 CØ  49Ø  RMØ      LDA  STATUS
23A3: 2C EF 23  491           BIT  RXREADY     ;Byte da
23A6: FØ ØD     492           BEQ  RM1
                493  *        JSR  SERIN       ;lies Byte
23A8: AD A5 CØ  494  RMØØ     LDA  DATEN
23AB: 8D F5 23  495  RMØ9     STA  LBYTE
23AE: Ø9 8Ø     496           ORA  #$8Ø
23BØ: 8D 1D 2Ø  497           STA  RXBYTE
23B3: 18        498           CLC  OK
23B4: 6Ø        499           RTS
                5ØØ
23B5: 2C ØØ CØ  5Ø1  RM1      BIT  KBD         ;Taste
23B8: 1Ø 22     5Ø2           BPL  RM4         ;nein
23BA: 2C 1Ø CØ  5Ø3           BIT  KBDSTB
23BD: AD ØØ CØ  5Ø4           LDA  KBD
23CØ: C9 1B     5Ø5           CMP  #27         ;ESC
23C2: DØ Ø7     5Ø6           BNE  RM3
23C4: A9 Ø1     5Ø7           LDA  #1
23C6: 8D 1E 2Ø  5Ø8  RM2      STA  ERRBYTE     ;Abbruch Code
23C9: 38        5Ø9           SEC              ;Fehler
23CA: 6Ø        51Ø           RTS
                511
23CB: C9 Ø2     512  RM3      CMP  #Ø2         ;Ctrl-B
23CD: DØ ØD     513           BNE  RM4
```

```
23CF: AD 23 2Ø  514           LDA  STATE
23D2: C9 55     515           CMP  #'U'        ;UPLOAD?
23D4: FØ Ø6     516           BEQ  RM4         ;keine Anzeige
23D6: 2Ø Ø2 21  517           JSR  BUFF
23D9: 4C AØ 23  518           JMP  RMØ
                519
23DC: EE F1 23  52Ø  RM4      INC  TIMEØ
23DF: DØ BF     521           BNE  RMØ
23E1: EE F2 23  522           INC  TIME1
23E4: DØ BA     523           BNE  RMØ
23E6: CE F3 23  524           DEC  TIME2
23E9: DØ B5     525           BNE  RMØ
23EB: A9 Ø3     526           LDA  #3
23ED: DØ D7     527           BNE  RM2         ;TIMEOUT
                528
                529  *-----------------------------------
                53Ø
23EF: Ø1        531  RXREADY  DFB  1           ;STATUS
23FØ: Ø2        532  TXREADY  DFB  2
23F1: ØØ        533  TIMEØ    DFB  Ø           ;Time-Count
23F2: ØØ        534  TIME1    DFB  Ø
23F3: ØØ        535  TIME2    DFB  Ø
23F4: ØØ        536  TIME3    DFB  Ø
23F5: ØØ        537  LBYTE    DFB  Ø           ;letztes Byte
                538
23F6: ØØ        539  CTRL     DFB  Ø
23F7: Ø1        54Ø  SOH      DFB  1
23F8: Ø2        541  STXØ     DFB  2
23F9: Ø3        542  ETX      DFB  3
23FA: Ø4        543  EOT      DFB  4
23FB: Ø5        544  ENQ      DFB  5
23FC: Ø6        545  ACK      DFB  6
23FD: Ø7        546  BEL      DFB  7
23FE: Ø8 Ø9     547  BS       DFB  8,9
24ØØ: ØA ØB ØC  548  LF       DFB  1Ø,11,12
24Ø3: ØD        549  CR       DFB  13
24Ø4: ØE        55Ø  SO       DFB  14
24Ø5: ØF 1Ø     551  SI       DFB  15,16
24Ø7: 11 12     552  GOON     DFB  17,18
24Ø9: 13 14     553  WAIT     DFB  19,2Ø
24ØB: 15        554  NAK      DFB  21
24ØC: 16        555  SYN      DFB  22
24ØD: 17        556  ETB      DFB  23
24ØE: 18 19 1A  557  CAN      DFB  24,25,26,27,28,29,3Ø,31
2411: 1B 1C 1D 1E 1F
```

1Ø46 Bytes

Der nächste Peeker Heft 5/1985 erscheint am 22.04.85

In diesem zweiten Teil vermitteln wir das systematische Grundgerüst der Befehle, Parameter und Dateitypen des ProDOS-BASIC.SYSTEMs. Falls Sie Apple-Neuling sind und deshalb das alte DOS 3.3 nicht kennen, so werden Sie einiges zunächst nicht verstehen. Lassen Sie sich jedoch dadurch nicht entmutigen, denn diese Folge ist zum späteren Nachschlagen sowie als Gesamtüberblick gedacht. In den nachfolgenden Teilen unserer ProDOS-Serie wird jeder Befehl detailliert mit vereinfachten Beispielprogrammen erläutert.

ProDOS für Anfänger

von Ulrich Stiehl

Teil 2: Das definitorische Grundgerüst

1. BEFEHLSTYPEN

Befehle an das BASIC.SYSTEM bestehen aus Befehlsnamen, die wie unter DOS 3.3 in zwei Formen erteilt werden können:

Direkte Befehle (Nicht-RUN-Modus):
Diese können über die Tastatur eingegeben werden. Vor dem Befehl muß implizit und nach dem Befehl explizit ein Return (Rtn, Ctrl-M) stehen, z.B.
(Rtn implizit)
CATALOG Rtn
Das Return vor dem Befehl gilt immer dann als implizit vorhanden, wenn der Cursor nach dem Prompt-Zeichen „Ü" am linken Bildschirmrand blinkt. Führende Leertasten wären erlaubt, z.B.
  CATALOG
Dagegen sind andere Zeichen nicht zulässig, z.B.
:CATALOG
Schließlich sind auch *mehrere* BASIC.SYSTEM-Befehle innerhalb *derselben* Eingabezeile nicht möglich, z.B.
CATALOG: CATALOG

Indirekte Befehle (RUN-Modus):
Diese werden während des laufenden, nicht-compilierten Applesoft-Programms an das BASIC.SYSTEM erteilt. Man beachte also, daß im Gegensatz zu DOS 3.3 kein Integer-Basic-, kein compiliertes Applesoft- und auch kein Assembler-Programm als RUN-Modus gilt. Dem Befehl, der im RUN-Modus in Anführungszeichen gesetzt wird, geht (normalerweise) ein implizites Return sowie auf alle Fälle stets ein Ctrl-D = CHR$ (4) voraus. Ferner muß der Befehl durch ein explizites Return abgeschlossen werden, z.B.
10 PRINT: PRINT CHR$ (4) "CATALOG"
Das Ctrl-D muß das erste Zeichen einer PRINT-Anweisung sein, die ihrerseits entweder unmittelbar der Zeilennummer oder unmittelbar dem Doppelpunkt (Statement-Separator) folgen muß. Mehrfach-Befehle in derselben Programmzeile sind zulässig, z.B.
10 PRINT CHR$ (4) "CAT": PRINT CHR$ (4) "CAT"
Der Befehl in Anführungszeichen kann durch eine String-Variable ersetzt werden, z.B.
10 X$ = CHR$ (4) + "CAT"
20 PRINT X$
Im Gegensatz zu DOS 3.3 befindet sich das BASIC.SYSTEM stets im sog. nicht-sichtbaren Pseudo-Trace-Modus, der nach jeder neuen Zeilennummer sowie nach jedem Doppelpunkt wirksam wird. Deshalb sind einige syntaktische Feinheiten gegenüber DOS 3.3 zu beachten:

Beispiel 1:
10 X$ = CHR$ (4) + "CATALOG"
20 PRINT "AAA";: PRINT X$
Hier wird der CATALOG-Befehl ausgeführt, obwohl nach „AAA" ein Semikolon folgt und somit kein implizites Return vorangeht. Grund: PRINT X$ steht nach einem Doppelpunkt. Bei DOS 3.3 würde demgegenüber nur „AAACATALOG" am Bildschirm angezeigt und somit der CATALOG-Befehl ignoriert.

Beispiel 2:
10 X$ = CHR$ (4) + "CATALOG"
20 Y$ = ",D1"
30 PRINT X$; Y$: REM richtig
40 PRINT X$;: PRINT Y$: REM falsch
Hier sind Befehlsname und Parameter getrennte Strings. Werden diese wie in Zeile 30 als ein einziger PRINT-Befehl ausgeführt, dann ist das BASIC.SYSTEM „zufrieden". Erfolgt indessen eine Aufsplittung wie in Zeile 40, so führt dies zu einem Syntax-Error. Unter DOS 3.3 wäre demgegenüber eine Aufteilung zulässig.

Beispiel 3:
10 R$ = CHR$ (13): REM Ctrl-M
20 C$ = CHR$ (4) + "CATALOG"
30 PRINT R$: PRINT C$
40 PRINT R$; C$
Hier wird zusätzlich ein nacktes Return als String-Variable definiert. In Zeile 30 wäre das BASIC.SYSTEM „zufrieden" (im Gegensatz zu der falschen Angabe in dem englischen ProDOS-Handbuch der Firma Apple). In Zeile 40 hingegen würde eine Fehlermeldung entstehen, weil nunmehr Ctrl-D nicht mehr das *erste* Zeichen der Zeichenkette wäre. Unter DOS 3.3 würde Zeile 40 fehlerfrei ausgeführt.

Als Fazit halten wir fest, daß ein indirekter BASIC.SYSTEM-Befehl nur dann korrekt ausgeführt wird, wenn das Ctrl-D das *erste* Zeichen der PRINT-Anweisung ist und wenn diese bis zum nächsten Statement-Separator („:" oder $00 als End of Line) *komplett* gePRINTet wird und nicht durch ein Semikolon, das das Return unterdrükken würde, abgeschlossen wird. Ferner sei darauf hingewiesen, daß das Ctrl-D wahlweise als CHR$ (4) mit Bit 7 off oder als CHR$ (132) mit Bit 7 on (4 + 128 = 132) definiert werden könnte.

Die Textdatei-Befehle OPEN, READ, WRITE, POSITION und APPEND sind übrigens reine Indirekt-Befehle, während alle anderen Befehle sowohl im RUN- wie

auch im Nicht-RUN-Modus an das BASIC.SYSTEM erteilt werden können.

Im BASIC.SYSTEM sind folgende Befehle implementiert worden:

Allgemeine Befehle
CAT: Anzeige des Catalogs in 40 Z/Z
CATALOG: Anzeige des Catalogs in 80 Z/Z
PREFIX: Definition des Default-Directory
CREATE: Anlage einer neuen Datei
RENAME: Umbenennung einer Datei
VERIFY: Dateiname prüfen
DELETE: Datei löschen
LOCK: Dateiname mit Sternchen versehen (Schreibschutz)
UNLOCK: Sternchen entfernen
PR#: Änderung der Ausgabe-Vektoren
IN#: Änderung der Eingabe-Vektoren
BYE (Reboot bei ProDOS 1.1.1)

Applesoft-Befehle
RUN: Starten eines Applesoft-Programms
LOAD: Laden eines Applesoft-Programms
SAVE: Speichern eines Applesoft-Programms
CHAIN: Starten eines Applesoft-Moduls
STORE: Speichern aller Applesoft-Variablen
RESTORE: Laden aller Applesoft-Variablen

Textdatei-Befehle
OPEN: Öffnen einer Textdatei
READ: Lesen von einer Textdatei
WRITE: Schreiben auf eine Textdatei
APPEND: Textdatei erweitern
POSITION: Textdatei-Feldzeiger ändern
CLOSE: Schließen einer Textdatei
FLUSH: Puffer auf Textdatei übertragen
FRE: String-Garbage-Collection
EXEC: Textbefehlsdatei starten

Binärdatei-Befehle
BRUN: Maschinenprogramm starten
BLOAD: (Binär)datei laden
BSAVE: (Binär)datei speichern

Allgemeiner Start-Befehl
–: Ersatz für RUN, BRUN, EXEC

Diejenigen, die unter DOS 3.3 programmiert haben, werden bemerken, daß die Befehle FP, INT, MON, NOMON, MAXFILES und INIT fehlen. FP und INT mußten entfallen, weil das BASIC.SYSTEM nur für Applesoft und nicht für Integer-Basic gedacht ist. MON und NOMON fehlen wahrscheinlich aus Platzgründen. Dasselbe gilt für den INIT-Befehl. MAXFILES ist nicht mehr möglich, weil mit dynamischen I/O-Puffern gearbeitet wird.
Dafür gibt es einige völlig neue Befehle, nämlich CAT, CREATE, FLUSH, PREFIX, STORE, RESTORE und FRE. Darüber hinaus wurden die meisten gleichnamigen DOS-Befehle unter ProDOS funktionell erweitert und verbessert.

2. PARAMETERTYPEN

Die Parameter sind Zusätze zu den Befehlsnamen. Sie bestehen aus dem Parameter-Buchstaben und dem Parameter-Wert und werden durch Kommas getrennt, wobei bei mehreren Befehlsparametern deren Reihenfolge beliebig ist, z.B. CATALOG, S6, D2 oder CATALOG, D2, S6.
Es gibt zwei Ausnahmen:
a) Namen werden durch Schrägstriche getrennt, z.B. CAT/DISK/DEMOS.
b) Der Parameter-Wert für die Slot-Nummer wird bei den PR#/IN#-Befehlen direkt an das Nummernkreuz angehängt, z.B. PR#1

/n: Name (max. 64 Zeichen)
s: Slot bei PR#s, IN#s (1-7)
,Ss: Slot (1-7)
,Dd: Drive (1-2)
,Aa: Anfangsadresse ($0000-$BFFF)
,Ee: Endadresse ($0000-$BFFF)
,Ll: Länge ($0001-$C000)
,Bb: Byte-Offset ($000000-$FFFFFF)
,Rr: Recordnummer (0-16777215)
,Ff: Feldnummer (0-65535)
,§z:Zeilennummer (0-63999)
,Tt: Dateityp (3-Buchstaben-Kürzel)

Beispiele
CATALOG/VOLUME/BRIEFE
PR#3
CAT, S3, D2
BSAVE BILDER, A$2000, E$5FFF
BSAVE BILDER, A$2000, L$4000
BLOAD BILDER, A$4000, B$2000
PRINT CHR$ (4) "READ FILE, R1000"
PRINT CHR$ (4) "READ FILE, F3"
LOAD PROGRAMM, §2000
BLOAD/VOLUME, A$1000, TDIR

3. DATEITYPEN

Wie bereits im ersten Teil der Serie erwähnt wurde, kann ein ProDOS-Datenträger (Diskette, Festplatte) im Rahmen der maximalen Speicherkapazität von 32 M beliebig viele Dateien umfassen, wobei eine einzelne Datei bis zu 16 M groß sein kann. Die ProDOS-Dateitypen lassen sich in drei Gruppen einteilen:

SOS-Dateitypen: Dies waren die Dateitypen des „Sophisticated Operating System" des Apple III, die inzwischen keine Rolle mehr spielen, da der Apple III nicht mehr produziert wird.
Selbstdefinierte Dateitypen: Ein Anwenderprogramm könnte sich seine eigenen Dateitypen mit den Typ-Nummern im Bereich $F1 bis $F8 definieren. Ein Bedarf hierfür besteht jedoch normalerweise nicht.
Standard-Dateitypen: Dies sind die normalen oder allgemeinen Dateitypen, und zwar u.a.:

SYS: Systemprogramm
TXT: Textdatei
BIN: Binärdatei
BAS: Applesoft-Programm
VAR: Applesoft-Variablendatei
DIR: Subdirectory

Der Dateityp besteht aus dem Parameter „T" und dem jeweiligen 3-Buchstaben-Kürzel, z.B. TSYS, TTXT, TBIN, TBAS, TVAR, TDIR usw.

SYS-Dateien umfassen neben PRODOS und BASIC.SYSTEM weitere mögliche Systemprogramme, z.B. FILER, CONVERT u.a. SYS-Dateien werden mit
BLOAD SYSTEMDATEI, TSYS, A$2000
geladen und mit
–SYSTEMDATEI oder
BRUN SYSTEMDATEI, A$2000, TSYS
gestartet.

TXT-Dateien sind normale ASCII-Dateien. Falls diese mit den BASIC.SYSTEM-Befehlen OPEN, WRITE, READ, APPEND, POSITION, FLUSH und CLOSE bearbeitet werden, ist bei jedem ASCII-Zeichen das Bit 7 auf 0 gesetzt, also z.B. $0D statt $8D für Return. Unter DOS 3.3 war es genau umgekehrt. Programmtechnisch können TXT-Dateien sequentielle oder Random-Dateien sein.

BIN-Dateien sind Binärdateien aller Art, z.B. Maschinenprogramme, Hires-Bilder usw. Im Gegensatz zu DOS 3.3 steht die Startadresse und Länge einer BIN-Datei nicht im ersten Dateiblock, sondern im Catalog-Dateieintrag. Auf BIN-Dateien kann ohne den TBIN-Zusatz mit BSAVE, BLOAD und BRUN zugegriffen werden. Doch sind die BLOAD/BSAVE-Befehle im BASIC.SYSTEM dergestalt generalisiert worden, daß *jede* andere Dateiart mit dem entsprechenden Typzusatz geladen und gespeichert werden kann, z.B.
BLOAD TEXTFILE, TTXT, A$1000
BLOAD BASICPROGRAMM, TBAS, A$0801

Directories (Volume-Directory und Subdirectories) können ebenfalls mit BLOAD geladen, nicht jedoch (aus Sicherheitsgründen) mit BSAVE zurückgespeichert werden:
BLOAD/VOLUMEDIR, A$2000, TDIR
BLOAD/VOLUMEDIR/SUBDIR, A$1000, TDIR

BAS-Dateien sind Applesoft-Programme, auf die mit den Befehlen SAVE, LOAD, RUN und CHAIN zugegriffen wird. Im Gegensatz zu DOS 3.3 steht die Länge des Applesoft-Programms nicht im ersten Dateiblock, sondern im Catalog-Dateieintrag.

VAR-Dateien umfassen die Namen und Werte *aller* Variablen eines Applesoft-Programms und werden mit STORE/VARIABLENDATEI gespeichert und mit RESTORE/VARIABLENDATEI geladen.

DIR-Dateien sind Inhaltsverzeichnisse, und zwar entweder das Hauptinhaltsverzeichnis (Volume-Directory) oder eines der beliebig vielen Unterinhaltsverzeichnisse (Subdirectories).

4. CAT, CATALOG, PREFIX, CREATE

CAT/VOLUME/DIR, Ss, Dd
CATALOG/VOLUME/DIR, Ss, Dd
PREFIX/VOLUME/DIR, Ss, Dd
CREATE/VOLUME/DIR,Tt, Ss, Dd

Nach diesem relativ trockenen, aber notwendigen theoretischen Gesamtüberblick können wir uns nunmehr den einzelnen Befehlen im Detail zuwenden. Nachdem wir die USERS.DISK (oder eine andere ungeschützte ProDOS-Diskette) mit PR#6 o.ä. gebootet und dann ein ggf. vorhandenes Menü verlassen haben, befinden wir uns im Nicht-RUN-Modus, so daß wir jetzt über die Tastatur Befehle eingeben können. Mit
CAT
oder mit
CATALOG
können wir uns nunmehr den Catalog oder das Directory (Inhaltsverzeichnis) der Diskette in 40 Zeichen/Zeile (CAT) oder in 80 Z/Z ansehen. Das Listing „PRODOS.DIRECTORY“ zeigt einen solchen ProDOS-Catalog in 80 Z/Z. Betrachten wir hier den ersten Dateieintrag „PRODOS“:

Sternchen: Ein vorangestelltes Sternchen zeigt an, daß die Datei „PRODOS“ schreibgeschützt ist. Fehlte das Sternchen, dann könnte die Datei gelöscht werden.
Dateiname: Der Name „PRODOS“ besteht hier aus 6 Großbuchstaben. Insgesamt kann eine Dateiname bis zu 15 Zeichen umfassen, und zwar nur Großbuchstaben, Ziffern und Punkte, aber keine Leerzeichen, Ctrl-Buchstaben, Kleinbuchstaben und sonstigen Sonderzeichen. Das erste Zeichen des Dateinamens muß ein Großbuchstabe sein. Beispiele:
SPIELPROGRAMM10 (mit Ziffern)
BASIC.SYSTEM (mit Punkt)
MEIN PROGRAMM (falsch, da Leertaste)
BASIC-PROGRAMM (falsch, da Bindestrich)
Dateityp: 3-Buchstaben-Kürzel, hier „SYS“
Blockanzahl: Anzahl der von der Datei belegten Blocks, hier „31“.
Datum der Modifikation: In der Form TT-MON-JJ, hier 1-JAN-84.
Uhrzeit der Modifikation: In der Form HH:MM.
Datum der Anlage: Tag, an dem die Datei erstmals angelegt („kreiert“) wurde, hier „<NO DATE>“.
Uhrzeit der Anlage: Diese fehlt hier.
Dateilänge: Anzahl der Bytes der Datei, hier „15360“. ENDFILE = End of File steht sowohl für die Gesamtanzahl der Bytes wie auch für die absolute Position des letzten Bytes der Datei, d.h. „PRODOS“ umfaßt 15360 Bytes, und das letzte Byte dieser Datei ist das 15359ste Byte, da die Zählung mit Null beginnt.
Zusatzinfo: Die Zusatzinformation wird im ProDOS-Catalog als SUBTYPE bezeichnet, während das technische Manual von AUXTYPE (Auxiliary Type) spricht. Bei BIN-Dateien steht hier die Startadresse. Bei SYS-Dateien usw. bleibt das Feld leer.

Unter ProDOS gibt es zwei Arten von Inhaltsverzeichnisses:
Volume-Directory: Das Volume-Directory ist das Hauptinhaltsverzeichnis. Es belegt auf der Diskette die Blocks 2-5 und kann insgesamt 51 Dateieinträge aufnehmen (12 im ersten Block und je 13 in den drei weiteren Blocks = 12 + 3 * 13 = 51). Für 140K-Disketten ist dies oft ausreichend.
Subdirectory: Neben dem einzigen Volume-Directory können mit dem CREATE-Befehl beliebig viele Subdirectories angelegt werden, wobei jedes Subdirectory im übrigen beliebig viele Dateieinträge umfassen kann. Ein Subdirectory ist in dem Volume-Directory durch den Dateityp „DIR“ gekennzeichnet. Für Subdirectories gibt es keine festen Blocks auf der Diskette, d.h. sie können sich überall dort befinden, wo auf der Diskette gerade Platz ist. Übrigens kann auch das Subdirectory selbst wieder Sub-Subdirectories enthalten (hierarchische Catalog-Struktur), doch sollte man diese möglichst vermeiden, denn je mehr (verschachtelte) Subdirectories, desto länger ist der Diskettenzugriff.

Unter ProDOS lassen sich drei Arten von Namen unterscheiden:

Volume-Name: Dies ist der beim Formatieren festgelegte Name der Diskette selbst, wie er aus dem Volume-Directory ersichtlich ist (bei unserem Listing „/STIEHL“).
Subdirectory-Name: Dies ist der Name eines von beliebig vielen Subdirectories.
File-Name: Dies ist der eigentliche Name der Datei.

Die Summe aus „/“ + Volume-Name + „/“ + Subdirectory-Name + „/“ + File-Name bildet den sog. Pfadnamen (Pathname) oder vollständigen Namen, der inklusive Schrägstriche maximal 64 Zeichen umfassen kann. Die Namensbestandteile müssen durch Schrägstriche abgegrenzt werden, wobei der Schrägstrich nach dem letzten Bestandteil fakultativ ist, also
CAT/STIEHL/DIR/
oder
CAT/STIEHL/DIR

Mit dem PREFIX-Befehl kann man das Präfix, d.h. den Dateinamenvorspann festlegen. Danach genügt wie bei DOS 3.3 der eigentliche Dateiname (ohne Schrägstrich als Zusatz zu den Befehlen, also statt z.B.
LOAD/VOL/SUBDIR/PROGRAMM.XYZ
zunächst
PREFIX/VOL/SUBDIR
und dann nur noch
LOAD PROGRAMM.XYZ

Beispiele:

1. NEW
2. 10 HOME
3. CREATE/STIEHL/DIR, TDIR
4. PREFIX/STIEHL/DIR
5. SAVE DEMO
6. CATALOG
7. PREFIX/STIEHL
8. CATALOG
9. PREFIX

1. Mit NEW löschen wird zunächst ein möglicherweise noch im Speicher befindliches Applesoft-Programm.

2. Mit 10 HOME legen wir jetzt ein Testprogramm an.
3. Mit CREATE/STIEHL/DIR, TDIR erstellen wir jetzt auf der Diskette mit dem Volume-Namen STIEHL ein Subdirectory mit dem Namen DIR.
4. Mit PREFIX/STIEHL/DIR legen wir das Präfix „/STIEHL/DIR/" fest.
5. Der Befehl SAVE DEMO speichert damit das Testprogramm genauso im Subdirectory DIR, als hätten wir SAVE/STIEHL/DIR/DEMO gesagt.
6. Mit CATALOG können wir automatisch das Subdirectory DIR ansehen.
7. Mit PREFIX/STIEHL können wir wieder zum Volume-Directory zurückschalten.
8. Der CATALOG-Befehl zeigt damit wieder das Volume-Directory an.
9. Der PREFIX-Befehl zeigt jetzt (wieder) „/STIEHL/" an.

Der PREFIX-Befehl bietet folgende Besonderheiten:

PREFIX/VOLUME/SUB legt das neue Präfix fest.
PREFIX, S6, D1 legt das *Volume*-Directory der Diskette in Slot 6, Drive 1 als neues Präfix fest.
PREFIX oder *10 PRINT CHR$ (4) „PREFIX": INPUT P$* zeigt im Nicht-RUN-Modus das momentane Präfix an oder weist im RUN-Modus bei einem unmittelbar folgenden INPUT der String-Variablen – der Name der Variablen ist beliebig – das momentane Präfix zu.
PREFIX/ entfernt das Präfix insgesamt. Danach verhält sich das BASIC.SYSTEM wie DOS 3.3, d.h. es können jetzt mit den Parametern Ss und Dd die jeweils aktiven Slots und Drives (als Default-Werte) bestimmt werden.

Beispiel:

1. PREFIX, S6, D1
2. CAT, D2
3. CAT
4. PREFIX/
5. CAT, D2
6. CAT

1. Zunächst wird der Volume-Name von S6, D1 als Präfix festgelegt.
2. CAT, D2 zeigt den Catalog der Diskette in Drive 2 an.
3. CAT ohne Parameter zeigt jedoch jetzt wieder Drive 1 an, da das alte Präfix noch gültig ist (Dies weicht von DOS 3.3 ab).
4. PREFIX/ entfernt jedes Präfix.
5. CAT, D2 zeigt Catalog von Drive 2 an.
6. CAT zeigt jetzt erneut Drive 2 an (Dies entspricht jetzt DOS 3.3).

PRODOS.DIRECTORY

```
/STIEHL
 NAME            TYPE  BLOCKS  MODIFIED         CREATED          ENDFILE SUBTYPE
123456789Ø123456789Ø123456789Ø123456789Ø123456789Ø123456789Ø123456789Ø123456789
1                18       27  31         42    48          59       67    73
*                *        *   *          *     *           *        *     *
*PRODOS          SYS      31   1-JAN-84  Ø:ØØ  <NO DATE>           15360
*BASIC.SYSTEM    SYS      21  15-NOV-83  Ø:ØØ  <NO DATE>           1Ø24Ø
*FILER           SYS      51  <NO DATE>        <NO DATE>           256ØØ
*CONVERT         SYS      42   1-NOV-83  Ø:ØØ  <NO DATE>           2Ø481
*CLEAR           BIN       1  <NO DATE>        <NO DATE>             184 A=$Ø3ØØ
*LOE             BIN       1  <NO DATE>        <NO DATE>             111 A=$Ø29Ø
*LCREAD          BIN       1  <NO DATE>        <NO DATE>             134 A=$Ø3ØØ
*PEEKER          BIN       1  <NO DATE>        <NO DATE>             448 A=$Ø8Ø3
*BLOCKREADER     BIN       1  <NO DATE>        <NO DATE>             312 A=$Ø8Ø3
*BLOCKTRACER     BIN       1  <NO DATE>        <NO DATE>             149 A=$Ø3ØØ
*STARTUP         BAS       1  23-DEC-84  Ø:ØØ  <NO DATE>             176
*MACROS          BAS       3  23-DEC-84  Ø:ØØ  23-DEC-84  Ø:ØØ       882
*RAMKARTE        TXT       1  21-JUN-84  Ø:ØØ  <NO DATE>             399 R= 4Ø96
*PRODOS.EDITOR   BIN      13   1-AUG-84  Ø:ØØ  <NO DATE>            57Ø7 A=$3FA5
*PRODOS.DATUM    BIN       1  21-JUN-84  Ø:ØØ  <NO DATE>             267 A=$6ØØØ
*FUSION          TXT       1  21-JUN-84  Ø:ØØ  <NO DATE>             326 R= 4Ø96
*PRODOS.INIT8Ø   BIN       9  <NO DATE>        <NO DATE>            4Ø96 A=$8ØØØ
*BLOCK28ØPATCH   BIN       1  <NO DATE>        <NO DATE>             121 A=$Ø3ØØ
*RAMSUCHER       BIN       1  21-JUN-84  Ø:ØØ  <NO DATE>             222 A=$Ø2FØ
*BLOCKTRACER1    BIN       1   2-SEP-84  Ø:ØØ   2-SEP-84  Ø:ØØ       147 A=$Ø3ØØ
*QUICKCOPY       BIN       5  <NO DATE>        <NO DATE>            1881 A=$Ø8AE
*PRODOS.COPYOBJ  BIN       4  <NO DATE>        <NO DATE>            15Ø2 A=$ØBB8
*PRODOS.COPYA    BAS       3  <NO DATE>        <NO DATE>              65Ø
*FORMAT.LC.START BAS       1  13-JAN-85  Ø:ØØ  13-JAN-85  Ø:ØØ       481
*FORMAT.LC       BIN       8  <NO DATE>        <NO DATE>            3232 A=$8ØBB
 EINTRAG.ANALYSE BAS       6  13-JAN-85  Ø:ØØ  <NO DATE>            2112
 CAT.SAVER       BAS       1  <NO DATE>        <NO DATE>             345
 CAT.ARRAY       BAS       1  <NO DATE>        <NO DATE>             385
 EINTRAG.SUCHER  BAS       1  <NO DATE>        <NO DATE>             333
 CAT.SAVE        TXT       7  <NO DATE>        <NO DATE>            2571 R=    Ø

BLOCKS FREE:   53     BLOCKS USED:  227     TOTAL BLOCKS:  28Ø
```

CAT.ARRAY

```
  1ØØ  GOSUB 6ØØØØ: END
6ØØØØ  HOME : REM CAT.ARRAY
6ØØ1Ø  PRINT CHR$ (4);"PREFIX": INPUT P$
6ØØ2Ø  DIM D$(51):D1 = Ø:D2 = 51
6ØØ3Ø  PRINT CHR$ (4);"OPEN";P$;",TDIR"
6ØØ4Ø  PRINT CHR$ (4);"READ";P$
6ØØ5Ø  INPUT D1$: INPUT D2$: INPUT D3$
6ØØ6Ø  D1 = D1 + 1: IF D1 > D2 THEN 6ØØ8Ø
6ØØ7Ø  INPUT D$(D1): IF D$(D1) < > "" THEN 6ØØ6Ø
6ØØ8Ø  INPUT D3$: PRINT CHR$ (4);"CLOSE";P$
6ØØ9Ø  PRINT "Diskette ";D1$: REM Prefix
6Ø1ØØ  PRINT : PRINT LEFT$ (D2$,39): REM Kopfzeile
6Ø11Ø  PRINT :D1 = D1 - 1
6Ø12Ø  FOR X = 1 TO D1
6Ø13Ø  IF MID$ (D$(X),18,3) = "BAS" THEN PRINT LEFT$ (D$(X),39)
6Ø14Ø  NEXT
6Ø15Ø  PRINT : PRINT LEFT$ (D3$,39): REM Fußzeile
6Ø16Ø  RETURN
```

CAT.SAVER

```
1ØØ  HOME : REM CAT.SAVER
11Ø  PRINT CHR$ (4);"PREFIX": INPUT P$
12Ø  DIM D$(51):D1 = Ø:D2 = 51
13Ø  PRINT CHR$ (4);"OPEN";P$;",TDIR"
14Ø  PRINT CHR$ (4);"READ";P$
15Ø  INPUT D1$: INPUT D2$: INPUT D3$
16Ø  D1 = D1 + 1: IF D1 > D2 THEN 18Ø
17Ø  INPUT D$(D1): IF D$(D1) < > "" THEN 16Ø
18Ø  INPUT D3$: PRINT CHR$ (4);"CLOSE";P$
19Ø  PRINT CHR$ (4);"OPEN CAT.SAVE": PRINT CHR$ (4);"WRITE CAT.SAVE"
2ØØ  PRINT D1$
21Ø  PRINT : PRINT D2$
22Ø  PRINT :D1 = D1 - 1
23Ø  FOR X = 1 TO D1
24Ø  PRINT D$(X)
25Ø  NEXT
26Ø  PRINT : PRINT D3$
27Ø  PRINT CHR$ (4);"CLOSE"
```

Default-Werte (Ersatzwerte)

a) DOS-3.3-Emulation-Default: Wenn wir mit PREFIX/ das Präfix ganz entfernen, so „weiß" ProDOS nichts mehr vom Präfix und verhält sich hinsichtlich der Slot-Drive-Default-Werte wie DOS 3.3: Der zuletzt angesprochene Slot/Drive bleibt solange aktiv, bis explizit ein neuer Slot/Drive angegeben wird.
b) ProDOS-Default: Das mit PREFIX/VOLUME.NAME/SUBDIR.NAME definierte Default-Präfix bleibt solange erhalten, bis es explizit neu definiert wird. Slot- und Drive-Parameter bei BASIC.SYSTEM-Befehlen ändern nicht das Präfix (außer natürlich durch den Präfix-Befehl PREFIX, Ss, Dd selbst).

5. ERLÄUTERUNG ZU DEN PROGRAMMEN

Die Beispielprogramme gehen etwas über den bislang vermittelten ProDOS-Stoff hinaus und sollten deshalb hier nur kursorisch erklärt werden.

CAT.ARRAY

CAT.ARRAY, das als Unterroutine gedacht ist, gibt Ihnen die Möglichkeit, das gesamte Directory, wie man es beim CATALOG-Befehl sieht, in einem String-Array abzulegen, auf den Sie dann gezielt zugreifen können. So sind beispielsweise die Zeilen 60100, 60130 und 60150 dergestalt modifiziert, daß nur eine Catalog-Anzeige in 40 Z/Z erfolgt. Ferner werden durch Zeile 60130 selektiv nur die BAS-Dateien angezeigt. In dem Listing PRODOS.DIRECTORY ist eine „Zählzeile 12345..." eingefügt, die es Ihnen leicht ermöglicht, auf die jeweiligen Positionen der einzelnen Dateieintrag-Strings D$(X) mit MID$ zuzugreifen. Falls Sie das Programm auf umfangreiche Subdirectories anwenden wollen, so erhöhen Sie bitte die DIM-Anweisung in Zeile 60020.

CAT.SAVER

CAT.SAVER entspricht CAT.ARRAY, doch wird der Catalog nunmehr nicht am Bildschirm angezeigt, sondern auf der Diskette unter dem TXT-File „CAT.SAVE" gespeichert.

EINTRAG.SUCHER

Der EINTRAG.SUCHER sucht einen per INPUT eingegebenen, beliebigen Dateieintrag aus einem nicht inhaltlich bekannten Directory heraus und vermeidet damit die Programmunterbrechung für den Fall, daß der gesuchte Name nicht existiert.

EINTRAG.SUCHER

```
  10Ø  INPUT "Dateieintrag: ";S$
  11Ø  GOSUB 61ØØØ: END
61ØØØ  REM EINTRAG.SUCHER
61Ø1Ø  PRINT CHR$ (4);"PREFIX": INPUT P$
61Ø2Ø  PRINT CHR$ (4);"OPEN";P$;",TDIR"
61Ø3Ø  PRINT CHR$ (4);"READ";P$
61Ø4Ø  L = LEN (S$)
61Ø5Ø  INPUT D$: INPUT D$: INPUT D$: REM Ignorieren!
61Ø6Ø  F = Ø: INPUT D$: IF MID$ (D$,2,L) = S$ AND MID$ (D$,L + 2,1) = " "
       THEN F = 1: GOTO 61Ø8Ø: REM F=Such-Flag
61Ø7Ø  IF D$ < > "" THEN 61Ø6Ø
61Ø8Ø  PRINT CHR$ (4);"CLOSE";P$
61Ø9Ø  IF F = 1 THEN PRINT P$;S$;" da!": GOTO 611Ø
611Ø  PRINT P$;S$;" fehlt!"
6111Ø  RETURN
```

EINTRAG.ANALYSE

```
1ØØ  REM EINTRAG.ANALYSE
11Ø  REM U.Stiehl/1985
12Ø  PRINT CHR$ (4);"PR#3"
13Ø  PRINT CHR$ (4);"CATALOG"
14Ø  INPUT "Dateieintrag: ";S$: IF S$ = "" THEN 14Ø
15Ø  HOME : GOSUB 53Ø
16Ø  REM 768 = Byte
17Ø  REM CALL 769 = Hexbyte
18Ø  REM CALL 776 = Low Nibble
19Ø  REM CALL 783 = High Nibble
2ØØ  DATA Ø,173,Ø,3,32,218,253,96,173,Ø
21Ø  DATA 3,32,227,253,96,173,Ø,3,74,74
22Ø  DATA 74,74,32,227,253,96
23Ø  RESTORE
24Ø  FOR X = 768 TO 793: READ Y: POKE X,Y: NEXT
25Ø  REM Dateieintrag per Binärfile suchen
26Ø  A = 8192:B = Ø:L = 512: REM Address, Byte, Length
27Ø  O = 4 + 39: REM Offset für nächsten Eintrag
28Ø  PRINT CHR$ (4)"BLOAD";P$;",TDIR,A"A";",B";B;",L"L
29Ø  X = 1
3ØØ  ON PEEK (A + O + X) < > ASC ( MID$ (S$,X,1)) GOTO 31Ø:X = X + 1:
     ON X < = LL GOTO 3ØØ: GOTO 34Ø
31Ø  O = O + 39: IF O < 511 THEN 29Ø
32Ø  IF  PEEK (A + 2) < > Ø OR  PEEK (A + 3) < > Ø THEN B = B + L:O = 4:
     GOTO 28Ø: REM Nächster Block
33Ø  PRINT "Nicht gefunden!": END
34Ø  PRINT "Gefunden!": IF F = Ø THEN PRINT "Jedoch gelöschte Datei!"
35Ø  REM Komponenten des Dateieintrags
36Ø  PRINT : PRINT "Speichertyp: $";: POKE 768, PEEK (A + O + Ø): CALL 783:
     REM High Nibble
37Ø  PRINT : PRINT "Namenslänge: $";: POKE 768, PEEK (A + O + Ø): CALL 776:
     REM Low Nibble
38Ø  PRINT : PRINT "Dateiname:  ";: FOR X = (A + O + 1) TO (A + O + 15):
     PRINT " $";: POKE 768, PEEK (X): CALL 769: NEXT
39Ø  PRINT : PRINT "Dateityp:     $";: POKE 768, PEEK (A + O + 16): CALL 769
4ØØ  PRINT : PRINT "Hauptzeiger: $";: POKE 768, PEEK (A + O + 18): CALL 769:
     POKE 768, PEEK (A + O + 17): CALL 769: PRINT "   ($HHLL)";
41Ø  PRINT : PRINT "Blockanzahl: $";: POKE 768, PEEK (A + O + 2Ø): CALL 769:
     POKE 768, PEEK (A + O + 19): CALL 769: PRINT "   ($HHLL)";
42Ø  PRINT : PRINT "Dateilänge:  $";: POKE 768, PEEK (A + O + 23): CALL 769:
     POKE 768, PEEK (A + O + 22): CALL 769: POKE 768, PEEK (A + O + 21):
     CALL 769: PRINT " ($HHMMLL)";
43Ø  PRINT : PRINT "Create-Dat:  $";: POKE 768, PEEK (A + O + 24): CALL 769:
     PRINT " $";: POKE 768, PEEK (A + O + 25): CALL 769
44Ø  PRINT : PRINT "Create-Uhr:  $";: POKE 768, PEEK (A + O + 26): CALL 769:
     PRINT " $";: POKE 768, PEEK (A + O + 27): CALL 769
45Ø  PRINT : PRINT "Version-Nr:  $";: POKE 768, PEEK (A + O + 28): CALL 769:
     PRINT " $";: POKE 768, PEEK (A + O + 29): CALL 769
46Ø  PRINT : PRINT "Zugriff:     $";: POKE 768, PEEK (A + O + 3Ø): CALL 769
47Ø  PRINT : PRINT "Zusatzinfo:  $";: POKE 768, PEEK (A + O + 32): CALL 769:
     POKE 768, PEEK (A + O + 31): CALL 769: PRINT "   ($HHLL)";
48Ø  PRINT : PRINT "Mod-Datum:   $";: POKE 768, PEEK (A + O + 33): CALL 769:
     PRINT " $";: POKE 768, PEEK (A + O + 34): CALL 769
49Ø  PRINT : PRINT "Mod-Uhr:     $";: POKE 768, PEEK (A + O + 35): CALL 769:
     PRINT " $";: POKE 768, PEEK (A + O + 36): CALL 769
5ØØ  PRINT : PRINT "Kopfzeiger:  $";: POKE 768, PEEK (A + O + 38): CALL 769:
     POKE 768, PEEK (A + O + 37): CALL 769: PRINT "   ($HHLL)";
51Ø  END
52Ø  REM Dateieintrag per Textfile suchen
53Ø  PRINT CHR$ (4);"PREFIX": INPUT P$
54Ø  PRINT CHR$ (4);"OPEN";P$;",TDIR"
55Ø  PRINT CHR$ (4);"READ";P$
56Ø  LL = LEN (S$)
57Ø  INPUT D$: INPUT K$: INPUT D$: REM Ignorieren!
58Ø  F = Ø: INPUT D$: IF MID$ (D$,2,LL) = S$ AND MID$ (D$,LL + 2,1) = " "
     THEN F = 1: GOTO 6ØØ: REM F=Such-Flag
```

EINTRAG.ANALYSE

Dieses umfangreiche, aber nützliche Programm erzeugt einen Hex-Dump aller Bestandteile eine Dateieintrages, und zwar auch solcher Bestandteile, die normalerweise nicht aus dem Catalog ersichtlich sind. Ferner können auch Dateieinträge von gelöschten Dateien angezeigt werden. Das Programm setzt theoretische Kenntnisse der Directory-Struktur voraus (s. „ProDOS für Aufsteiger", Bd. 1, S. 120ff). Ferner macht das Programm von der in einer späteren Folge erläuterten Möglichkeit Gebrauch, eine Datei (hier das Directory) mit dem BLOAD-B-Parameter schubweise einzulesen.

```
590  IF D$ < > "" THEN 580
600  PRINT CHR$ (4);"CLOSE";P$
610  PRINT P$;S$
620  IF F = 1 THEN PRINT : PRINT K$: PRINT D$
630  PRINT : RETURN


Beispiel: Programm "EINTRAG.ANALYSE" auf Datei "EINTRAG.ANALYSE"
          selbst angewandt (hier enthalten in Subdirectory "DIR")

/STIEHL/DIR/EINTRAG.ANALYSE

 NAME            TYPE   BLOCKS  MODIFIED         CREATED          ENDFILE SUBTYPE
 EINTRAG.ANALYSE BAS         6  13-JAN-85  0:00  13-JAN-85  0:00     2112

Speichertyp: $2
Namenslänge: $F
Dateiname:   $45 $49 $4E $54 $52 $41 $47 $2E $41 $4E $41 $4C $59 $53 $45
Dateityp:    $FC
Hauptzeiger: $00D5   ($HHLL)
Blockanzahl: $0006   ($HHLL)
Dateilänge:  $000840 ($HHMMLL)
Create-Dat:  $2D $AA
Create-Uhr:  $00 $00
Version-Nr:  $00 $00
Zugriff:     $E3
Zusatzinfo:  $0801   ($HHLL)
Mod-Datum:   $2D $AA
Mod-Uhr:     $00 $00
Kopfzeiger:  $00D3   ($HHLL)
```

Trace-Bug

Infolge diverser Pseudo-Trace-Fehler gerät das BASIC.SYSTEM oft ungewollt in den echten Trace-Modus (#-Zeilennummernanzeige während des laufenden Applesoft-Programms). Da zur Zeit noch verschiedene „Vorversionen" des BASIC .SYSTEMs in Umlauf gesetzt werden, ist es für einen definitiven Patch noch zu früh. Beispiele:

```
10 FOR X = 1 TO 60: REM ***PR#S-Bug
20 IF X = 20 THEN PR#0: PRINT „Trace-Anfang"
30 IF X = 40 THE POKE 242, 0: PRINT „Trace-Ende"
40 NEXT
```

```
10 FOR X = 1 TO 60: REM ***PR#S-FLASH-Bug
20 IF X = 20 THEN FLASH: PRINT „Trace-Anfang"
30 IF X = 40 THEN NORMAL: PRINT „Trace-Ende"
40 NEXT
```

PRODOS.READER

```
Simulierter CP/M-TYPE-Befehl für ProDOS
von U.Stiehl, 25.02.85

Liest ProDOS-Dateien beliebiger Art und Größe
schubweise in den RAM-Speicher $1000-$8FFF
und erzeugt einen ASCII-Dump am Bildschirm.
Speziell für sequentielle Textfiles gedacht.
Ctrl-Zeichen (außer Return) werden invers
dargestellt. Bei IIc/IIe auch mit 80-Zeichenkarte
möglich. Der Quellcode des Maschinenprogramms
befindet sich auf der Peeker-Sammeldisk.


100 REM PRODOS.READER von U.Stiehl/24.02.85
110 DATA 76,7,3,0,16,0,144,173,3,3
120 DATA 141,47,3,173,4,3,141,48,3,76
130 DATA 30,3,238,47,3,208,3,238,48,3
140 DATA 24,173,5,3,237,47,3,173,6,3
150 DATA 237,48,3,176,1,96,173,255,255,9
160 DATA 128,201,255,240,223,201,141,240,15,201
170 DATA 160,176,11,105,64,141,99,3,32,128
180 DATA 254,173,99,3,32,237,253,32,132,254
190 DATA 173,0,192,16,193,44,16,192,173,0
200 DATA 192,16,251,44,16,192,76,22,3
210 RESTORE : FOR X = 768 TO 866: READ Y: POKE X,Y: NEXT
220 TEXT : HOME : INVERSE : PRINT "PRODOS.READER":
    NORMAL : PRINT : PRINT : PRINT "START J/N ";: GET
    X$: PRINT X$: ON X$ = "J" OR X$ = "j" GOTO 230:
    ON X$ < > "N" AND X$ < > "n" GOTO 220: HOME : END
230 PRINT : PRINT CHR$ (4);"PREFIX": INPUT P$: PRINT
    CHR$ (4);"CAT";P$: PRINT : INPUT "DATEI:";S$: IF S$
    = "" THEN 220
240 REM Dateieintrag suchen
250 PRINT CHR$ (4);"OPEN";P$;",TDIR": PRINT CHR$
    (4);"READ";P$:L = LEN (S$)
260 INPUT D$: INPUT D$: INPUT D$: REM Ignorieren!
270 F = 0: INPUT D$: IF MID$ (D$,2,L) = S$ AND MID$
    (D$,L + 2,1) = " " THEN F = 1: GOTO 290:
    REM F=Such-Flag
280 IF D$ < > "" THEN 270
290 PRINT CHR$ (4);"CLOSE";P$
300 IF F = 0 THEN PRINT P$;S$;" FEHLT!": GET X$:
    GOTO 220
310 REM Lesepuffer $1000-$9000
320 T$ = "T" + MID$ (D$,18,3):LL = VAL ( MID$
    (D$,66,6))
330 HOME : PRINT CHR$ (4);"FRE": POKE 773,0: POKE
    774,144: REM $9000
340 IF LL < 32768 THEN R = LL: GOSUB 380:B = 0:L = R:
    GOSUB 390: GOTO 400
350 B = 0:L = 32768: GOSUB 390
360 B = B + L: IF B + L < LL THEN GOSUB 390: GOTO 360
370 R = LL - B: GOSUB 380:L = R: GOSUB 390: GOTO 400
380 POKE 773,R - INT (R / 256) * 256: POKE 774, INT (R
    / 256) + 16: RETURN
390 PRINT CHR$ (4)"BLOAD";P$;S$;",";T$;",A$1000";
    ",B";B;",L";L: CALL 768: RETURN
400 PRINT : GET X$: PRINT CHR$ (4): GOTO 220
```

BUCH-SHOP

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

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

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

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

Aus der Praxis heraus präsentiert der Autor Tips und Vorschläge, die das Programmieren leichter und zugleich effizienter machen. Alle Applesoft- und Integer-BASIC-Begriffe sind alphabetisch aufgelistet und werden eingehend erklärt. Dazu werden alle DOS-Befehle (neben vielen Begriffen der Computerterminologie) vorgestellt.
Beispielprogramme zeigen dem Nutzer, wie jeder Befehl funktioniert und helfen, die richtige Anwendung zu üben. Unter anderem lernt der Leser den besten Weg, um FOR/NEXT-Schleifen und IF/THEN-Entscheidungen für seine Zwecke einzusetzen.
Durch die präzise und leicht verständliche Sprache des Autors werden auch schwierige Befehle einfach in der Anwendung.

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

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

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

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

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

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

„Apple Assembler" wendet sich an alle, die bereits Anfängerkenntnisse der 6502-Programmierung haben – z. B. aufgrund des Buches „Apple Maschinensprache" – und nunmehr ein Nachschlagewerk für ihren Apple II Plus/IIe/IIc suchen, in dem alle wichtigen ROM-Routinen sowie eine Vielzahl sonstiger Hilfsprogramme in einer systematischen Form zusammengestellt werden. Insgesamt umfaßt dieses Buch über 40 Utilities, darunter mehrere völlig neuartige Programme wie Double-Lores, Double Hires, Screen-Format u. a.
Der erste Teil enthält ein Repetitorium der wichtigsten Befehle, Adressierungsarten und sonstigen Besonderheiten des 6502.
Im zweiten Teil werden alle Adressen des Monitors zusammengestellt, die für Assembler-Programmierer von Nutzen sein können. Darüber hinaus findet der Leser Unterroutinen für hexadezimale Addition/Subtraktion/Multiplikation/Division, Binär-Hex-ASCII-Umwandlung usw.
Der dritte Teil befaßt sich mit der Speicherverwaltung der Language Card und der IIe-64K-Karte und enthält Move-Programme zum Verschieben von Daten in die und aus der Language Card sowie der 64K-Karte.
Der vierte Teil ist dem Applesoft-ROM gewidmet und listet eine große Anzahl nützlicher Interpreter-Adressen. Bei den Utility-Programmen liegt das Schwergewicht auf Fließkommamathematik einschließlich Print Using.
Der letzte Teil behandelt den Text- und Graphikspeicher. Neben einem professionellen Maskengeneratorprogramm werden auch Routinen zur Double-Lores- und Double-Hires-Grafik vorgestellt.

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

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

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

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

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

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

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

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

Beachten Sie die Buch-Shop-Karte

Apples Maus lernt jetzt auch Pascal

von Jürgen Geiß

Apples neuestes Eingabegerät heißt Maus. Es ist ein flaches Kästchen mit einem breiten „Kopf“ (Knopf) auf dem „Rükken“ und einem grauen „Schwanz“ (Kabel), der mit dem Computer verbunden wird. Erstmals wurde die Maus bei der Lisa vorgestellt. Beim Macintosh ist dieses Gerät bereits ein unabdingbarer Bestandteil des Mikrocomputers, d.h. der Mac läßt sich ohne Maus überhaupt nicht bedienen. Apple-II-Besitzer, die die Maus einsetzen wollen, können nunmehr das sog. Maus-Paket erwerben. Allerdings läuft die Maus bislang nicht unter Pascal. Dieser Mangel wird durch die nachfolgende Pascal-Utility behoben.

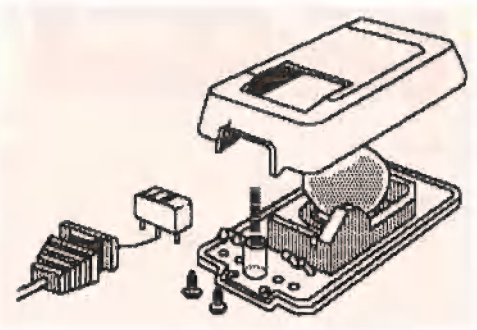
Das Maus-Paket umfaßt eine Interfacekarte, die in einen beliebigen Slot des Apple II, II+ oder IIe gesteckt werden kann, des weiteren die Maus, die mit der Karte zu verbinden ist, und eine Diskette mit dem Programm MousePaint, um die Benutzung der Maus zu demonstrieren. Für den Apple IIc ist keine Interfacekarte nötig; vielmehr wird hier das Kabelende einfach auf der Rückseite eingesteckt.

Allgemeines zur Pascal-Implementierung

Pascal bietet die Möglichkeit, Unterprogramme in einer Bibliothek abzulegen, die bei häufigem Benutzen einfach mittels einer „uses“-Deklaration im Programm erklärt werden. Diese Unterprogramme werden in der **SYSTEM.LIBRARY** zusammengefaßt. Will man Assemblerprogramme in die SYSTEM.LIBRARY einbinden, so muß auch eine Pascal-Hostdatei existieren, die mit dem Assemblerprogramm zu linken ist, ehe sie in die SYSTEM.LIBRARY eingebunden werden kann. Die Benutzung der Maus erfolgt durch die **Unit Mousestuff**, die aus einer Pascal-Hostdatei und einem Assemblerteil besteht.
Zuerst wird der Pascalteil eingegeben und übersetzt. Am besten nennt man die Pascal-Hostdatei **MOUSESTUFF.TEXT** und den Code entsprechend **MOUSESTUFF.-CODE**. Anschließend wird der Assemblerteil eingegeben und assembliert. Nennen wir den Assemblerteil **MOUSE.-ASS.TEXT** bzw. **MOUSE.ASS.CODE**. Nun wird die Pascal-Hostdatei mit dem Assemblerprogramm mittels des SYSTEM.LINKERs gelinkt. Die Anweisungen für das Linken entnehme man dem „Apple Pascal Operating System Reference Manual“. Schließlich wird der gelinkte Code noch in die SYSTEM.LIBRARY eingebunden. Dies geschieht mit dem LIBRARY-Programm, das sich auf der APPLE3: Systemdiskette befindet. Die Bedienung dieses Programmes ist wiederum dem Handbuch zu entnehmen.

Die Beschreibung der Mouse-Unit

Die Typen: Dem Benutzer der Mouse-Unit stehen zwei Pascal-Typen für die Programmierung der Maus zur Verfügung: **Mousemode** und **Mousestatus**.
Mousemode enthält folgende Komponenten:
Mouse_on: Soll die Maus eingeschaltet werden, so wird Mouse_on:= true gesetzt, andernfalls Mouse_on:= false.
Mouse_moved: Sollen Interrupts bei Bewegung der Maus erzeugt werden, so wird Mouse_moved:= true gesetzt, sonst false.
Button_pressed: Sollen Interrupts beim Drücken des Maus-Knopfes erzeugt werden, so muß Button_pressed:= true gesetzt werden, sonst false.
Screen_refresh: Sollen Interrupts jede 1/60 Sekunde erzeugt werden, so muß Screen_refresh:= true gesetzt werden, sonst false.
Reservd1 – Reservd4 sind reserviert für zukünftige Erweiterungen.
Nach dem Setzen der Komponenten von Mousemode muß die Prozedur Setmouse aufgerufen werden, die dann die Modi an das Mouse-Interface übergibt (siehe Beispielprogramme).
Mousestatus ist das „Button und Interrupt Status Byte“ und enthält folgende Komponenten:
Reservd1: siehe oben.
Mouse_moved: true, wenn ein Interrupt durch Mausbewegung erzeugt wurde.
Button_pressed: true, wenn ein Interrupt durch das Drücken des Maus-Knopfes erzeugt wurde.
Screen_refresh: true, wenn ein Interrupt erzeugt wurde, der mit Screen_refresh im



Mousemode gesetzt wurde (jede 1/60 Sekunde).
Reservd2: siehe oben.
X_Y_changed: true, wenn die Maus bewegt wurde und *nicht*, wie im Maus-Handbuch steht, wenn sich die Maus-Koordinaten geändert haben.
Button_dwn_lst: true, wenn der Knopf beim vorletzten Lesen der Mausdaten gedrückt war.
Button_down: true, wenn der Knopf gedrückt ist.

Die Prozeduren

Dem Benutzer stehen folgende Prozeduren zur Verfügung:
procedure setmouse (mode: mousemode): Diese Prozedur setzt die Maus in den Modus, der in der Variablen Mode angegeben wurde.
procedure servemouse (var status: mousestatus; var mouse_interrupted: boolean): Wenn ein Interrupt durch die Maus verursacht wurde, wird der Statusrecord neu gesetzt. Ferner kann abgefragt werden, ob die Maus den Interrupt verursacht hatte.
procedure readmouse (var x, y: integer;

var status: mousestatus; var mode: mousemode): Diese Prozedur liest die gesamten Maus-Daten in die angegebenen Variablen ein. Befindet sich die Maus im passiven Modus, so ist dies die einzige Prozedur, mit der die gesamten Maus-Daten gelesen werden können.

procedure posmouse (x, y: integer): Diese Prozedur setzt die Maus-Koordinaten auf die neuen X- und Y-Werte.

procedure clampmouse (x_min, x_max, y_min, y_max: integer): Der Wertebereich der Maus-Koordinaten kann durch diese Prozedur neu bestimmt werden. Wird clampmouse nicht aufgerufen, so gelten folgende Standardwerte: x_min = y_min:= 0, x_max = y_max:= 1023.

procedure homemouse: Durch diese Prozedur wird die Maus-Position auf die unterste Grenze gesetzt, die mit clampmouse gesetzt wurde. Wenn clampmouse nicht aufgerufen wurde, so werden die Maus-Koordinaten auf x = y:= 0 gesetzt.

procedure initmouse: Diese Prozedur muß vor allen anderen Prozeduren aufgerufen werden. Sie initialisiert die Maus und setzt interne Default-Werte für clampmouse. *Vorsicht:* Bei einem Apple II und Apple II+ überschreibt initmouse die Seite 1 der Hires-Grafik. Deshalb ist es erforderlich, diese Prozedur aufzurufen, bevor Grafiken erzeugt wurden.

function mouse_found: boolean Das Funktionsresultat ist true, wenn sich eine Maus im System befindet.

Nun können alle Mausprozeduren, die im Maus-Handbuch beschrieben sind, von Pascal aus mittels der uses-Mousestuff-Deklaration aufgerufen werden. So kann zum Beispiel leicht folgender Programmteil benutzt werden:

```
if mouse_found then initmouse usw.
```

Ist keine Maus an das System angeschlossen und werden trotzdem Maus-Prozeduren aufgerufen, so erfolgt keine Fehlermeldung und es werden auch keine ROM Routinen angesprungen, so daß das System auch korrekt weiterläuft, ohne sich „aufzuhängen".

Im Handbuch zur Maus werden einige Mausmodi angegeben, die mit **Interrupt** arbeiten. Da die Pascal-Versionen 1.0, 1.1, 1.2 keine Interrupts verarbeiten können, empfiehlt es sich nicht, die Maus in einem dieser Modi zu betreiben. Daher sollte man die Maus nur im **passiven** Modus wie folgt benutzen:

```
with Mode do
begin
  Mouse_on := true;
  Mouse_moved := false;
  Button_pressed := false;
  Screen_refresh := false
end;
```

Mit diesem Modus wird Setmouse aufgerufen. Die Maus befindet sich nun im passiven Modus und sendet keine Interrupts an das System. Dann können alle Mausprozeduren benutzt werden.

Beispielprogramme

Zwei Programme sollen den Umgang mit der Unit Mousestuff verdeutlichen.

Das erste Beispielprogramm, **Testmouse** genannt, druckt die X-und Y-Werte und den Status des Druckknopfes aus. Damit kann man die Bewegungen und das Drükken des Knopfes der Maus genau beobachten. Das Programm wird verlassen, wenn mit dem Knopf der Maus zweimal geklickt wurde, ohne daß die Maus bewegt wurde.

Das zweite Beispielprogramm, **Draw-Mouse** genannt, demonstriert, wie man mit Hilfe der Maus Grafiken erzeugen und bearbeiten kann.

Das Programm fragt zuerst nach einem Dateinamen des Bildes. Der Name darf nicht das Suffix „.GRAF" enthalten, da dieses automatisch hinzugefügt wird, und darf zudem nicht länger als 10 Buchstaben sein. Wird die Datei auf der Diskette gefunden, so wird das entsprechende Bild geladen.

Anschließend befindet man sich im Grafik-Bildschirm und hat einen **Bleistift** vor sich, mit dem gezeichnet werden kann, solange der Knopf der Maus gedrückt bleibt. Klickt man mit dem Knopf zweimal, ohne sich zu bewegen, hat man einen **Radiergummi** vor sich, mit dem man eventuelle Fehler wieder beseitigen kann. Auch der Radiergummi ist nur aktiv, solange der Knopf der Maus gedrückt bleibt. Zweimaliges Klicken wandelt den Radiergummi wieder in einen Bleistift um usw.

Wird die Tastatur benutzt, so werden die entsprechenden Zeichen an der Stelle, an der der Cursor (Bleistift oder Radiergummi) steht, auf dem Bildschirm ausgegeben. Damit kann man seine Schöpfung mit Texten versehen.

Das Programm kann durch Drücken der ESC-Taste verlassen werden, wobei das Bild auf Diskette gespeichert wird. Das Bild kann dann wieder beim erneuten Aufruf des Programmes geladen und weiterverarbeitet werden.

MOUSESTUFF

```
{$S+}
{$von Jürgen Geiß; September 1984}

unit mousestuff; intrinsic code 25;

interface

type mousemode   = packed record
                     Mouse_on       : boolean;
                     Mouse_moved    : boolean;
                     Button_pressed : boolean;
                     Screen_refresh : boolean;
                     Reservd1       : boolean;
                     Reservd2       : boolean;
                     Reservd3       : boolean;
                     Reservd4       : boolean;
                   end; {Mousemode}

     mousestatus = packed record
                     Reservd1       : boolean;
                     Mouse_moved    : boolean;
                     Button_pressed : boolean;
                     Screen_refresh : boolean;
                     Reservd2       : boolean;
                     X_Y_changed    : boolean;
                     Button_dwn_lst : boolean;
                     Button_down    : boolean;
                   end; {Mousestatus}

procedure setmouse (mode : mousemode);
procedure servemouse (var status : mousestatus;
                      var mouse_interrupted : boolean);
procedure readmouse (var x, y : integer;
                     var status : mousestatus;
                     var mode : mousemode);
procedure clearmouse;
procedure posmouse (x, y : integer);
procedure clampmouse (x_min, x_max, y_min, y_max : integer);
procedure homemouse;
procedure initmouse;

function  mouse_found : boolean;

implementation

{-------------------------------------------------------------}
procedure setmouse {mode : mousemode};
external;
procedure servemouse {var status : mousestatus;
                      var mouse_interrupted : boolean};
external;
procedure readmouse {var x, y : integer;
                     var status : mousestatus;
                     var mode : mousemode};
external;
procedure clearmouse;
external;
procedure posmouse {x, y : integer};
external;
procedure clampmouse {x_min, x_max, y_min, y_max : integer};
external;
procedure homemouse;
external;
procedure initmouse;
external;
function mouse_found {: boolean};
external;

procedure searchmouse;
external;

begin {mousestuff}
  searchmouse
end {mousestuff}.
```

Beispielprogramm 1: TESTMOUSE

```
program Testmouse (output);

uses Mousestuff;

var I          : integer;
    Mode       : mousemode;
    Status     : mousestatus;
    X, Y       : integer;
    ClickCount : integer;

begin
  initmouse;
  writeln ('Maustest: ');
  fillchar (Mode, size_of (Mode), Ø);
  Mode.Mouse_on := true;
  setmouse (Mode);
  clearmouse;
  readmouse (X, Y, Status, Mode);


  with Status do
  begin
    gotoxy (Ø, 1Ø);
    writeln ('X: ', X : 16);
    writeln ('Y: ', Y : 16);
    writeln ('X_Y_changed:      ', ord (X_Y_changed));
    writeln ('Button_down_last: ', ord (Button_dwn_lst));
    writeln ('Button_down:      ', ord (Button_down))
  end; {with}

  ClickCount := Ø;

  repeat
    with Status do
    begin
      readmouse (X, Y, Status, Mode);
      if X_Y_changed
      then ClickCount := Ø
      else
        if not Button_down and Button_dwn_lst
        then ClickCount := ClickCount + 1;

      if X_Y_changed or Button_down then
      begin
        gotoxy (15, 1Ø);
        write (X : 4);
        gotoxy (15, 11);
        write (Y : 4);
        gotoxy (18, 12);
        write (ord (X_Y_changed));
        gotoxy (18, 13);
        write (ord (Button_dwn_lst));
        gotoxy (18, 14);
        write (ord (Button_down))
      end {if}
    end {with}
  until ClickCount = 2;

  fillchar (Mode, size_of (Mode), Ø);
  setmouse (Mode) {Schalte Maus ab}
end.
```

Beispielprogramm 2: DRAWMOUSE

```
{$I-}
{$R-}
program DrawMouse (input, output);

uses Turtlegraphics, Applestuff, Mousestuff;

const Hiresblks    =    16;
      Xdots        =   28Ø;
      Ydots        =   192;
      Xmax         =   279;
      Ymax         =   191;
      AdrPage1     =  8192;

      PencilX      =     8;
      PencilY      =    14;
      EraserX      =    12;
      EraserY      =     8;

      BEL          =     7;
      ESC          =    27;
```

```
type  Penciltype   = packed array [1..PencilY, 1..PencilX]
                     of boolean;
      Erasertype   = packed array [1..EraserY, 1..EraserX]
                     of boolean;
      Trickarray   = packed array [Ø..Ø] of Ø..255;
      RWtype       = (DoRead, DoWrite);
      XY_type      = record
                       X : integer;
                       Y : integer
                     end; {XYtype}

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

var   Cursor       : XY_type;
      LastCursor   : XY_type;
      Memory       : ↑Trickarray;
      Moved        : boolean;
      Key          : char;
      Keyhit       : boolean;
      Button_down  : boolean;
      Last_down    : boolean;
      ClickCount   : integer;
      Command      : integer;

      Pencil       : Penciltype;
      Eraser       : Erasertype;

      Graphfile    : file;
      Graphname    : string;
      Mode         : Mousemode;
      Status       : Mousestatus;
      I            : integer;

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

procedure Init;

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

  procedure InitPencil;

  {erzeugt einen Bleistift}

  var Row : integer;

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

    procedure  CreateLine (S : string);

    var I : integer;

    begin {Createline}
      for I := 1 to length (S)
      do Pencil [Row, I] := S [I] <> ' ';
      Row := Row - 1
    end; {Createline}

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

  begin {InitPencil}
    Row := PencilY;
    Createline ('    *** ');
    Createline ('   *  **');
    Createline ('   **  *');
    Createline ('  *  ***');
    Createline ('  *    * ');
    Createline (' *     * ');
    Createline (' *    *  ');
    Createline ('*     *  ');
    Createline ('*    *   ');
    Createline ('**  *    ');
    Createline ('****     ');
    Createline ('***      ');
    Createline ('**       ');
    Createline ('         ')
  end; {InitPencil}

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

  procedure InitEraser;

  {erzeugt einen Radiergummi}

  var Row : integer;
```

```
    procedure  CreateLine (S : string);

    var I : integer;

    begin {Createline}
      for I := 1 to length (S)
      do Eraser [Row, I] := S [I] <> ' ';
      Row := Row - 1
    end; {Createline}

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

  begin {InitEraser}
    Row := EraserY;
    Createline ('     *******');
    Createline ('    *      *');
    Createline ('   *      **');
    Createline ('  *      ***');
    Createline (' *      *** ');
    Createline ('**********  ');
    Createline ('*      **   ');
    Createline ('********    ')
  end; {InitEraser}

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

begin {Init}
  Cursor.X         := 14Ø;
  Cursor.Y         := 92;
  LastCursor       := Cursor;
  Memory           := nil;
  Command          := 1;

  Moved            := false;
  Keyhit           := false;
  Key              := chr (Ø);
  Button_down      := false;
  Last_down        := Button_down;

  if Mouse_found
  then
    begin
      Initmouse;
      fillchar (Mode, size_of (Mode), Ø);
      Mode.Mouse_on := true;
      Setmouse (Mode); {setzte Maus auf passiven Modus
                        (keine Interrupts)}
      Clearmouse;
      Clampmouse (Ø, 2 * Xdots - 1, Ø, 2 * Ydots - 1);
      {2 * Xdots, 2 * Ydots; damit werden
       feinere Bewegungen möglich}
      with Cursor do posmouse (X * 2, (Ymax - Y) * 2)
    end {then}
  else
    begin
      writeln
      (chr (BEL), 'Fehler: Keine Maus angeschlossen!');
      exit (program)
    end; {else}

  InitPencil;
  InitEraser;

  initturtle;
  fillscreen (white);
  with Cursor do moveto (X, Y);
  textmode
end; {Init}

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

procedure RW_Graph (ReadWrite : RWtype);

var Blocks : integer;

begin {RW_Graph}
  case ReadWrite of
    DoRead  : Blocks := blockread
              (Graphfile, Memory↑ [AdrPage1], Hiresblks, Ø);
    DoWrite : Blocks := blockwrite
              (Graphfile, Memory↑ [AdrPage1], Hiresblks, Ø)
  end; {case}

  if Blocks <> Hiresblks then
  begin
    writeln
    (chr (BEL), 'Fehler beim Lesen/Schreiben der Grafik!');
    exit (program)
  end {if}
end; {RW_Graph}
```

peeker Börse

Gelegenheitsanzeigen

Sie können unter dieser Rubrik zu einem besonders günstigen Preis

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

Musteranzeige privat

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

Beispiel:

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

nur DM 17,10 inkl. MwSt.

Musteranzeige gewerblich

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

Beispiel:

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

nur DM 137,94 inkl. MwSt.

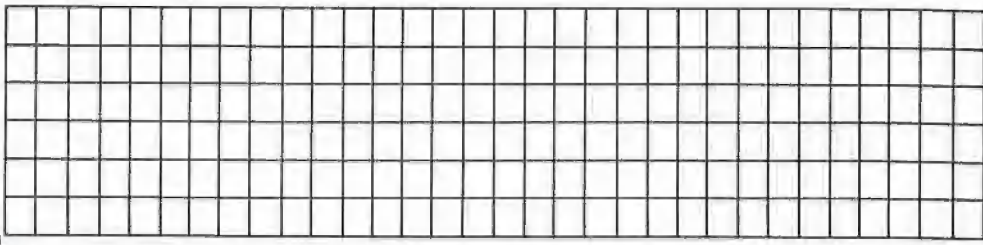
peeker MAGAZIN FÜR APPLE-COMPUTER

peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

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

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



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

peeker MAGAZIN FÜR APPLE-COMPUTER

Produkt-Karte

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

über _____

_____

bitte ich um detaillierte Information.

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

Menge	Produkt und Bestellnummer	à DM	gesamt DM

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

_____

Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

peeker MAGAZIN FÜR APPLE-COMPUTER

Info-Karte

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

_____

_____

_____

_____

_____

_____

peeker-Börse

Vorname, Name

Beruf

Straße

Wohnort

PLZ

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

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

Datum Unterschrift

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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

Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

ANTWORTKARTE

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

POSTKARTE

Firma

Straße

PLZ/Ort

POSTKARTE

peeker
Redaktion
Postfach 10 28 69
6900 Heidelberg 1

Produkt-Karte

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

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

Vorher aber nicht vergessen:
kreuzen Sie an, welchen Informationswunsch Sie haben.

Damit erleichtern Sie dem Hersteller eine gezielte Beantwortung Ihrer Anfrage.

Zum Schluß tragen Sie auf der Rückseite die genaue Anschrift des Inserenten/Herstellers und Ihre vollständige Firmenanschrift ein.

PEEKER
MAGAZIN FÜR APPLE-COMPUTER

```
procedure AskFile;

begin {AskFile}
  repeat
    write
    ('Name der Grafik-Datei (ohne das Suffix "GRAF"): ');
    readln (Graphname);
    if Graphname = '' then Graphname := '*SYSTEM.WRK';
    Graphname := concat (Graphname, '.GRAF');
    reset (Graphfile, Graphname);

    if IOresult = 0
    then
      begin
        writeln;
        writeln ('Lese ', Graphname);
        grafmode;
        RW_Graph (DoRead);
        textmode
      end {then}
    else rewrite (Graphfile, Graphname)
  until IOresult = 0
end; {AskFile}

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

procedure GetEvent;

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

  procedure Testmouse;

  {liest die Mausdaten}

  begin {Testmouse}
    with Cursor do
    begin
      Readmouse (X, Y, Status, Mode);

      X := X div 2;
      Y := Y div 2;
      Y := Ymax - Y
    end {with}
  end; {Testmouse}

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

begin {GetEvent}
  LastCursor := Cursor;
  Last_down  := Button_down;
  Keyhit     := keypress;
  if Keyhit then read (keyboard, Key);

  Testmouse;
  Button_down := Status.Button_down;
  with Cursor do Moved := (X <> LastCursor.X) or
   (Y <> LastCursor.Y);

  if Moved
  then ClickCount := 0
  else if Last_down and not Button_down then
  ClickCount := ClickCount + 1
end; {GetEvent}

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

procedure Draw;

var  Color : screencolor;

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

  procedure PencilDraw (Position : XY_type);

  begin {PencilDraw}
    with Position do
    if (X in [0..Xmax]) and
       (Y in [0..Ymax]) then
       drawblock
       (Pencil, 2, 0, 0, PencilX, PencilY, X, Y, 6)
  end; {PencilDraw}

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

procedure DrawPencil;

begin {DrawPencil}
  PencilDraw (Cursor)
end; {DrawPencil}
```

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

  procedure ErasePencil;

  begin {ErasePencil}
    PencilDraw (LastCursor)
  end; {ErasePencil}

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

  procedure EraserDraw (Position : XY_type; Mode : integer);

  begin {EraserDraw}
    with Position do
    if (X in [0..Xmax]) and
       (Y in [0..Ymax]) then
       drawblock
       (Eraser, 2, 0, 0, EraserX, EraserY, X, Y, Mode)
  end; {EraserDraw}

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

  procedure DrawEraser;

  begin {DrawEraser}
    EraserDraw (Cursor, 6)
  end; {DrawEraser}

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

  procedure EraseEraser;

  begin {EraseEraser}
    EraserDraw (LastCursor, 6)
  end; {EraseEraser}

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

  procedure DrawCursor;

  begin {DrawCursor}
    case Command of
       1 : DrawPencil;
       2 : DrawEraser
    end {case}
  end; {DrawCursor}

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

  procedure EraseCursor;

  begin {EraseCursor}
    case Command of
       1 : ErasePencil;
       2 : EraseEraser
    end {case}
  end; {EraseCursor}

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


  procedure HandleMove;

  var Mode : integer;

  begin {HandleMove}
    case Command of
      1 : with Cursor do moveto (X, Y);
      2 : if Button_down then
          begin
            Mode := 15;
            EraserDraw (Cursor, Mode)
          end {if}
    end {case}
  end; {HandleMove}

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

  procedure HandleKeys;

  begin {HandleKeys}
    EraseCursor;
    wchar (Key);
    DrawCursor
  end; {HandleKeys}

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

```
procedure Init;

begin {Init}
  Color := none;
  pencolor (Color);
  chartype (5);
  with Cursor do moveto (X, Y);
  DrawCursor
end; {Init}
{-------------------------------------------------------------}
begin {Draw}
  Init;

  repeat
    GetEvent;
    if Clickcount = 2 then
    begin
      ClickCount := 0;
      EraseCursor;
      Command := Command + 1;
      if Command = 3 then Command := 1;
      DrawCursor
    end; {if}

    if Moved then EraseCursor;
    if Keyhit then HandleKeys;

    if Button_down
    then Color := black
    else Color := none;
    pencolor (Color);

    if Moved then
    begin
      HandleMove;
      DrawCursor
    end {if}
  until Keyhit and (Key = chr (ESC));

  LastCursor := Cursor;
  EraseCursor;
end; {Draw}
{-------------------------------------------------------------}
begin
  page (output);
  Init;
  AskFile;
  Grafmode;
  Draw;
  textmode;
  writeln;
  write ('Grafik wird abgespeichert...');
  RW_Graph_(DoWrite);
  close (Graphfile, lock)
end.
```

MAUS.ASS

```
;*************************************************************
;*                                                           *
;*          Externe Prozeduren für Unit MOUSESTUFF           *
;*                                                           *
;*                  APPLE PASCAL [1.0], [1.1], [1.2]         *
;*                                                           *
;* type mousemode   = packed record                          *
;*                      Mouse_on       : boolean;            *
;*                      Mouse_moved    : boolean;            *
;*                      Button_pressed : boolean;            *
;*                      Screen_refresh : boolean;            *
;*                      Reservdl       : boolean;            *
;*                      Reservd2       : boolean;            *
;*                      Reservd3       : boolean;            *
;*                      Reservd4       : boolean             *
;*                    end; {mousemode}                       *
;*                                                           *
;*      mousestatus = packed record                          *
;*                      Reservdl       : boolean;            *
;*                      Mouse_moved    : boolean;            *
;*                      Button_pressed : boolean;            *
;*                      Screen_refresh : boolean;            *
;*                      Reservd2       : boolean;            *
;*                      X_Y_changed    : boolean;            *
;*                      Button_dwn_lst : boolean;            *
;*                      Button_down    : boolean             *
;*                    end; {mousestatus}                     *
;*                                                           *
;* procedure setmouse (mode : mousemode);                    *
;* procedure servemouse (var status : mousestatus;           *
;*           mouse_interrupted : boolean)                    *
;* procedure readmouse (var x, y : integer;                  *
;*           var status : mousestatus;                       *
;*           var mode : mousemode);                          *
;* procedure clearmouse;                                     *
;* procedure posmouse (x, y : integer);                      *
;* procedure clampmouse (x_min, x_max, y_min, y_max :        *
;*                       integer);                           *
;* procedure homemouse;                                      *
;* procedure initmouse;                                      *
;* function  mouse_found : boolean;                          *
;*                                                           *
;*************************************************************
```

Fortsetzung S. 57

Wie man das Maus-Modul implementiert:

Schritt 1: MOUSESTUFF und MOUSE.ASS als Pascal-Textfile eingeben und abspeichern oder von Peeker-Sammeldisk mit GETDOS konvertieren

Schritt 2: MOUSESTUFF compilieren über C-Kommando unter MOUSESTUFF.CODE.

Schritt 3: MOUSE.ASS assemblieren über A-Kommando unter MOUSE.ASS.CODE.

Schritt 4: Linker über L-Kommando aufrufen:

Schritt 4a: Auf Frage „Host-File?“ mit MOUSESTUFF antworten.

Schritt 4b: Auf Frage „Lib-File?“ mit MOUSE.ASS antworten.

Schritt 4c: Mit Return Linker beenden.

Schritt 4d: Auf Frage „Map-File“ auch mit Return antworten.

Schritt 4e: Auf Frage „Output-File“ mit MOUSE antworten. Somit entsteht die fertige Library unter dem Namen „MOUSE.CODE“.

Schritt 5: LIBRARY-Programm von APPLE3-Diskette über X-Kommando aufrufen:

Schritt 5a: Auf Frage „Output-Code-File“ mit „*“ antworten.

Schritt 5b: Auf Frage „Link-Code-File“ mit „*“ antworten.

Schritt 5c: Nunmehr mit „=“-Kommando alle Slots kopieren.

Schritt 5d: Nun N-Kommando eingeben und wie in Schritt 5b mit „MOUSE“ antworten.

Schritt 5e: „1“ Return und danach z.B. „6“ Return (= muß ein noch freier Slot sein) eingeben.

Schritt 5f: Mit Q-Kommando LIBRARY-Programm verlassen

```
                .NOPATCHLIST

RETURN          .EQU ØØ
CNØØ            .EQU Ø2

;Maus-Konstanten

SET             .EQU 12
SERVE           .EQU 13
READ            .EQU 14
CLEAR           .EQU 15
POS             .EQU 16
CLAMP           .EQU 17
HOME            .EQU 18
INIT            .EQU 19

X_LOW           .EQU 478
Y_LOW           .EQU 4F8
X_HIGH          .EQU 578
Y_HIGH          .EQU 5F8
RESERVD1        .EQU 678
RESERVD2        .EQU 6F8
STATUS          .EQU 778
MODE            .EQU 7F8

;************************************

                .MACRO PULL

                PLA
                STA %1
                PLA
                STA %1+1

                .ENDM

;************************************

                .MACRO PUSH

                LDA %1+1
                PHA
                LDA %1
                PHA

                .ENDM

;************************************

                .PROC SET_CN_NØ

                .DEF SLOT

                PHA
                LDA SLOT
                ORA #ØCØ
                TAX
                ASL A
                ASL A
                ASL A
                ASL A
                TAY
                PLA
                RTS

SLOT            .WORD Ø

;************************************

                .PROC CALLMOUSE

                .REF SLOT,SET_CN_NØ

                PHA
                LDA #ØØ
                STA CNØØ
                LDA SLOT
                ORA #ØCØ
                STA CNØØ+1
                LDA (CNØØ),Y
                JSR SET_CN_NØ
                STA INDIRECT
                STX INDIRECT+1
                PLA
                JMP INDIRECT

INDIRECT        .WORD Ø
```

```
;************************************

                .PROC SETMOUSE,1

                .REF CALLMOUSE,SLOT

MODE_ADR        .EQU 1Ø

                PULL RETURN

;hole Adresse des Modus
                PLA
                STA MODE_ADR
                PLA
                STA MODE_ADR+1

                PUSH RETURN

                LDA SLOT
                BEQ $1

                LDY #ØØ
                LDA (MODE_ADR),Y
                LDY #SET
                JSR CALLMOUSE

$1              RTS

;************************************

                .PROC SERVEMOUSE,2

                .REF CALLMOUSE,SLOT

STATUSADR       .EQU 1Ø
MOUSE_INTERADR  .EQU 12

                PULL RETURN

                PLA
                STA MOUSE_INTERADR
                PLA
                STA MOUSE_INTERADR+1

                PLA
                STA STATUSADR
                PLA
                STA STATUSADR+1

                PUSH RETURN

                LDA SLOT
                BEQ $1

                LDY #SERVE
                JSR CALLMOUSE
;Hole Carry Bit in Akku als Bit Ø
                ROL A
                AND #Ø1

                LDY #ØØ
                STA (MOUSE_INTERADR),Y
                INY
                LDA #ØØ
;Bit 7 immer Ø
                STA (MOUSE_INTERADR),Y

                LDY #ØØ
                LDX SLOT
                LDA STATUS,X
                STA (STATUSADR),Y

$1              RTS

;************************************

                .PROC READMOUSE,4

                .REF CALLMOUSE,SLOT

X_ADR           .EQU 1Ø
Y_ADR           .EQU 12
STATUS_ADR      .EQU 14
MODE_ADR        .EQU 16

                PULL RETURN
```

```
                PLA
                STA MODEADR
                PLA
                STA MODEADR+1

                PLA
                STA STATUS_ADR
                PLA
                STA STATUS_ADR+1

                PLA
                STA Y_ADR
                PLA
                STA Y_ADR+1

                PLA
                STA X_ADR
                PLA
                STA X_ADR+1

                PUSH RETURN

                LDA SLOT
                BEQ $1

                LDY #READ
                JSR CALLMOUSE
                LDX SLOT

                LDY #ØØ
                LDA X_LOW,X
                STA (X_ADR),Y
                INY
                LDA X_HIGH,X
                STA (X_ADR),Y

                LDY #ØØ
                LDA Y_LOW,X
                STA (Y_ADR),Y
                INY
                LDA Y_HIGH,X
                STA (Y_ADR),Y

                LDY #ØØ
                LDA STATUS,X
                STA (STATUS_ADR),Y

                LDA MODE,X
                STA (MODE_ADR),Y

$1              RTS

;************************************

                .PROC CLEARMOUSE

                .REF CALLMOUSE,SLOT

                LDA SLOT
                BEQ $1

                LDY #CLEAR
                JSR CALLMOUSE

$1              RTS

;************************************

                .PROC POSMOUSE,2

                .REF CALLMOUSE,SLOT

                PULL RETURN
                LDX SLOT

                PLA
                STA Y_LOW,X
                PLA
                STA Y_HIGH,X

                PLA
                STA X_LOW,X
                PLA
                STA X_HIGH,X
```

```
                PUSH RETURN

                LDA SLOT
                BEQ $1

                LDY #POS
                JSR CALLMOUSE

$1              RTS

;***************************************

                .PROC CLAMPMOUSE,4

                .REF CALLMOUSE,SLOT

XY_MIN_LOW      .EQU 478
XY_MAX_LOW      .EQU 4F8
XY_MIN_HIGH     .EQU 578
XY_MAX_HIGH     .EQU 5F8

                PULL RETURN

                PLA
                STA XY_MAX_LOW
                PLA
                STA XY_MAX_HIGH
                PLA
                STA XY_MIN_LOW
                PLA
                STA XY_MIN_HIGH

                LDA SLOT
                BEQ $1

;Setze neue Grenzen für Y-Koordinate
                LDA #Ø1
                LDY #CLAMP
                JSR CALLMOUSE

$1              PLA
                STA XY_MAX_LOW
                PLA
                STA XY_MAX_HIGH
                PLA
                STA XY_MIN_LOW
                PLA
                STA XY_MIN_HIGH

                PUSH RETURN
```

```
                LDA SLOT
                BEQ $2

;Setze neue Grenzen für X-Koordinate
                LDA #ØØ
                LDY #CLAMP
                JSR CALLMOUSE

$2              RTS

;***************************************

                .PROC HOMEMOUSE

                .REF CALLMOUSE,SLOT

                LDA SLOT
                BEQ $1

                LDY #HOME
                JSR CALLMOUSE

$1              RTS

;***************************************

                .PROC INITMOUSE

                .REF CALLMOUSE,SLOT

                LDA SLOT
                BEQ $1

                LDY #INIT
                JSR CALLMOUSE

$1              RTS

;***************************************

                .FUNC MOUSE_FOUND

                .REF SLOT

                PULL RETURN

                PLA
                PLA
                PLA
                PLA
```

```
;Das höchste Byte einer Booleschen
;Variablen ist immer Ø
                LDA #ØØ
                PHA
                LDA SLOT
;Keine Maus gefunden, dann
                BEQ $1
                LDA #Ø1         ;true

$1              PHA             ;false

                PUSH RETURN
                RTS

;***************************************

                .PROC SEARCHMOUSE

                .REF SLOT

TEMP            .EQU ØØ

                LDX #Ø8
                LDA #ØØ
                STA SLOT
                STA TEMP
                LDA #ØC8
                STA TEMP+1

$1              DEC TEMP+1
                DEX
                BEQ EXIT
                LDY #ØC
                LDA (TEMP),Y
                CMP #2Ø
                BNE $1
                LDY #ØFB
                LDA (TEMP),Y
                CMP #ØD6
                BNE $1
                STX SLOT
EXIT            RTS

;***************************************

                .END ;mousestuff
```

CP/M für Einsteiger

von Jörg Lange

Dieser Artikel soll Grundkenntnisse über die Benutzung des Betriebssystems CP/M 2.2/2.23 vermitteln. Er richtet sich nicht nur an diejenigen, die ihre ersten Schritte in CP/M machen wollen, sondern auch an die Anwender kommerzieller Programme wie WORDSTAR und DBASE II oder der CP/M-Dienstprogramme, z.B. zur Herstellung von Sicherheitskopien.

CP/M wurde 1973 als herstellerunabhängiges Betriebssystem entwickelt. Sein hohes Alter ist auf der einen Seite der Grund für einen erheblichen Mangel an Komfort, andererseits steht dem Anwender heute ein großes Angebot an Software zur Verfügung, das auf den Rechnern unterschiedlichster Hersteller läuft – vorausgesetzt, sie enthalten einen Z80-Prozessor. Um CP/M auf dem Apple zu benutzen, ist daher die Anschaffung der sog. „Z80-Softcard" erforderlich.

Installierung der Z80-Softcard

Bei der Installierung dieser Zusatzkarte sollten Sie gleich die Belegung der anderen Slots Ihres Apples überprüfen. CP/M erwartet folgende Zuordnung:
Disk-Controller: Unbedingt in Slot 6, weitere Laufwerke können an Slot 5 und 4 angeschlossen werden.
Drucker-Interface: Grundsätzlich in Slot 1, eventuell auch 2.
Videokarte (80 Z/Z): Slot 3
Die Z80-Softcard kann an beliebiger Stelle eingesteckt werden, allerdings wird Slot 4 empfohlen. Eine Ausnahme bildet lediglich der Steckplatz 0, eine dort befindliche 16K-Karte wird von CP/M – je nach gewählter Version – mitbenutzt.

Erste Annäherung

Starten von CP/M 2.2

Zum besseren Verständnis der nun folgenden Beschreibung des Betriebssystems sollten Sie jetzt an Ihrem Rechner Platz nehmen und CP/M starten. Dazu müssen Sie lediglich Ihre CP/M-Masterdisk in das Bootlaufwerk (Slot 6, Drive 1) einlegen und den Computer einschalten.

Ein wichtiger Hinweis: Es gibt (mindestens) drei Varianten der aktuellen CP/M-Version 2.2, von denen nur eine (CP/M 2.2 44K) auf Geräten mit 48K RAM läuft. CP/M 2.2 56K /60K benötigen, wie es schon die Bezeichnung vermuten läßt, einen vollausgebauten 64K-Speicher, was zur Arbeit mit kommerziellen Programmen aber auch dringend empfohlen werden muß.

Nach dem Booten meldet sich CP/M unter Angabe der genauen Versionsbezeichnung (z.B. CP/M 2.23 60K ...) mit dem sog. *Prompt* „A>". CP/M erwartet jetzt von Ihnen die Eingabe einer Anweisung; der Buchstabe im Prompt gibt dabei das Bezugslaufwerk an (Wir kommen gleich noch darauf zurück, was es damit auf sich hat).
Unter CP/M werden übrigens alle Laufwerke mit Buchstaben bezeichnet; es gilt dabei folgende Laufwerkszuordnung:
A: Slot 6, Drive 1
B: Slot 6, Drive 2
C: Slot 5, Drive 1
D: Slot 5, Drive 2

DIR – Ausgabe des Directory

Tippen Sie jetzt einmal den Befehl „DIR" ein. Diese Anweisung muß wie fast alle CP/M-Kommandos mit Betätigung der <RETURN>-Taste abgeschlossen werden. Der Rechner gibt Ihnen nun das Inhaltsverzeichnis (englisch: DIRectory), und zwar das vom Drive A:, dem derzeitigen Bezugslaufwerk, aus.
CP/M wird alle Ihre Befehle auf dieses Laufwerk beziehen, solange diese keine Laufwerksangabe (Kennbuchstabe und „:") enthalten. Wollen Sie also das DIRectory einer Disk im Drive B: betrachten, müssen Sie „DIR B:" benutzen.
Es gibt noch eine Alternative: Sie können dem Rechner mitteilen, daß ein anderes Bezugslaufwerk verwendet werden soll. Geben Sie einfach den entsprechenden Laufwerkskenner ein, beispielsweise „B:<RETURN>". Es erscheint als Prompt „B>". Der Befehl „DIR <RETURN>" zeigt jetzt den Inhalt der Disk B: an. Mit „A:<RETURN>" gelangen Sie wieder in die Ausgangsstellung zurück. (Falls Sie noch keine zweite CP/M-Disk haben, sollten Sie das hier Gesagte noch nicht ausprobieren, da der Versuch, auf ein Laufwerk zu wechseln, das keine lesbare Disk enthält, i.d.R. zum „Absturz" des Betriebssystems führt. Sie müssen dann den Rechner abschalten!)

Dateien und Dateinamen

Zunächst wollen wir uns noch einmal näher mit dem DIRectory befassen: Dieses Inhaltsverzeichnis gibt die Namen aller auf der Diskette befindlichen Dateien (engl.: Files) aus. Wie Sie selber schnell feststellen können, sind diese Filenamen alle ähnlich aufgebaut: ein bis zu acht Buchstaben umfassender eigentlicher Name, gefolgt von einem „." und einer sog. „Filetype"-Angabe, die im Regelfall grundsätzliche Aussagen über den Inhalt einer Datei zuläßt. So kennzeichnet z.B. „.BAS" ein Basic-Programm, „.TXT" einen gespeicherten Text und „.COM" einen *Command-File*. Was verbirgt sich nun hinter dieser letztgenannten Bezeichnung?
Ein Command-File enthält ein ausführbares Programm, das von CP/M aus gestartet werden kann. So gibt es z.B. auf der CP/M-Masterdisk „COPY.COM", eine Utility (Hilfsprogramm) zum Kopieren ganzer Disketten. Es wird insbesondere benötigt, um Sicherheitskopien von Datendisks oder Programmen anzufertigen, was übrigens nicht nur für kommerzielle Computerbenutzer ratsam ist.

COPY – Kopieren ganzer Disketten

Der Befehl zum Starten eines solchen Programms hat im wesentlichen die Form „Laufwerk:Programmname ...weitere Angaben...". Betrachten wir nun – um konkret zu werden – den Fall, daß Sie eine Kopie (engl.: Backup) ihrer CP/M-Masterdisk, die jetzt in Drive A: liegt (davon sind wir ja ausgegangen) machen wollen. Legen Sie eine leere Diskette in Drive B: ein und geben Sie „COPY B:=A: <RETURN>" ein.
(Falls Sie als Bezugslaufwerk Disk B: gewählt haben – Sie erkennen das am Prompt „B>" – müssen Sie das Kopierprogramm mit „A:COPY B:=A: <RETURN>" starten, um dem Rechner mitzuteilen, daß er „COPY.COM" im Laufwerk A: findet. Tun Sie das nicht, wird Sie CP/M

mit der Meldung „COPY?“ darauf hinweisen, daß es den entsprechenden File nicht gefunden hat.)
Das Kopierprogramm wird sich dann mit einer Überschrift (APPLE II CP/M/16 SECTOR DISK COPY PROGRAM) melden und Sie auffordern, das Original (engl.: MASTER) in Laufwerk A: und die Zieldiskette (SLAVE) in Drive B: einzulegen. Vergewissern Sie sich nochmals, ob Original und Duplikat in den richtigen Laufwerken liegen und drücken Sie <RETURN> zum Starten des Kopiervorganges. Sobald dieser beendet ist, werden Sie gefragt, ob Sie eine weitere Kopie anfertigen wollen. Bei negativer Antwort fordert Sie der Rechner zum Einlegen der CP/M Masterdisk in Drive A: auf – in unserem speziellen Fall sollte das ja erfüllt sein.
Einige Bemerkungen sind zum Kopiervorgang noch zu machen:
Der allgemeine Befehl zum Starten des Kopierprogramms lautet „COPY Ziellaufwerk:=Originallaufwerk: /Parameter“. Sie können dabei auch Kopien unter Benutzung nur eines Laufwerks anfertigen (z.B. „COPY A:=A:“). Das Programm fordert Sie dann entsprechend zum Diskettenwechseln auf. Eine weitere Form dieses Kommandos ermöglicht es Ihnen, nur das Betriebssystem CP/M zu kopieren: „COPY B:=A: /S“. Wir werden später darauf zurückkommen.

FORMAT – Formatieren von Disketten
Es gibt Versionen des COPY-Programms, die nicht in der Lage sind, auf unbenutzte Disketten zu kopieren. Diese müssen dann vorher erst entsprechend hergerichtet (formatiert) werden.
Dafür ist die FORMAT-Utility zuständig. Dieses Programm befindet sich ebenfalls auf der CP/M-Masterdisk und wird mit „FORMAT Laufwerk: <RETURN>“ gestartet. Legen Sie also z.B. eine neue Diskette in Drive B: ein und tippen Sie „FORMAT B: <RETURN>“. Überprüfen Sie, ob die zu formatierende Disk tatsächlich der Aufforderung des Programms entsprechend eingelegt wurde, und betätigen Sie die <RETURN>-Taste zum Start des Formatiervorganges. Nach dessen Beendigung haben Sie eine leere Diskette zur Verfügung, auf die Sie jetzt Dateien kopieren können. Überzeugen Sie sich selbst („DIR B:“, s.o.)! Selbstverständlich kann die leere Disk auch zur Datenspeicherung für Programme wie WORDSTAR oder DBASE benutzt werden. Wollen Sie sie allerdings als Bootdisk zum Starten des CP/M-Systems verwenden, müssen Sie noch mit „COPY B:=A:/S <RETURN>“ das Betriebssystem auf Ihre neue Disk kopieren.

Das COPY- sowie das FORMAT-Programm benutzen folgende Fehlermeldungen:
DISK WRITE PROTECTED: Die Schreibschutzkerbe auf der rechten Diskettenseite ist überklebt.
DISK I/O-ERROR: Die Diskette ist defekt oder – falls es die Zieldiskette ist – nicht formatiert (s.o.).
COMMAND ERROR: Sie haben beim Start des Programms einen Fehler gemacht, z.B. „COPY B=A“ anstatt „COPY B:=A:“ geschrieben.

Am besten fertigen Sie sich jetzt noch eine weitere Kopie der CP/M-Masterdisk an, damit Sie eine Möglichkeit haben, die Benutzung der im folgenden erklärten Befehle zu üben. Es geht dabei um die sog. „Built-In Commands“.

„Eingebaute Befehle“

Befehle, die sozusagen in das Betriebssystem „eingebaut“ sind, werden als *Built-In Commands* bezeichnet (Der Gegensatz dazu sind die „Transient Commands“ wie der oben erwähnte „FORMAT“-Befehl, zu dessen Ausführung ein Hilfsprogramm von der Masterdisk geladen werden mußte. Die eingebauten Befehle dagegen funktionieren immer, egal, welche Diskette eingelegt ist). Das erste „Built-In Command“ haben Sie übrigens schon kennengelernt; es war der DIR-Befehl zur Anzeige des Inhaltsverzeichnisses.

ERA – Löschen von Dateien
Zum Löschen von Dateien dient der ERA-Befehl. Er hat die allgemeine Form „ERA Filename.Filetyp <RETURN>“, also z.B. „ERA COPY.COM <RETURN>“ oder „ERA B:ED.COM <RETURN>“.
Nun kann es sein, daß Sie mehr als nur einen File löschen wollen, z.B. alle Dateien vom Typ „.TXT“. Für diesen Zweck gibt es sog. *Wildcard*-Zeichen. So löscht der Befehl „ERA B:*.BAS“ alle Basic-Programme auf der Disk B:; „ERA KUNDEN.*“ entfernt alle Dateien, die mit „KUNDEN“ beginnen (KUNDEN.TXT, KUNDEN.DAT usw.) und mit „ERA C:*.*“ räumen Sie die gesamte Diskette im Laufwerk C: leer! Am besten probieren Sie dies alles einmal aus – aber bitte nur auf einer der vorhin angefertigten Kopien!

REN – Ändern von Dateinamen
Manchmal erscheint es sinnvoll, einen Dateinamen zu ändern. Das ist mit Hilfe des REN-Kommandos (engl.: REName umbenennen) kein Problem. Hier lautet die allgemeine Form „REN neuer Name = alter Name <RETURN>“; so wird mit dem Befehl „REN KOPIE.COM = COPY.COM“ der Name des schon bekannten Kopierprogramms in KOPIE.COM umgewandelt. Probieren Sie auch dies und überzeugen Sie sich mit Hilfe des DIR-Befehls von der Wirkung. (Wenn Sie diese neu benannte Utility benutzen wollen, müssen Sie natürlich z.B. „KOPIE B:=A: <RETURN>“ eingeben!)
Zwei weitere, eingebaute Befehle, die aber keine große Bedeutung haben, sind der USER-Befehl, mit dem eine Diskette in verschiedene Benutzerbereiche aufgetrennt werden kann, und das TYPE-Kommando. Mit dem letztgenannten ist es möglich, den Inhalt einer Datei auf dem Bildschirm oder Drucker auszugeben (z.B. TYPE DEMO.TXT <RETURN>). Sie sollten diesen Befehl aber nur auf Files anwenden, die Texte enthalten, keineswegs aber auf Programme wie COPY.COM, da sonst die merkwürdigsten Dinge geschehen könnten.

↑P – Einschalten des Druckers
Sicherlich haben Sie sich schon gefragt, wie man z.B. ein Inhaltsverzeichnis ausdrucken kann: Das Ein- und Ausschalten des Druckers geschieht durch die Eingabe von „↑P<RETURN>“ (↑P steht dabei für Ctrl-P).
Die Befehlsfolge „↑P<RETURN> DIR B: <RETURN> ↑P<RETURN>“ beispielsweise druckt ein Inhaltsverzeichnis der Disk B: und schaltet danach den Printer wieder ab.

... noch mehr Control-Zeichen
Lassen Sie uns in diesem Zusammenhang noch auf einige weitere Control-Zeichen und ihre Bedeutung eingehen:
Da sind zunächst einmal Steuerzeichen für die Ein- und Ausgabe:
Ctrl-H: (←) Cursor eine Spalte zurück
Ctrl-X: Gesamte eingegebene Zeile löschen
Ctrl-S: Stoppen der Ausgabe auf Bildschirm / Drucker
Ctrl-P: Drucker an/aus (s.o.)
Es gibt noch einige andere, die aber von geringerer Bedeutung sind und deshalb hier nicht aufgeführt werden sollen. Um die Betrachtung des Ctrl-C Befehls kommen wir jedoch nicht herum:

↑C – Warmstart des CP/M
Die Eingabe von „↑C“ am Beginn einer Zeile verursacht – wie auch die Betätigung

der Resettaste – einen sogenannten *Warmstart*; das CP/M-Betriebssystem wird neu initialisiert. Im Prinzip bleiben dabei alle Daten im Speicher unversehrt, sie sind jedoch im allgemeinen nicht mehr „erreichbar".
Ein wichtiger Hinweis: Falls bei einem derartigen Warmstart in dem Bootlaufwerk eine Diskette mit einer anderen CP/M-Version liegt als zum Beginn Ihrer Arbeit, „stürzt das System ab". Aus diesem Grunde sollten Sie dafür sorgen, daß auf allen Disketten dieselbe CP/M-Variante (z.B. 2.23/60K) vorhanden ist. Zur Erinnerung: Das Betriebssystem wird z.B. mit „COPY B:=A:/S" kopiert.
Der Ctrl-C Befehl (↑C) hat zwei Funktionen:
Erstens können Sie damit viele Programme abbrechen (z.B. COPY, FORMAT) und auf die CP/M-Kommandoebene (A>) zurückkehren. Bei Datenverarbeitungsprogrammen u.ä. jedoch kann diese Methode nicht empfohlen werden, da bei einer derartigen Beendigung oftmals nicht alle Daten korrekt abgespeichert werden!
Zweitens müssen Sie immer dann Ctrl-C eingeben, wenn Sie mit dem Betriebssystem arbeiten und Disketten gewechselt haben. Tun Sie dies nicht, wird sich der Rechner weigern, auf die neu eingelegte Disk zu schreiben. Grundsätzlich sollten Sie sich auch daran gewöhnen, bei kommerziellen Programmen nur dann Disketten zu wechseln, wenn der Computer Sie dazu auffordert, da Sie sonst wirklich in „Teufel's Küche" kommen können. (Der Autor dieses Artikels verdankt einem vergessenen RESET-Befehl im CP/M-MBASIC – entspricht dem „↑C" in CP/M – den Verlust einer Stunde Programmierarbeit!)
Auch nach einer Fehlermeldung des CP/M-Systems (BDOS ERR ON X:) sollten Sie mit „↑C" den Rechner neu starten.
Lassen Sie uns zum Abschluß dieses Abschnittes über die eingebauten Kommandos noch kurz auf eben diese Fehlermeldungen eingehen. Es gibt insgesamt nur drei verschiedene:
BAD SECTOR – deutet auf eine beschädigte oder nicht formatierte Diskette hin (s. FORMAT);
R/O – (Read Only/ Nur schreiben) besagt, daß die eingelegte Disk entweder schreibgeschützt ist (Schreibschutzaufkleber entfernen!) oder neu eingelegt wurde, ohne „↑C" einzugegeben;
SELECT – schließlich gibt an, daß ein angesprochenes Laufwerk nicht vorhanden ist.

Weitere Utilities

Kehren wir nun wieder zu den Utilities zurück. Zwei dieser Hilfsprogramme wollen wir noch näher unter die Lupe nehmen. Da ist zum einen „STAT", mit dessen Hilfe sich der Benutzer Informationen über Ein- und Ausgabeeinheiten sowie Dateien geben lassen kann. Nur die letztere Möglichkeit soll hier vorgestellt werden:

STAT – „Statistische Daten"

Der Befehl „STAT <RETURN>" zeigt den Status der eingelegten Disketten – „R/O" steht für schreibgeschützt, „R/W" heißt „Schreiben und Lesen" – sowie den freien Speicherplatz in Kilobytes an. Gibt man dagegen z.B. „STAT B: <RETURN>" ein, erhält man nur die Information über den verfügbaren Speicherraum auf der angesprochenen Disk (hier Drive B:).

Wenn man Genaueres über einzelne Dateien wissen will, bietet sich der Aufruf „STAT filename.typ" an. So erhält man mit „STAT COPY.COM" u.a. Angaben über die Länge des COPY-Programms. Das STAT-Programm erlaubt es übrigens, die oben genannten Wildcard-Characters zu benutzen. So listet der Befehl „STAT B:*.* <RETURN>" Informationen über alle auf der Disk B: enthaltenen Files auf.
Wenn Sie sich diese Tabelle näher ansehen, werden Sie auch die Spalte ACCess (engl.: Zugriff) bemerken. Sie gibt an, ob eine Datei nur gelesen (R/O) oder auch überschrieben werden darf (R/W). CP/M bietet damit eine Möglichkeit, ein versehentliches Löschen von Daten zu verhindern. „Eingeschaltet" wird dieser Schutz mit dem Befehl „STAT filename.filetyp $R/O", die Rücksetzung erfolgt mit „STAT filename.filetyp $R/W". Die Eingabe von „STAT TEST.* $R/O" bewirkt also die Sicherung aller Files, deren Name „TEST" ist (also z.B. TEST.COM, TEST.TEXT), gegen Überschreiben.

PIP – Kopieren von Dateien

PIP ist eine Abkürzung für Peripheral Interchange Program. Dieses Programm bietet zahlreiche Möglichkeiten, Dateien zu transferieren und zu verändern. Umgekehrt proportional dazu ist seine Bedienungsfreundlichkeit. Wir wollen uns deswegen lediglich mit der Aufgabenstellung befassen, die in der Praxis am häufigsten vorkommt, dem Kopieren von Dateien von einer Diskette zur anderen:

Die Lösung dieses simplen Problems indes ist unter CP/M nicht frei von Tücken:

Lädt man nämlich PIP von der Masterdisk und legt danach die Disketten ein, die man zum Kopiervorgang benötigt, müßte man eigentlich mit „↑C" einen Warmstart durchführen, um den Spielregeln von CP/M zu genügen (s.o.: Warmstart/Diskettenwechsel). Dies aber würde zum Abbruch des Kopierprogramms führen.
Es gibt zwei Methoden, um diese Schwierigkeiten zu umgehen:

Ersten können Sie dafür sorgen, daß entweder auf der Original- oder Quellendiskette das PIP-Programm vorhanden ist, da dann ein etwaiger Diskettenwechsel entfällt. Dazu muß es sich selbst kopieren, z.B. mit der Anweisung „PIP B:=A:PIP.COM <RETURN>". Durch diesen Befehl überträgt sich das auf der Disk A: befindliche PIP-Programm selbst auf die Disk B:.
Der zweite Weg ist weniger kompliziert, unterwirft Sie aber gewissen Beschränkungen: Sie betätigen „↑C" und starten dann PIP nach untenstehender Anleitung. Entnehmen Sie die Masterdisk mit dem Kopierprogramm und legen Sie in dasselbe Laufwerk Ihre Originaldisk, von der Sie kopieren wollen, ein. PIP wird nun einwandfrei arbeiten, solange Sie nicht versuchen, auch auf die Originaldisk zu schreiben. Sie können also nur Daten in einer Richtung übertragen.

PIP wird aktiviert durch die Anweisung „PIP <RETURN>" und meldet sich mit einem „*" am linken Bildschirmrand. Die allgemeine Form des Kopierbefehls lautet „Ziellaufwerk:=Originallaufwerk:Filename.Filetyp". Beispielsweise überträgt die Anweisung „B:=A:COPY.COM <RETURN>" das Programm „COPY" von der Disk A: auf das Laufwerk B:. Will man mehrere Dateien gleichzeitig ansprechen, kann man auch hier die Wildcards verwenden. So kopiert „A:=B:*.*" alle Dateien von B: nach A:.

Noch drei Anmerkungen zu PIP:

1. Das Programm wird durch die Eingabe von „↑C" oder <RETURN> verlassen.

2. Es ist nicht möglich, Daten von einer Diskette auf eine andere unter Benutzung nur eines Laufwerkes zu übertragen.

3. Befindet sich auf der Zieldiskette bereits eine Datei gleichen Namens, die nicht schreibgeschützt ist, wird sie ohne vorherige Warnung überschrieben.

CP/M 2.2 Referenztabelle

A. LAUFWERKSBEZEICHNUNGEN

Laufwerke werden durch einen Buchstaben (A..F) mit nachfolgendem Doppelpunkt bezeichnet. Beim APPLE gilt folgende Zuordnung:

A: Slot 6, Drive 1
B: Slot 6, Drive 2
C: Slot 5, Drive 1

B. DATEINAMEN

Filenamen bestehen aus 1 – 8 Buchstaben (fname), einem Punkt sowie einer Filetypangabe (ftyp).

Die Bedeutung einiger Filetypangaben:

ASC – ASCII-Textfile
ASM – 8Ø8Ø-Assemblerquelltext
BAK – Sicherheitskopie eines Files (z.B. von WS erzeugt)
BAS – BASIC-Programmdatei
COM – Direkt ausführbares Programm
DOC, TXT, TEX, READ.ME – Textdateien
LIB – Library-Datei
PAS – Pascal-Programmdatei (z.B. TURBO-Pascal)
PRN – Assemblerlisting
$$$ – Unvollständige Files, Arbeitsdateien (WS) usw.

Filenamen können mit einem Laufwerkskenner d: verbunden werden, um Bezug auf ein bestimmtes Laufwerk zu nehmen.

Bsp.: A:COPY.COM, B:INFO.TXT, C:DEMO.DAT

Einige CP/M-Kommandos erlauben den Einsatz sog. "WILDCARDs", um mehrere Files gleichzeitig anzusprechen.

* – ersetzt fname oder ftyp (*.COM, *.*, TEXT.*)
? – ersetzt einzelne Buchstaben (TEST????.TXT)

C. "BUILD-IN"-KOMMANDOS

"Eingebaute" Kommandos sind ständig verfügbar. Alle Eingaben sind mit <RETURN> abzuschließen.
(Ausnahme : Control-Zeichen)

C1. Bezugslaufwerkswechsel (logged Drive)

<d:> – wechselt auf Drive d:

C2. ↑C – Warmstart

↑C (Ctrl-C) – initialisiert CP/M neu.

↑C bricht u.U. laufende Programme ab und muß unbedingt bei Diskettenwechsel auf Betriebssystemebene eingegeben werden.

C3. Ein-/ Ausgabe-Control-Zeichen

↑X – Zeile löschen
↑H – (<-) letztes Zeichen löschen
↑S – Ausgabe stoppen
↑P – Drucker an/aus

C4. DIR – Anzeige des Inhaltsverzeichnisses

DIR – Anzeige des DIRectories im Bezugslaufwerk
DIR d: – dasselbe, nur für Laufwerk d: (z.B.: DIR A:)
DIR fname.typ – Anzeige nur bestimmter Namen/ Typen

Bsp.: DIR A:*.COM

C5. ERA – Löschen von Dateien

ERA d:fname.ftyp – Löschen der Datei fname.ftyp in Drive d:
Wildcards erlaubt (ERA *.COM).

C6. REN – Neubenennen von Dateien

REN d:neuerfname.ftyp = d:alterfname.ftyp –
Datei alterfname.ftyp in Drive d: wird in neuerfname.ftyp umbenannt.
Wildcards sind nicht erlaubt.

C7. USER – Umschalten der Benutzerbereiche

USER n – Schaltet auf Benutzerbereich n (1...15) um. Nach Booten ist Userbereich Ø eingeschaltet.

C8. TYPE – Ausgabe einer Datei am Bildschirm / Drucker

TYPE d:fname.ftyp – Listet eine Datei am Bildschirm oder auf dem Drucker (TYPE READ.ME).

D. "TRANSIENTE" KOMMANDOS

Zur Ausführung eines "transienten" Kommandos muß ein entsprechender File (ftyp: .COM) auf dem Bezugslaufwerk vorhanden sein.

D1. COPY – Kopieren ganzer Disketten

COPY neuerd:=alterd: – Kopiert gesamte Diskette von Drive alterd: nach neuerd:.
COPY neuerd:=alterd: /S – Kopiert nur das CP/M-Betriebssystem

Je nach Version des COPY-Programms kann eine vorherige Formatierung notwendig sein.

D2. FORMAT – Formatieren von Disketten

FORMAT d: – Formatiert Diskette in Drive d:

D3. STAT – "Statistische" Angaben

STAT d: – Zeigt freien Speicherplatz auf Disk d: an.
STAT d:fname.ftyp – Zeigt Länge und Status der Datei fname.ftyp an. Wildcards erlaubt.
STAT d:fn.typ $stat – Setzt angesprochene Datei auf Status stat:
stat = R/O ; Datei schreibgeschützt
R/W ; Datei nicht schreibgesch.

D4. PIP – Kopieren von Dateien

PIP zield:=origd:fname.ftyp – Kopiert Datei fname.ftyp von Laufwerk origd: nach zield:. Wildcards erlaubt.

Weitere Aufrufmöglichkeiten der vorgenannten Befehle sowie Angaben über andere CP/M 2.2 Anweisungen entnehmen Sie bitte den entsprechenden Handbüchern.

Microsoft Basic leicht geMACht

von Pit Capitain

Teil 3: Die Ein- und Ausgabe

6. EIN-/AUSGABEBEFEHLE

Nachdem im letzten Teil *Funktionen* erklärt wurden, die mit der Ein-/Ausgabe im Zusammenhang standen, werden nachfolgend die entsprechenden *Befehle* vorgestellt.
Es gibt wie im Applesoft-Basic Befehle, die nur *eine* Datei ansprechen (z.B. den Bildschirm), und solche Befehle, mit denen *verschiedene* Dateien bearbeitet werden können (z.B. der Befehl „OPEN“).

6.1. Tastatur

Es gibt zwei Befehle, die von der Tastatur Zeichen einlesen:
INPUT; “String“; Variablenliste – Es wird wie beim Applesoft-Basic eine Liste von Variablen eingelesen. Falls im Befehl ein String angegeben ist, wird dieser ausgegeben, anschließend ein Fragezeichen. Soll das Fragezeichen unterdrückt werden, so kann nach dem String auch ein Komma stehen. Der Befehl
INPUT “Name: “, N$
gibt also nur den String “Name: “ ohne ein Fragezeichen aus. Falls direkt nach dem Wort „INPUT“ ein Strichpunkt steht, dann wird nach der Eingabe der Variablen *kein* Return ausgegeben. Man kann dann also direkt hinter der Eingabe weiter ausdrucken.
LINE INPUT; “String“; Stringvariable – Mit diesem Befehl wird eine Stringvariable eingelesen. Es wird zunächst der im Befehl angegebene String ohne ein nachfolgendes Fragezeichen ausgegeben. Dieser String kann auch weggelassen werden. Anschließend werden alle Zeichen bis zu der Return-Taste in die Stringvariable eingelesen, mit Kommas und allen Sonderzeichen. Falls direkt nach dem Wort „LINE INPUT“ ein Strichpunkt folgt, dann wird wie beim „INPUT“ nach der Eingabe kein Return ausgegeben.

6.2. Lautsprecher

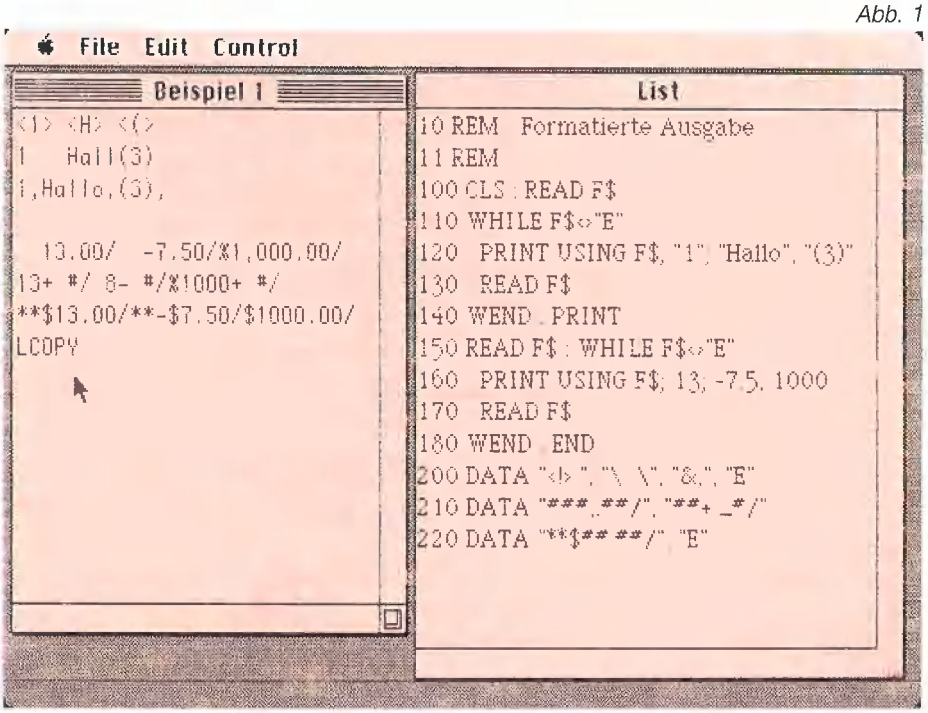
BEEP – Der Lautsprecher piepst einmal. Denselben Effekt erreicht man durch den Befehl „PRINT CHR$ (7);“.

Obwohl der Macintosh drei Software-Tongeneratoren eingebaut hat, ist dies der einzige Basic-Befehl zur Tonerzeugung. Die Tongeneratoren können leider nur über Maschinensprache benutzt werden.

6.3. Bildschirm

CLS – löscht das Ausgabefenster. Dies entspricht dem Applesoft-Befehl „HOME“.
PRINT Liste von Ausdrücken – entspricht dem Applesoft-Befehl „PRINT“.

Abb. 1



Unterschiede gibt es nur bei der Ausgabe von Zahlen: Nach Zahlen wird grundsätzlich ein Leerzeichen ausgegeben, desgleichen vor positiven Zahlen. Vor negativen Zahlen steht ein Minuszeichen. Einfach genaue Zahlen (vgl. Teil 1 der Serie), die mit weniger als 8 Ziffern dargestellt werden können, werden ohne Hochzahl ausgegeben, z.B. wird 1E-7 als „.0000001" ausgegeben (7 Ziffern), dagegen 1E-8 als „1E-08". Analoges gilt für doppelt genaue Zahlen. Hier liegt die Grenze bei 17 Ziffern.

SPC (I) – gibt I Leerzeichen aus. I muß im Bereich -32768 bis 32767 liegen, wobei negative Zahlen natürlich kein Leerzeichen erzeugen. Dieser Befehl kann nur in einem „PRINT"- oder „LPRINT"-Befehl (s.u.) benutzt werden.
Falls die Zeilenlänge des Bildschirms mit dem Befehl „WIDTH" (s.u.) auf W gesetzt wurde, werden nur I MOD W Leerzeichen ausgegeben.
Ansonsten (Zeilenlänge auf „unendlich") wird beim ersten Erreichen der Position 255 durch den „SPC"-Befehl ein Return eingefügt! Dies steht im Widerspruch zu den Angaben im Handbuch.

TAB (I) – bewegt den Ausgabezeiger zur horizontalen Position I. Falls sich der Ausgabezeiger schon hinter dieser Position befindet, wird zu dieser Position in der nächsten Zeile gesprungen. I muß im Bereich von -32768 bis 32767 liegen (negative Werte entsprechen der Position ganz links). Dieser Befehl kann nur in einem „PRINT"- oder einem „LPRINT"-Befehl (s.u.) verwendet werden. Auch hier wird bei einer Zeilenlänge von W Zeichen nur zur Position I MOD W gesprungen.

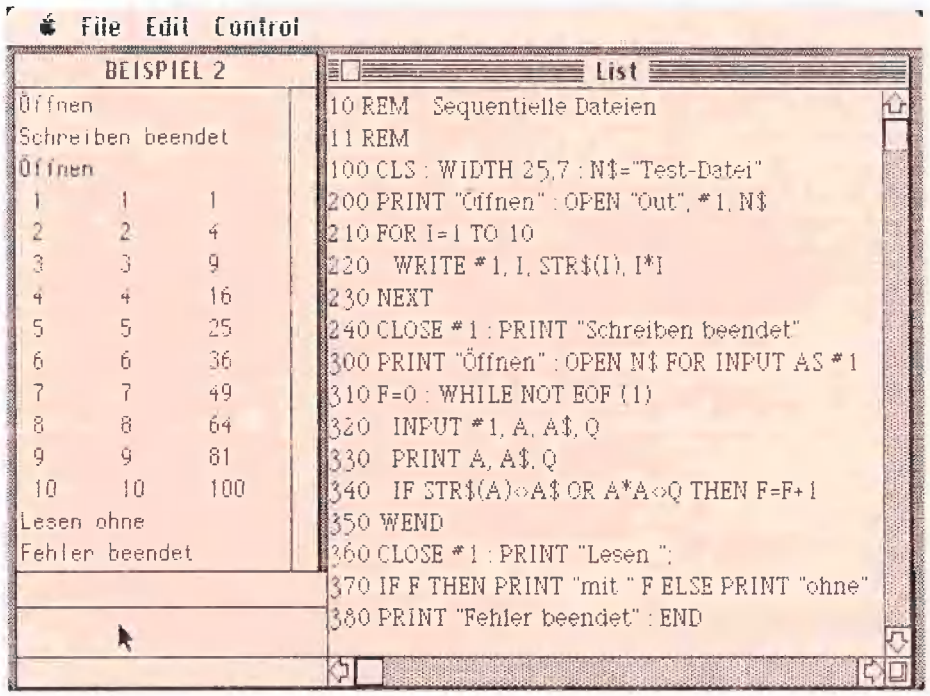
PTAB (I) – bewegt den Ausgabezeiger zum I-ten Pixel (= einzelner Punkt des Grafik-Bildschirms). Falls sich der Ausgabezeiger schon hinter dieser Position befindet, so wird – im Unterschied zum Befehl „TAB" – zu der Position in *derselben* Zeile zurückgesprungen. I muß im Bereich von 0 bis 32767 liegen. Dieser Befehl kann nur in einem „PRINT"-Befehl verwendet werden.

PRINT USING F$; Liste von Ausdrükken – gibt die Zahlen und Strings in der angegebenen Liste in einem bestimmten Format aus. Dieses Format wird durch den ersten String F$ festgelegt. Dieser String enthält bestimmte Zeichen, die das Ausgabeformat bestimmen. Es gibt folgende Zeichen:

Zur Formatierung von Strings:
! – Nur das erste Zeichen eines Strings wird ausgegeben.
\n Leerzeichen – Es werden 2 + n Zeichen eines Strings ausgegeben. Falls der String zu kurz ist, werden Leerzeichen ergänzt.
& – Der String wird unverändert ausgegeben.

Zur Formatierung von Zahlen:
– stellt eine Ziffer dar. Falls die Zahl weniger Ziffern als gefordert hat, so werden vorne Leerzeichen ergänzt. Falls die Zahl aber zu lang ist, so wird vor der Zahl ein „%" ausgegeben. Zwischen den Ziffer-Zeichen kann ein Dezimalpunkt („.") angegeben werden.
+ – am Anfang bzw. am Ende des Formatier-Strings bedeutet, daß das Vorzeichen einer Zahl (+ oder -) vor bzw. hinter der Zahl ausgegeben wird.
- – am Ende des Formatier-Strings bedeutet, daß negative Zahlen von einem Minuszeichen gefolgt werden.
****** – am Anfang des Formatier-Strings bedeutet, daß bei Zahlen, die weniger Ziffern als gefordert haben, Sternchen („*") anstatt Leerzeichen ergänzt werden.
$$ – Ein Dollar-Zeichen wird vor den formatierten Zahlen ausgegeben.
****$** – Die beiden eben beschriebenen Wirkungen werden kombiniert.
, – direkt vor einem Dezimalpunkt bewirkt, daß nach Tausender-, Millionen-Ziffern usw. ein Komma ausgegeben wird (amerikanische Notation).
↑ ↑ ↑ ↑ – am Ende einer Ziffernfolge bewirkt, daß eine Zahl im exponentiellen Format ausgegeben wird. Die vier Pfeile stehen für die vier Zeichen "E+xx", mit denen der Exponent angegeben wird.
_ – bewirkt, daß das nächste Zeichen des Formatier-Strings direkt ausgegeben wird.
Beispiele für manche der vielen Formatier-Zeichen findet man in der **Abb. 1**.

Abb. 2



WRITE Liste von Ausdrücken – ähnlich wie der Befehl „PRINT". Die einzelnen Ausdrücke werden aber durch Kommas getrennt ausgegeben. Strings werden in Anführungszeichen eingeschlossen. Vor und nach Zahlen wird im Gegensatz zum „PRINT"-Befehl kein Leerzeichen ausgegeben. Nach dem letzten Ausdruck wird ein Return ausgegeben. Z.B. erzeugt der Befehl
WRITE 1,A$,"Test",A,-1
die folgende Ausgabe (mit A$="Hallo", A=27):
1,"Hallo","Test",27,-1

LIST Zeile1-Zeile2 – öffnet ein Listfenster und gibt das Programmlisting ab der Zeile mit der Nummer Zeile1 aus. Dies entspricht fast dem Menü-Befehl „LIST", der jedoch immer den *Anfang* des Programms zeigt.
Man kann das Listing auch an ein anderes Ausgabegerät schicken, indem man an den Befehl den Zusatz ", Gerät" anhängt. Der Befehl
LIST 100-999, "COM1:"
schickt ein Listing der Programmzeilen 100 bis 999 an die serielle Schnittstelle.

6.4. Drucker

Es gibt einige Befehle, die direkt den Drucker ansteuern, ohne daß wie beim Applesoft-Basic eine Datei dafür geöffnet werden müßte:

LCOPY – druckt den augenblicklichen Bildschirminhalt auf dem Imagewriter aus. Die **Abb. 1** wurde mit diesem Befehl erzeugt.

LPRINT und **LPRINT USING** – entsprechen den Befehlen „PRINT" und „PRINT USING" (s.o.), aber Ausgabe auf dem Drucker.

LLIST Zeile1-Zeile2 – druckt ein Listing des im Speicher befindlichen Programms mit dem angegebenen Zeilenbereich. Dieselbe Wirkung kann man auch mit dem Befehl
LIST Zeile1-Zeile2, "LPT1:"
erzielen.

6.5. Allgemeine Dateien

In diesem Abschnitt werden solche Befehle erklärt, mit denen unterschiedliche Dateien bearbeitet werden können. Das heißt, daß diese Befehle z.B. Dateien auf verschiedenen Disketten oder den Bildschirm, den Drucker usw. ansprechen können. Die Bezeichnung von Dateien und die Namen von Ein-/Ausgabegeräten wurden bereits im Teil 1 der Serie behandelt.

6.5.1. Arten von Dateien

Es gibt zwei unterschiedliche Arten, eine Datei zu benutzen:
– man kann entweder Dateien *sequentiell* lesen bzw. beschreiben, d.h. immer ein Zeichen nach dem anderen, oder
– man kann an ganz bestimmten Stellen auf eine Datei zugreifen, ohne eine bestimmte Reihenfolge einhalten zu müssen (wahlfreier Zugriff, *Random Access*). Die Datei ist dazu in mehrere gleichlange „Felder" aufgeteilt, die numeriert sind. Man kann nun wahlweise auf das Feld mit einer beliebigen Nummer zugreifen.
In diesem Zusammenhang spricht man dann auch von sequentiellen Dateien bzw. von Random-Access-Dateien. Diese Unterscheidung gibt es auch beim Apple-DOS. Sie wird bei den nächsten Abschnitten immer wieder auftauchen.

6.5.2. Allgemeine Dateibefehle

Dateien werden mit dem Befehl „OPEN" geöffnet. Diesen Befehl kann man auf zwei verschiedene Arten schreiben. Die erste Möglichkeit ist:

OPEN Modus, #N, Name, Länge – öffnet die Datei mit dem angegebenen Namen unter der Dateinummer N. Modus ist ein String, dessen erster Buchstabe die Art der Datei angibt:
"R" – Random-Access-Datei, Ein- oder Ausgabe
"I" – sequentielle Eingabedatei
"O" – sequentielle Ausgabedatei
"A" – sequentielle Ausgabedatei. Der Dateizeiger steht aber am Ende der Datei. Dadurch wird also an die Datei etwas angehängt (Append).
Bei Random-Access-Dateien kann angegeben werden, wie lang die einzelnen Felder sind. Falls diese Angabe weggelassen wird, nimmt das Microsoft-Basic eine Länge von 128 Bytes an.
Die zweite Art des „OPEN"-Befehls hat eine andere (etwas besser lesbare) Syntax:

OPEN Name FOR Modus AS #N LEN=Länge – Die Datei mit dem angegebenen Namen wird unter der Dateinummer N geöffnet. Modus ist eines der folgenden Wörter:

INPUT – sequentielle Eingabedatei
OUTPUT – sequentielle Ausgabedatei
APPEND – sequentielle Ausgabedatei. Der Dateizeiger steht aber am Ende der Datei.

Falls der Ausdruck „FOR Modus" weggelassen wird, handelt es sich um eine Random-Access-Datei. In diesem Fall kann die Länge der einzelnen Felder angegeben werden. Bei fehlender Angabe wird die Länge 128 angenommen.

CLOSE #N1,#N2,... – schließt die angegebenen Dateien. Falls keine Dateinummer angegeben ist, werden alle offenen Dateien geschlossen.

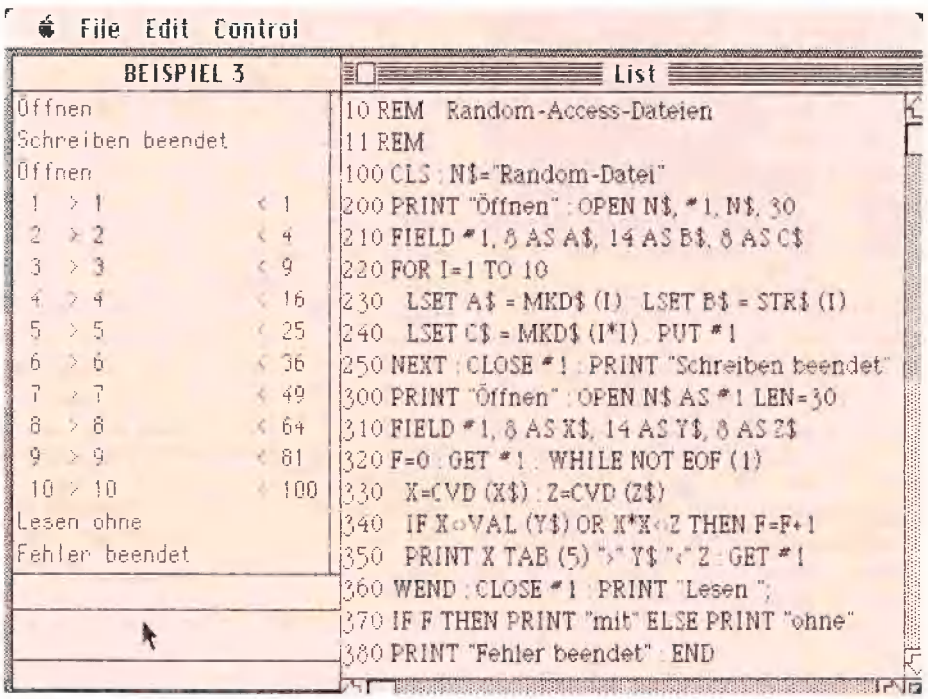
RESET – schließt ebenfalls alle offenen Dateien. Der Unterschied zum „CLOSE"-Befehl besteht darin, daß zusätzlich das neue Disketteninhaltsverzeichnis auf Diskette geschrieben wird.
Bei „CLOSE" wird das Inhaltsverzeichnis nur im Speicher auf den neuesten Stand gebracht, nicht aber auf der Diskette. Das kann bei einem Stromausfall oder z.B. bei Maschinensprache-Experimenten die Folge haben, daß das Inhaltsverzeichnis einer Diskette nicht mit ihrem Inhalt übereinstimmt.

FILES Name – gibt an, ob sich eine Datei mit dem angegebenen Namen auf Diskette befindet. In diesem Fall wird der Name der Datei im Ausgabefenster angegeben, anderenfalls wird die Fehlermeldung „FILE NOT FOUND" erzeugt.
Falls kein Name hinter dem Befehl angegeben ist, werden die Namen von allen auf Diskette befindlichen Dateien im Ausgabefenster ausgegeben.

KILL Name – löscht die angegebene Datei.

NAME Alt AS Neu – Ändert den Namen einer Disketten-Datei. Der alte Name muß existieren, der neue Name darf nicht schon einmal vorkommen, ansonsten wird ein Fehler erzeugt.

Abb. 3



Bei Random-Access-Dateien besteht die Möglichkeit, die Felder, in die die Datei aufgeteilt ist, zu „formatieren". Dies geht mit dem folgenden Befehl:
FIELD #N, B1 AS S1$, B2 AS S2$, ... – Dieser Befehl formatiert die Felder einer geöffneten Datei (Nummer N) wie folgt: Die ersten B1 Zeichen können unter dem Namen S1$ bearbeitet werden, die nächsten B2 Zeichen unter dem Namen S2$, usw.
Beispiele für die Verwendung dieser Befehle sieht man in **Abb. 2** und **Abb. 3**. Diese Programme schreiben zunächst Zahlen und Strings auf eine Datei. Danach wird die Datei wieder gelesen und die Daten werden auf ihre Korrektheit hin überprüft. In Abb. 2 wird eine sequentielle Datei, in Abb. 3 eine Random-Access-Datei verwendet.

6.5.3. Eingabe

INPUT #N, Variablenliste – liest die angegebenen Variablen von der geöffneten Datei mit der Nummer N ein.
LINE INPUT #N, Stringvariable – liest eine Zeile von der Datei mit der Nummer N in die angegebene Stringvariable ein (wie bei dem Befehl „LINE INPUT", s.o.). Dies kann z.B. dazu benutzt werden, um ein im ASCII-Format gespeichertes Basic-Programm von einem anderen Programm aus einzulesen.
GET #N, R – liest das Feld Nummer R von der Random-Access-Datei mit der Nummer N in den Speicher. Falls R nicht angegeben ist, wird das nächste Feld (nach dem letzten „GET") eingelesen. Nach diesem Befehl können u.a. mit den beiden vorigen Befehlen Daten aus dem Feld eingelesen werden, aber auch über die Stringvariablen in einem „FIELD"-Befehl. Beispiele zu diesen Befehlen sieht man in den Abb. 2 und 3.

6.5.4. Ausgabe

WIDTH Gerät, W, T oder
WIDTH #N, W, T – legt die Zeilenlänge und Tabulator-Positionen des Ausgabegerätes bzw. der Ausgabedatei fest.
W ist die Länge der Zeilen. Nach W Zeichen fügt Microsoft-Basic ein Return ein. Falls W den Wert 255 hat, so wird die Zeilenlänge auf „unendlich" gesetzt, d.h. es wird nirgends innerhalb von Zeilen ein Return eingefügt (man beachte aber die obige Anmerkung zum Befehl „SPC").
Tabulatoren werden alle T Zeichen gesetzt. Zu diesen Marken wird zum Beispiel gesprungen, wenn in einer „PRINT"-Anweisung ein Komma zwischen zwei Ausdrücken steht, also beispielsweise bei PRINT A, B,.

PRINT #N, Ausdrücke – entspricht „PRINT", wobei aber auf die Datei mit der Nummer N geschrieben wird.
PRINT #N, USING F$; Ausdrücke – wie „PRINT USING F$"; es wird aber auf die Datei mit der Nummer N geschrieben.
WRITE #N, Ausdrücke – wie „WRITE"; es wird aber auf die Datei mit der Nummer N geschrieben. Dies ist recht nützlich, da die Ausdrücke, die dadurch auf die Datei geschrieben werden, später mit dem „INPUT #"-Befehl wieder eingelesen werden können, da sie schon voneinander durch Kommas usw. getrennt sind.

Für Random-Access-Dateien gibt es noch mehr Befehle:
PUT #N, R – Gegenstück zum „GET #"-Befehl (s.o.). Es wird also ein Feld, das vorher mit „PRINT #", „WRITE #" oder mit den beiden nachfolgenden Befehlen gefüllt wurde, auf die Datei mit der Nummer N geschrieben, und zwar an die Stelle mit der Nummer R.
LSET S$ = Stringausdruck und
RSET S$ = Stringausdruck – setzt den angegebenen Stringausdruck in das Feld einer Random-Access-Datei ein, und zwar an der Stelle, die in einem „FIELD"-Befehl (s.o.) der Variablen S$ zugeordnet wurde. Mit „LSET" wird der Stringausdruck linksbündig, mit „RSET" rechtsbündig eingesetzt, wobei eventuell Leerzeichen ergänzt werden. Dies kann auch unabhängig von Dateien benutzt werden, z.B.:
A$=SPACE$(20) : RSET A$=N$
Diese beiden Befehle haben zur Folge, daß in dem String A$ (durch den ersten Befehl 20 Zeichen lang) der String N$ rechtsbündig eingefügt wird (zweiter Befehl).
Falls man nicht Strings, sondern Zahlen in ein formatiertes Feld einsetzen will, muß man die Funktionen „CVI", „CVS", „CVD", „MKI$", „MKS$" oder „MKD$" verwenden (s. Teil 2 der Serie).
Beispiele zu diesen Befehlen sind in den Abb. 2 und 3 zu sehen.
Um das im Speicher befindliche Programm auf eine Datei zu schreiben, gibt es noch den Befehl
SAVE Name – schreibt das Programm aus dem Speicher auf die Datei mit dem angegebenen Namen, und zwar im *binären* Format.
Um das Format zu ändern, können an den Befehl noch die folgenden Zusätze angehängt werden:
,A – ASCII-Format
,P – geschütztes Format (Protected).
SAVE "Prog1",A speichert also das Programm unter dem Namen "Prog1" im ASCII-Format ab.
Falls kein Name angegeben wird, so erscheint ein Dialogfenster, in dem der Benutzer den Dateinamen und das Format festlegen kann (s. Teil 1 der Serie).

Redaktioneller Hinweis: Inzwischen ist die Version 2.0 Microsoft-Basic bei uns eingegangen. Da diese Neuversion stark geändert wurde (Full-Screen-Editor, Zeilennummern fakultativ usw.), standen wir vor dem Problem, ob wir die bereits vorliegende Serie zur Version 1.0 bzw. 1.1 abschließen sollten. Um Verwirrungen zu vermeiden, haben wir uns dazu entschlossen, die Aufsatzfolge in der bisherigen Form zu Ende zu führen. Da der Macintosh leider noch nicht völlig „ausgegoren" ist, werden sich Mac-Besitzer daran gewöhnen müssen, daß an den Mac-Programmen ständig gefeilt wird. Der Umstand, daß selbst ein so weltweit renommiertes Software-Haus wie Microsoft mit der Implementierung von Basic und Multiplan auf dem Mac sichtliche Schwierigkeiten hatte, läßt erahnen, wie es erst den kleineren Software-Häusern mit diesem eigenwilligen Gerät ergehen wird. Wer einen Macintosh erwirbt, muß wissen, daß von bestimmten Programmen in kürzeren Abständen, als dies bei anderen Computern der Fall ist, Neuversionen erscheinen werden. Wem dies zu teuer ist, muß bei einer Altversion bleiben, so daß die Fortführung der Serie von daher ihre Berechtigung hat. In den USA kostet der Umtausch von Microsoft-Basic ca. $ 50.00.

Orgasoft plant für die Zukunft

Orgasoft wurde 1980 als GmbH gegründet, deren beide Gesellschafter Klaus M. Dargel und Jürgen Hartwich mit je 50 % beteiligt sind. Der Vorläufer der Firma wurde im Jahre 1977 durch Jürgen Hartwich ins Leben gerufen, und zwar als Beratungsfirma für Materialfluß und Büroorganisation.
Seit 1980 ist Orgasoft Apple-Vertragshändler. Im Jahr 1984 wurde zusätzlich der Mikrocomputer Schneider CPC 464 ins Programm aufgenommen, um die Produktpalette nach unten abzurunden. Der Firmensitz ist in 7732 Niedereschach. Ein Verkaufsbüro befand sich seit 1. 1. 1983 in 7730 Villingen-Schwenningen.

Ladengeschäft: Jetzt am 23. Februar wurde dann das neue Verkaufsbüro in der Villinger Innenstadt mit angeschlossenem Computer-Shop eröffnet. Auf 150 qm Büro- und Ladenfläche wird die gesamte Apple-Produktpalette, der Schneider CPC 464 und ein umfangreiches Peripherie-Angebot von Plattenspeichern, Druckern usw. geführt. Orgasoft ist ein Level-1-Service-Center mit eigener Reparatur-Abteilung und hat 7 Mitarbeiter sowie einige freiberufliche Programmierer. In dem renovierten Altstadthaus werden in einer geschmackvollen Ladeneinrichtung weiterhin Zubehör sowie ein breites Sortiment an Fachbüchern und Fachzeitschriften angeboten. Der Kundenkreis setzt sich aus Privatpersonen, Klein- und Mittelbetrieben, Schulen und einigen Firmen der Großindustrie zusammen. Für das gesamte Produktprogramm von Apple und Schneider werden Abdeckhauben in eigener Regie hergestellt und an Endkunden sowie über das Händlernetz vertrieben.

Exportgeschäft: Nachdem Orgasoft sich schon seit 3 Jahren mit dem Export nach Tunesien befaßt und dort bereits zahlreiche Firmen mit Apple-Computern sowie mit französischsprachiger Software ausgestattet hat, ist man derzeit dabei, eine „Orgasoft Tunisie" zu gründen. Für den Export ist Geschäftsführer Jürgen Hartwich zuständig. U.a. wurden die deutschen Auslandshandelskammern in Tunesien und Ecuador mit Apple-Rechnern und spezieller Software ausgestattet.

Software-Entwicklung: Die Firma Orgasoft gehört zu den ganz wenigen Apple-Händlern mit einer eigenen Software-Abteilung, die Eigenentwicklungen und kundenspezifische Programme für Apple-Computer erstellt. Ca. 80 % der vertriebenen Software sind Eigenprodukte. Von einigen Fremdprogrammen wurden Lizenzen erworben.
Die Firma Orgasoft verfügt inzwischen über eine große Software-Palette, die an Endabnehmer und über das Händlernetz vertrieben wird, z.B. folgende Branchenlösungen:
- Reisebüro
- Autovermietung
- Verwaltung von Palettenlägern
- Anschluß von Wägesystemen an Apple-Computer
- Lohn und Gehalt
- Fakturierung
- Finanzbuchhaltung
- Immobilienmakler
- Baukalkulation

Das Fakturierungsprogramm läßt sich mit dem Applewriter IIe verknüpfen, so daß mit den Kundenadressen Serienbriefe erstellt werden können.

Selbstverständlich wird auch Software der großen, internationalen Softwarehäuser angeboten. Wie sich anläßlich der Eröffnung des Computer-Shops zeigte, ist Orgasoft immer gut für neue Marketing-Ideen. In den regelmäßig erscheinenden Orgasoft-News wird das Händlernetz über Aktionen, so z.B. über den Software-Koffer, bestehend aus einem Marken-Aktenkoffer mit 4 kompletten Programmpaketen als Händler-Demo, unterrichtet. Ferner werden auch herstellerneutrale Beratungen auf dem Sektor Materialfluß sowie Büroorganisation und EDV-Einführungen durchgeführt.

Auf dem Abstellgleis: Steve Wozniak

In der amerikanischen Zeitschrift „Byte" erscheint seit Dezember 1984 eine mehrteilige „Apple Story" in Form eines Interviews mit Steve Wozniak. Obgleich substantiell nichts Neues gebracht wird, denn Apples Aufstieg von der Garagenfirma bis zum Weltkonzern ist ja wohl jedem bekannt, überraschen doch die resignierenden Töne, die Wozniak insbesondere im Januar-Heft der „Byte", S. 167 anschlägt. Insider kann dies allerdings nicht mehr erstaunen, denn schon seit langem ist bekannt, daß Steve Wozniak nur noch ein Außenseiter, quasi ein „Odd-man-out" ist, also eine Figur, die nicht mehr in das Puzzle paßt.

– Wäre es nach Wozniak gegangen, dann würden Apple-Clubs wie CALL APPLE in den USA oder AUGE in Deutschland nicht von Apple ignoriert oder gar mit Füßen getreten.
– Wäre es nach Wozniak gegangen, dann hätte man beim Apple III und bei der Lisa weniger Geheimniskrämerei betrieben (und wahrscheinlich wären dann auch beide Projekte nicht „in die Hose gegangen").
– Wäre es nach Wozniak gegangen, dann hätte man den *Quell*-code der Betriebssysteme SOS (für Apple III) und ProDOS (für Apple II) publik gemacht. (Wozniak: „It's the greatest thing in the world, but I wish we gave out listings of it ... We made it very difficult for outside developers instead of providing all the information ..." Byte: „Has that attitude changed now?" Wozniak: „No. It's still the most negative thing in our whole company, and it will be for years.").

Bei Apple richtet man sich jedoch de facto schon lange nicht mehr nach Wozniak. Als ich in Verbindung mit meinem Buch „Apple ProDOS für Aufsteiger", das übrigens im Mai 1984 noch *vor* den ersten amerikanischen ProDOS-Publikationen erschien, einen disassemblierten Quellcode von ProDOS ankündigte, erhielt ich sofort eine Abmahnung der Rechtsanwaltskanzlei Strobl, Killius & Vorbrugg in München, die die Firma Apple Computer GmbH mit der klaren Order vertritt, jeden abzumahnen, der in einen Apfel beißt, egal ob frisch oder faul. Daß der ProDOS-Quellcode letztlich nicht erschien, lag indes weniger an der Einschüchterung denn an meiner zu knapp bemessenen Freizeit, denn ein kommentierter Objektcode ist genauso rechtens wie ein kommentiertes Gedicht, und in beiden Fällen darf übrigens das Original als Großzitat mitgeliefert werden, wie bereits jedem Schulbuben aus seinen Deutschbüchern bekannt ist. Soeben erschien im Franzis-Verlag ein DOS-Buch (Ruhland, „DOS 3.3"), das den *gesamten* Objektcode („Großzitat") enthält; eine Abmahnung ist damit praktisch vorprogrammiert.
Vergleicht man Wozniaks obigen Wunsch mit der heutigen Einstellung der Firma Apple, dann sieht man deutlich, daß inzwischen ganz andere Strömungen dominieren. Wozniak ist längst ausgebootet: „Woz is out. Poor Woz. That's the way the cookie crumbles."
Ulrich Stiehl

AUGE und Apple-München

Die Apple User Group Europe e.V. (AUGE) befindet sich seit geraumer Zeit in einer Krise, wobei die Fluktuation im Vereinsvorstand ironischerweise eher Wirkung denn Ursache der Misere ist, die wohl letztlich durch eine im Hintergrund agierende „zweite Führungsgarde" („Background-Mafia") bedingt ist.
Die auf die Gründungsphase unter Regie von Wolfgang Dederichs folgende Expansionsära bis 1973 unter dem Vorsitzenden Klaus Giese stellt meines Erachtens die fruchtbarste Periode der AUGE dar. Infolge permanenter, vereinsinterner Querelen ging es danach rapide bergab. Zunächst wurde im März 1983 Klaus-Peter Lucius zum Vorsitzenden gewählt, der jedoch bereits nach 5 Monaten seinen „Hut nahm". Auf der nächsten Hauptversammlung wurde Peter Derks im März 1984 zum Vorsitzenden erkoren, der ebenfalls nach kurzer Zeit im Juli 1984 „abdankte". Die „Interregnen" waren durch verwaiste „Hutständer" auf der einen und Vereinsnörgeleien auf der anderen Seite gekennzeichnet. Als dann unser „Peeker" und kurz danach „Apple's unabhängiges Magazin" erschienen, machte sich nervöse Hektik breit. Um eine Abtrift der Mitarbeiter des User-Magazins zu verhindern, wurde eine großzügige Entlohnung vom „kopflosen" Vorstand bewilligt, obgleich noch vor kurzem (Heft 3/1984, S. 9) sowie in früheren User-Magazinen (z.B. Heft 2/1982, S. 2) Bezahlungen aus idealistischen Gründen kategorisch abgelehnt wurden. Den ab 1985 auf DM 100,– erhöhten Jahresbeitrag wird man denn auch bitter nötig haben, da der halbe Vereinsetat jetzt durch Zahlungen an „ehrenamtliche" Mitarbeiter verschlungen werden dürfte, so daß eine weitere Jahresbeitragserhöhung praktisch vorprogrammiert ist. Um dem bereits einsetzenden Strom der Austretenden zu begegnen, wurde im Dezember (Heft 8/1984, S. 7) verkündet: „Wer bis zum 31.12.84 seinen Austritt rückgängig macht, braucht keine neue Aufnahmegebühr zu zahlen."
Da die Firma Apple die Existenz der AUGE („Kopierverein") völlig ignoriert und nunmehr sogar einen eigenen Apple-Club („Schweigeverein") plant, geht die AUGE jetzt im User-Magazin aufs Ganze: Manfred Schürmann alias Key B. Hacker alias Macintosh-Frustrierter löste Wolfgang Gohe als farblosen Leiter der Macintosh-Arbeitsgemeinschaft ab und avancierte zum Protagonisten eines knallharten Kurses (s. User-Magazin, Heft 2/1985, S. 5ff.). Während „Apple's unabhängiges Magazin" als „Potenzierung von Seichtheiten der Vorbeterindustrie" abgestempelt wird, bleibt unser „Peeker" einstweilen ungeschoren, obwohl oder gerade weil Schürmann in allen denkbaren Kombinationen „peekt" („anpeeken", „erpeeken" usw.). In seinem furiosen Aufsatz „Lieber der Fall als der Kniefall" versteigt er sich in Formulierungen, wie man sie bei Apple in München bislang noch nicht vernommen haben dürfte: „Profit, Profit und nochmals Profit. Und den macht Apple reichlich, um sich den Schaum zu leisten, mit dem immer wieder neue Kunden eingeseift werden ... Außer den stolzen Gewinnen verbirgt Apple so gut wie alles, was jede ordentliche Firma sonst an Informationen auf den Tisch legt ... Wer den Mund so voll nimmt, muß auch was drin haben außer warmer Luft". In seinem Diskurs hebt Schürmann immer wieder darauf ab, daß sich die Apple-Niederlassung in München völlig eingeigelt hat und jeden Kontakt mit der Außenwelt panisch meidet. Wird sich Apple-München durch solche Kritik ändern lassen? Wohl kaum, denn Apple-München ist lediglich der verlängerte Arm von Apple-Cupertino, und wenn Cupertino in Verkennung des spezifischen, bundesdeutschen Mikrocomputermarktes „No, no" sagt, dann kann München nicht „Doch, doch" sagen. Wie bei der AUGE ist auch bei Apple-München das Phänomen der Personalfluktuation (seit Anbeginn des „Peeker" haben sich drei Pressesprecherinnen die Klinke in die Hand gedrückt) nur Wirkung und nicht Ursache eines viel tiefer liegenden Management-Konflikts, der sich aus *amerikanischer* Sicht etwa wie folgt darstellt:
– In der Anfangszeit der amerikanischen Apple-Firma („Wozniak-Jobs-Ära") war man auf Apple-Clubs, Freaks und peeker-ähnliche Zeitschriften („Call Apple", „Nibble" usw.) angewiesen, da das eigene technische Knowhow aus personellen Gründen begrenzt war. Der kometenhafte Apple-Aufstieg ist letztlich durch peekende Clubs und Freaks bedingt worden.
– Dann kam eine Übergangsphase („Jobs-Wozniak-Ära"), in der die Clubs und Freaks gerade noch geduldet wurden.
– Schließlich trat man in die heutige amerikanische Phase („Sculley-Jobs-Ära"), die durch die Parole „Keine Infos an Clubs, Freaks und kritische Zeitschriften" gekennzeichnet ist.
Diese Entwicklung hat aus der Sicht der *amerikanischen* Marketing-Strategie ihre Berechtigung, weil die amerikanische Mutterfirma über das technische Knowhow verfügt. Aus *deutscher* Sicht sieht die Lage jedoch ganz anders aus:
– Erstens hinkt die deutsche Entwicklung der amerikanischen um mindestens zwei Jahre hinterher, und zwar hinsichtlich des Mikrocomputermarktes im allgemeinen sowie hinsichtlich der Adaptation amerikanischer Produkte im besonderen. Ferner hinkt auch die

deutsche Computermentalität der amerikanischen hinterher. Diese Mentalitätsunterschiede erklären auch, warum beispielsweise der Mecki in den USA sehr gut und in Deutschland sehr schlecht verkauft wird. Deutsche assoziieren EDV immer noch mit Effizienz und Schnelligkeit, während Amerikaner inzwischen zusätzlich das Spielerische und Ästhetische in den Vordergrund rücken. Angesichts einer Arbeitslosigkeit von über 10 % wird es zur Zeit nur wenigen Unternehmern in den Sinn kommen, den eigenen Mitarbeitern ein Schnickschnack-Gerät in der Art des Mecki zur Verfügung zu stellen, denn die Maus, die Grafiken und die diversen Schriftsorten würden zu Spielereien verlocken, für die die Arbeitsmentalität in deutschen Landen kaum Raum läßt.

– Zweitens verfügt Apple-München als reine *Marketing*-Firma über kein Knowhow, wie man es bei der amerikanischen Muttergesellschaft gewohnt ist. Oder kennen Sie eine Hardware oder Software, die in München entwickelt worden ist? Wenn man wie in den USA jetzt auch in Deutschland die Clubs und Freaks wie eine ausgequetschte Zitrone wegwirft, so entsteht ein technisches Informationsvakuum, das die Hilfesuchenden im Regen stehen läßt. So gesehen kann der Apple-Besitzer zwar auf Apple-München, nicht jedoch auf peeker-ähnliche Informationsquellen verzichten. Dies hat man sicherlich in München erkannt. Da jedoch die Direktiven aus Cupertino anders lauten, igelt man sich ein, damit man ja nichts Falsches sagt gemäß der Maxime „Si tacuisses philosophus mansisses".

Angesichts der Kursänderung der AUGE wird man mit Spannung die weitere Entwicklung nach der Wahl des neuen Vorsitzenden auf der bevorstehenden Jahreshauptversammlung im März verfolgen dürfen. Bleibt nur zu hoffen, daß man die exzessive Vereinsmeierei aufgibt und zu den ursprünglich gesteckten Zielen wieder zurückkehrt, denn Apple-München wird den Verein erst dann ernst nehmen, wenn man geschlossen und notfalls auch hart im Sinne des Schürmann-Kursus auftritt.
Ulrich Stiehl

Ergänzungen zu „ProDOS für Aufsteiger", Bd. 1

von Arne Schäpers

(Herr Schäpers, ein exzellenter ProDOS-Kenner, hat mein ProDOS-Buch kritisch unter die Lupe genommen und dabei einiges richtiggestellt, das mir durch die „Lappen" gegangen ist. us)

Seite 19 – Warum ist BLOAD (READ) erheblich schneller als BSAVE (WRITE)?
WRITE arbeitet Byte per Byte (Puffer kopieren), READ dagegen in drei Stufen:
Stufe 1: Byte per Byte (Pufferkopie), bei gesetztem NEWLINE bleibt es auch dabei. Stufe 2: Block Read: Der Datenteil des File-Puffers wird auf die Zieladresse gesetzt, und SET MARK liest unter Umgehung des File-Puffers direkt in den Zielbereich. Stufe 3: Wenn nach Stufe 1 und 2 immer noch ganze Blocks übrig sind, wird der Indexblock direkt ausgewertet und von READ aus der Disk-Driver mit entsprechender Zieladresse aufgerufen. Endet der READ nicht auf einer Blockgrenze, werden die letzten Bytes wieder Byte per Byte aus dem File-Puffer kopiert. Eine sehr durchdachte Routine.

Seite 27 – Liest Bootprogramm auch Sektor 1?
Das Boot-Programm liest Track 0, Sektor 0, danach Sektor 2 (physikalische Sektornummer). Der Block 1 wird nur für SOS gelesen.

Seite 32 – Was bedeuten Seek und Format beim Disk-Driver?
SEEK ist für CMD = 00 nicht implementiert, Arm-Bewegungen finden keine statt. Im MLI wird vorher noch Block 00 00 gesetzt, um ein eventuelles Aussteigen über Fehler-Nr. 27 (hier: illegale Blocknummer) zu vermeiden.
CMD = 03: FORMAT ist nur für die RAM-Disk implementiert. (Neue Erkenntnis: Wenn das sog. Init-Flag auf 0 gesetzt ist – nach dem Booten automatisch der Fall –, wird vor dem ersten RAM-Disk-Zugriff der Directory- und Volume-Bit-Map-Block „initialisiert". Setzt man durch eine kleine Poke-Routine das Init-Flag auf 255, so bleibt nach dem Neubooten der RAM-Disk-Inhalt unversehrt. Ein entsprechender Patch, der allerdings versionsabhängig ist, wird im nächsten „Peeker" veröffentlicht. us)
Seite 57 – Was bedeutet Label Präfix ($BF9A) ungleich 0?
Dies ist der Startindex des momentan gesetzten Präfixes in der Seite $F100. Bereich $C0–$FE.

Seite 63 – Korrekturen zu den internen Puffern (gilt nur für 1.0.1):
$EF25–$EFFF: Tabellen
$F000–$F0FF: Variablen
$F100–$F1EF: Neuer Pathname
$F1C0–$F1FF: Präfix
$F200: Volume Control Block
$F300: File Control Block
$F400–F5FF: Volume Bit Map
$F600–$F7FF: Druckfehler: Hier steht zum zweiten Mal $F4000–$F5FF. Außerdem ist der Begriff Catalog-Block mißverständlich, da das BASIC.SYSTEM ein OPEN des Directory mit eigenem File-Puffer und anschließendem READ durchführt. Folge: In $F600–$F7FF befindet sich zunächst der Hauptzeiger des Directory.

Seite 65 – Warum „Drive too fast"?
Die Fehlermeldung kommt daher, daß der Formatter anstelle von $DE $AA $EB die Bytes $DE $AA $DE $EB als Address Field Trailer schreibt. Das zweite $DE wird nirgendwo überprüft, nimmt aber pro Track 16 Bytes mehr Platz weg. Für künftige Zwecke reserviert?

Seite 105 – Warum nur 32M-Volumes und 16M-Files?
Max. Volume-Größe gesetzt durch 16-Bit-Blocknummer: 256 · 256 · 512 Bytes. Max. File-Größe gesetzt durch 24-Bit-Position im File: 256 · 256 · 256 Bytes. (Daß ich dies nicht begründete, war ein „totaler Blackout". us)

Seite 107 – Präfix mit Schrägstrich am Ende?
Dieser muß nicht definiert werden und wird auch nicht gespeichert. GET PREFIX hängt diesen Schrägstrich explizit an den ausgegebenen String an.

Seite 120 – Was macht DELETE im einzelnen?
DELETE (DESTROY) benutzt SET EOF, woraus sich die Begründung für das Löschen der Indexe ergibt. Allerdings wird immer nur der erste Block (Typ 3: Masterblock und erster Indexblock, Typ 2: Indexblock) auf der Diskette überschrieben. Folge: Von einer Datei des Typs 3 bleibt der zweite Indexblock auf der Diskette voll erhalten. Ob das etwas für einen UNDELETE-Befehl nützt, ist eine andere Frage.

Seite 132 – Was prüft „HUSTON"?
Byte $14 enthält als Standard die Zahl $75, „HUSTON" steht auf den relativen Bytes $15–$1B, nicht auf $14–$1B. Beim Lesen eines Subdirectory-Key-Blocks wird dieses Byte ($14) getestet. Die Zahl selber ist dabei unwichtig, es müssen 5 Bits gesetzt sein, ansonsten INCOMPATIBLE FILE FORMAT ($D885).

Seite 150 – Spezielle System-Bit-Map-Konflikte
Wenn ProDOS Daten in den Parameterblock zurückliefert und der Parameterblock dabei auf einen Puffer zeigt (GET PREFIX/ONLINE etc.), wird der Puffer mit der System-Bit-Map verglichen. Routinen mit einem Puffer in der Seite 3 funktionieren deshalb nicht. Wenn der Puffer nicht auf einer Seitengrenze anfängt, überprüft die Checkroutine nicht $200, sondern $300 Bytes.

ProDOS-Bücher und -Aufsätze

Wenn Sie sich gewundert haben, daß von der Serie „ProDOS für Anfänger" bislang nur der 1. Teil (im Peeker, 2/1984) erschienen war, so hat dies einen besonderen Grund: Wir möchten Ihnen stets aktuelle und detaillierte Informationen bieten. Dies ist uns jedoch bei ProDOS nur mit Mühe möglich, weil Apple allein im letzten Jahr

mindestens 3 Versionen – bekannt sind uns 1.0.1, 1.0.2, 1.1.1 – herausgebracht hat, die nicht völlig funktionsgleich sind. Bei einer oberflächlichen ProDOS-Darstellung in „Mecki-Manier" – viele Bildchen und wenige Worte – könnte man die Feinheiten ignorieren. Dies geht jedoch nicht an, wenn handfeste professionelle Hilfsprogramme veröffentlicht werden sollen. Einige Beispiele: Bei der Version 1.1.1 beginnt das Machine Language Interface MLI bei $DE00 und der Disk-Driver bei $D000, während sich noch bei der Version 1.0.1 das MLI bei $D000 und der Disk-Driver bei $F800 befand. Version 1.1.1 hat den neuen Befehl „BYE" (Reboot-Routine), der bei 1.0.1 und 1.0.2 fehlt. Der Uhr-Driver ist jetzt wieder verlegt worden usw. Der Grundsatz „Think first, program later" wird augenscheinlich ignoriert. Dies verrät – gelinde ausgedrückt – Konzeptionslosigkeit, es sei denn, daß die Irritation intendiert ist, um den Fremdfirmen (Beispiel: Erphi-Controller mit ProDOS-Driver-Patches) das Leben schwer zu machen.

Alle bisherigen „Peeker"-ProDOS-Utilities (QUICKCOPY, PRODOS.PATCH) beziehen sich auf die Version 1.0.1 und laufen *nicht* auf den späteren Versionen. Wir werden zukünftig die jeweils gültige ProDOS-Version deutlich vermerken und im übrigen unsere Utilities an neuere Versionen anpassen, wobei wir allerdings einige Zwischenversionen überspringen werden, da diese offenbar mehr „Übungscharakter" haben.

Zu ProDOS gibt es folgende Bücher:

a) *„ProDOS Technical Reference Manual"* (Apple): Sehr gründliche, aber zugleich sehr theoretische Beschreibung der MLI-Befehle (mit nur einem einzigen Anwendungsbeispiel!). Der Disk-Driver und das BASIC.SYSTEM werden nicht behandelt. ProDOS-Interna fehlen. Zielgruppe: Systemprogrammierer.

b) *„Beneath Apple ProDOS"* (Quality Software): Stark an (a) angelehnte Darstellung; zusätzlich eine Reihe von Anwendungsbeispielen sowie Patches (inzwischen teils überholt). Zielgruppe: Assemblerprogrammierer.

c) *„ProDOS für Aufsteiger, Band 1"* (Hüthig-Verlag): Wenig an (a) angelehnte Darstellung der internen und externen Speicherorganisation mit zahlreichen Block-Dumps und MLI-Beispielen, die auch unter Version 1.1.1 laufen. Die ProDOS-Global-Page ist wie bei (a) und (b) teils überholt. Zielgruppe: Assemblerprogrammierer.

d) *„ProDOS für Aufsteiger, Band 2"* (Hüthig-Verlag): In Vorbereitung befindliche Darstellung der BASIC.SYSTEM-Befehle. Zielgruppe: Applesoft-Programmierer. Aus Zeitgründen werde ich zunächst die Serie „ProDOS für Anfänger" im Peeker veröffentlichen und dann eine aktualisierte Zusammenfassung der Serie – bereichert um größere ProDOS-Utilities – als 2. Band von „ProDOS für Aufsteiger" herausbringen.

e) *„Die ProDOS-Analyse"* (Hüthig-Verlag): In Vorbereitung befindliche Beschreibung der ProDOS-Interna in Form eines sehr ausführlichen Disassembler-Kommentars. Zielgruppe: Assembler-Programmierer.

Ulrich Stiehl

INALL

INPUT für DOS/ProDOS

Die nachfolgende Utility simuliert den Input-Befehl (Eingabe von Tastatur/Diskette) und läßt alle Zeichen zu (Komma, Doppelpunkt usw.). Anstelle von „INPUT X$" verwendet man jetzt „CALL 768, X$". Der Variablename ist beliebig, doch muß es sich um eine Stringvariable handeln. Ein ausführlicher Kommentar folgt in einer der nächsten Peeker-Ausgaben.

```
10 FOR X = 768 TO 793: READ Y: POKE X, Y: NEXT
20 DATA 32,6,277,32,190,222,32,227,223,32,108,221,133,133,
   132,134,32,44,213,200,32,233,277,76,154,218:
   REM von H. Grumser/1985
30 CALL 768, X$: REM Beispiel
```

Apple-Bücher im Preisvergleich

Die Zahl der Bücher speziell über den Apple II wird immer größer. Neben einer Vielzahl amerikanischer Titel kann der Leser inzwischen auf eine große Auswahl deutscher Werke zurückgreifen. Mit der wachsenden Titelzahl wächst aber auch das Preisbewußtsein der Käufer. Die nachfolgende Aufstellung umfaßt deshalb eine nach dem Quotienten Preis/Seitenzahl aufsteigend sortierte Titelliste, die zeigt, daß man zwischen DM 0,07 bis DM 0,90 pro Seite bezahlen muß. Bei deutschen Verlagen unterliegen Bücher in der Regel der sog. „Preisbildung der zweiten Hand", d. h. die Endabnehmerpreise sind gebunden (nach GWB § 16 zulässig); „Feilschen" wäre also zwecklos. Ausländische Titel können demgegenüber von Importeuren frei kalkuliert werden.

P/S	Buchtitel	Seiten	Preis
0.07	Z8000 Assembly Language Programming	928	69.00
0.07	Programmierung des Z80	606	48.00
0.08	6502 Programmieren in Assembler	700	59.00
0.09	Einführung in Pascal	517	48.00
0.09	The 8086 Book	624	58.00
0.09	Computer Lexikon	317	29.80
0.09	6800 Programmieren in Assembler	512	49.00
0.10	Computer Dictionary	624	64.00
0.10	Z8000 Aufbau und Anwendung	464	49.00
0.10	8080/8085 Programmieren in Assembler	463	49.00
0.10	Microcomputer Applications in the Classroom	460	49.00
0.10	Apple II Basic Handbuch	300	32.00
0.11	Das Pascal Handbuch	530	59.00
0.11	Apple Graphics: Activities Handbook	422	47.00
0.11	Microcomputer Dictionary	606	68.00
0.11	Grundkurs in Pascal, Band 1	220	24.80
0.11	6809 Assembly Language Programming	576	65.00
0.11	Microprozessor-Interface Technik	425	48.00
0.11	File and Data Base Techniques	562	65.00
0.11	Principles of Programming Languages	544	64.00
0.12	Apple II Tips + Tricks	405	49.00
0.12	Programmierung des 6502	360	44.00
0.12	Vom Umgang mit CP/M	376	48.00
0.12	Einführungskurs: Apple II, II+, IIe	297	38.00
0.13	Applesoft Training	300	39.00
0.13	CP/M Anwenderhandbuch	303	39.80
0.13	The Apple Program Factory	331	43.80
0.13	Apple Basic	364	49.00
0.13	Introductory Experiments to 8080A Vol.1	494	68.00
0.13	Z-80 Programming & Interfacing Vol.2	494	68.00
0.13	Apple DOS 3.3 Tips & Tricks	203	28.00
0.13	Apple ProDOS für Aufsteiger	202	28.00
0.13	6502 Anwendungen	274	38.00
0.14	Apple II Anwenderhandbuch	400	56.00
0.14	Apple Pascal	420	59.00
0.14	Spiele für den Apple	270	38.00
0.14	CP/M-Handbuch	310	44.00
0.14	8080A Interfacing & Programming	507	72.00
0.14	The Creative Apple	450	64.00
0.14	Apple Files	414	59.00
0.14	The Book of Apple Software	490	69.90
0.14	Basic-Computer-Spiele 1	224	32.00
0.14	Basic-Computer-Spiele 2	224	32.00
0.14	Introductory Experiments to 8080A Vol.2	412	59.00
0.14	Apple Software Wegweiser	288	42.00
0.14	Apple II leichtgemacht	192	28.00
0.14	Basic Computer Programs for Business Vol.2	376	56.00
0.14	8080/85 Software Design 2	348	52.00
0.14	Apple Assembler	227	34.00
0.14	Computer Dictionary & Handbook	928	139.00
0.15	Apple Basic Übungen	252	38.00
0.15	Apple Pascal Games	376	58.00
0.15	Enhancing your Apple II Vol.1	376	58.00
0.15	An Apple in the Classroom	175	27.00
0.15	How to Program & Interface the 6800	414	64.00
0.15	Pascal Programme für Wissenschaftler & Ing.	375	58.00

KURZBERICHTE

P/S	Buchtitel	Seiten	Preis
0.15	The Programmers CP/M Handbook	500	77.50
0.15	8080/85 Software Design 1	333	52.00
0.15	Programming the 6809	362	57.00
0.15	Basic für den Kaufmann	240	38.00
0.16	Apple II Basic-Wegweiser	200	32.00
0.16	Apple Hardware Wegweiser	260	42.00
0.16	Basic-80 und CP/M	296	48.00
0.16	Microcomputer Graphics	300	49.00
0.16	Apple II Raster Grafik	300	49.00
0.16	Logo: Grafik,Sprache,Mathematik	257	42.00
0.16	Schul-Software-Katalog	110	18.00
0.16	Programming & Interfacing the 6502	414	68.00
0.16	Basic-Programme:Mathrmatik,Statistik,Inf.	347	58.00
0.16	Advanced 6502 Programming	292	49.00
0.16	6801,6871 & 6803 Interfacing	349	59.00
0.17	Wordstar für die Praxis	316	54.00
X0.17	Apple II Assembler-Programmierung	280	48.00
0.17	Beat the Odds	210	36.00
0.17	Z-80 Programming & Interfacing Vol.1	302	52.00
0.17	Erfolg mit Visicalc	218	38.00
0.17	Logo im Mathematikunterricht	250	44.00
0.17	An Introduction to engineered Software	322	57.00
0.17	The Apple in your Hand	220	39.00
0.17	Lehrspielzeug Computer:Apple	139	24.80
0.17	8255 Interfacing	217	39.00
0.18	Thinking about TLC Logo	236	43.00
0.18	Einführung in Wordstar	208	38.00
0.18	Programme für meinen Apple II	186	34.00
0.18	8085A Cookbook	350	64.00
0.18	CP/M Bible	429	79.00
0.18	Z-80 Microcomputer Handbook	304	56.00
0.18	Computer,Teaching & Learning	257	48.00
0.18	The Apple Connection	262	49.00
0.18	C-Programmierung	297	56.00
0.18	TEA an 8080/85 Editor/Assembler	254	48.00
0.18	Advanced Pascal Programming Techniques	370	70.00
0.18	Basic Computer Programs for Business Vol.1	264	50.00
0.19	Wirtschaft auf dem Apple II/IIe	200	38.00
0.19	Planen und Entscheiden mit Basic	200	38.00
0.19	Unix Primer Plus	414	79.00
0.19	Mathematik auf dem Apple	219	42.00
0.19	CP/M User Guide,Third Edition	327	63.30
0.19	Erfolg mit Multiplan	196	38.00
0.19	Basic Computer Programs in Science and Eng.	247	48.00
0.19	Soul of CP/M	391	76.00
0.19	Apple Basic:Data File Programming	303	59.00
0.19	An Apple for Kids	200	39.00
0.19	77 Basic-Programme	200	39.00
0.19	Apple II Technik & Wissen	250	49.00
0.19	The Visicalc Book,Apple Edition	301	59.00
0.19	Apple II Assembly Language	330	64.90
0.19	Z80 Software Gourmet Guide & Cookbook	324	64.00
0.19	Pascal for the Apple	496	98.00
0.19	Programming the Z8000	298	59.00
0.20	6502 Software Design	269	54.00
0.20	All About DOS	288	58.00
0.20	MS-DOS and PC-DOS,User's Guide	266	54.00
0.20	Using the 6800 Microprozessor	176	36.00
0.20	CP/M und Wordstar	144	29.80
0.20	Apple Basic for Business	301	63.00
0.20	Intermediate-Level Apple II Handbook	324	68.00
0.21	Apple Pascal Grafik	230	49.00
0.21	Running MS-DOS	350	75.00
0.21	Programming the Apple II & IIe	452	98.00
0.21	DOS Handbuch	184	39.90
0.21	Apple Interfacing	206	44.90
0.21	Games Apples play	270	59.00
0.21	6809 Programming & Interfacing	270	59.00
0.21	Basic aus der Praxis	183	40.00
0.22	Pascal Programs in Science and Engineering	339	75.00
0.22	8080 Software Gourmet Guide & Cookbook	234	52.00
0.22	Apple Programming for Learning & Teaching	305	68.00
0.22	Programmer's Guide to the 1802	156	35.00
0.22	Planen und Kalkulieren mit Multiplan	256	58.00
0.22	Microsoft Fortran	344	78.00
0.22	Supercalc im Einsatz	246	56.00
0.22	Z-80 Advanced Interfacing	347	79.00
0.22	6800 Software Gourmet Guide & Cookbook	228	52.00
0.22	Guide to your Apple III	276	63.30
0.23	Einführung in Logo	182	42.00
0.23	Fortran Programs for Scientists & Engineers	280	65.00
0.23	Apple II Applications	236	55.00
0.23	Numerical Basic	222	52.00
0.23	Hardware Interfacing with the Apple II plus	238	56.00
K0.23	Apple Maschinensprache	208	49.00

P/S	Buchtitel	Seiten	Preis
0.23	Learning Logo on the Apple II	250	59.00
0.23	Your Apple II needs you	336	79.90
0.23	Planen und Kalkulieren mit Visicalc	134	32.00
0.23	Das Datenbanksystem dBase II	284	68.00
0.23	Mostly Basic Vol.2	217	52.00
0.24	Golden Flutes and Great Escapes	200	48.00
0.24	The Academic Apple	162	39.00
0.24	Cobol for Beginners	369	89.00
0.24	Einführung in Datenbanksysteme mit dBase	280	68.00
0.24	Integer Basic Handbuch	122	29.90
0.25	Apple Fortran	236	59.00
0.25	How to make Money with your Microcomputer	256	64.00
0.25	Pascal Programming for the Apple	234	59.00
0.25	dBase: Eine Einführung	221	56.00
0.25	6502 Software Gourmet Guide & Cookbook	204	52.00
0.26	Apple Basic	183	48.00
0.26	Reference Manual IIe	266	69.90
0.26	Einführung in Forth	218	58.00
0.26	Using 6502 Assembly Language	300	79.90
0.26	How tp build a Program	352	94.00
0.27	How to use the Apple II and IIe	100	27.00
0.27	Science and Engineering Programs	220	59.90
0.27	8088 Assembler Programming for the IBM PC	235	64.00
0.27	All About Pascal	183	49.90
0.27	Multiplan richtig einsetzen	212	58.00
0.27	Advanced 6502 Interfacing	190	52.00
0.27	32 Basic Programs for the Apple	284	78.00
0.27	Apple Pascal	247	68.00
0.27	Forth-Programming	246	68.00
0.27	Handbook of Applesoft Basic II+/e	321	89.00
0.28	Z-80 Microcomputer Design Projects	208	59.00
0.28	The compleat Apple CP/M	233	67.00
0.29	Basic Apple IIe:A programming Guide	300	87.00
0.29	Inside concurrent CPM 86	300	87.00
0.29	Wörterbuch der Computerei	110	32.00
0.29	Kids to Kids	168	49.00
0.29	CP/M Database Management Systems	320	94.00
0.29	All About Applesoft	135	39.90
0.29	dBase II richtig eingesetzt	230	68.00
0.30	Pascal Programs for Business	212	64.00
0.30	Interface Projects for the Apple II	172	52.00
0.30	Basic-Dialekte im Vergleich	105	32.00X
0.30	Datenübertragung und Datenaustausch	155	48.00
0.31	CP/M Revealed	180	56.00
0.31	Microprozessor Circuits Vol.2	123	39.00
0.31	Apple II Word Processing	248	79.00
0.32	Computer Graphics Primer	184	59.00
0.32	Multiplan Deutsch	115	37.00
0.32	Practical Basic Programs	180	58.00
0.32	Picture Perfect	234	76.00
0.32	Mostly Basic Vol.1	158	52.00
0.33	Pascal Primer	206	68.00
0.33	Structural Problem Solving with Pascal	386	129.00
0.34	Graphically Speaking	170	58.00
0.35	The Basic Conversions Handbook	80	28.00
0.35	Apple II Schaltpläne	180	64.00
0.35	Microprozessor Circuits Vol.1	109	39.00
0.36	Apple Graphics Games	218	79.00
0.36	What's Where in the Apple	248	89.90
0.36	Wordstar Befehlsübersicht	81	29.80
0.37	The Apple Almanac	240	89.90
0.37	Mail Merge	98	37.00
0.38	Computer Playground	128	49.00
0.38	Intro to Arithm for Dig Sys Designers	308	118.00
0.38	The Turtle's Source-Book	226	88.00
0.39	Beneath Apple DOS	170	67.00
0.40	Graphics Cookbook for the Apple	71	29.00
0.41	Discover Forth	145	59.90
0.41	The Custom Apple	190	79.00
0.41	Unix Programmers Manual	425	177.00
0.42	Apple II Computer Graphics	188	79.00
0.43	Data Plotting Software for Micros	248	109.00
0.44	Microsoft Cobol	168	74.00
0.44	Animation,Games and Sound for Apple II/IIe	178	79.90
0.45	Wordstar	82	37.00
0.46	Circuit Design Programs for the Apple II	130	59.90
0.47	Graphic Software for Microcomputer	184	88.00
X0.50	Apple II ROM Listing	117	59.00
0.50	Visicalc mit Disk	153	78.00
0.53	CP/M Primer	92	49.00
0.54	A bit of Logo Magic	97	53.00
0.59	Apple II-6502 Ass.Lang.Tutor	234	139.00
0.80	Apple II DiskGuide	36	29.00
1.60	Author Kit	177	285.00
1.90	Visicalc DiskGuide	14	27.70

Leserbriefe

Hardware-Beiträge
Mit dieser Postkarte möchte ich Ihnen positive Resonanz verschaffen: Die von Ihnen verfaßten Bücher Apple Assembler und Apple DOS sind eine Fundgrube wertvoller Informationen zu einem ungewohnt moderaten Preis; gleiches Lob für die Zeitschrift Peeker, wobei einige „Hardware-Beiträge" (Interfacekarten und angewandtes Interfacing) sehr gut hineinpassen würden. Ansonsten: weiter so und vielen Dank! *Jürgen Treumer, München*

If...Then...Else (Peeker 1/84)
Die sehr effektive „Block-If"-Struktur, die z.B. Fortran 77 bietet, kann auch in Applesoft-Basic leicht implementiert werden. Dazu muß im wesentlichen nur der logische Vergleichsausdruck invertiert werden.
Fortran 77:
IF exp 1 THEN
...
Else IF exp2 THEN
...
ELSE
...
END IF
Basic:
aaa IF NOT exp1 THEN GOTO bbb
...
GOTO ddd
bbb IF NOT exp2 THEN GOTO ccc
...
GOTO ddd
ccc REM „entspricht ELSE"
...
ddd REM „Sammelpunkt"
Es dürfen beliebig viele „Else IF"-Blöcke (oder auch gar keine) auftreten. Der „ELSE"-Block bildet, falls vorhanden, den Abschluß. Das „NOT" im Basic-Text kann in den meisten Fällen durch einen „umgedrehten" Vergleichsoperator ersetzt werden (aber bitte <= statt > usw.). Eine Ausnahme bilden komplizierte logische Ausdrücke, die durch Umschreiben noch unübersichtlicher würden (und deshalb sowieso vermieden werden sollten).
Die Punkte entsprechen den Befehlen der einzelnen Verzweigungen. Auch hier dürfen natürlich wieder Abfragen auftreten (Verschachtelung).
Fortran erlaubt zwar einen beliebigen Aussprung aus dem Block-If, einen Einsprung aber nur via „IF exp1 THEN". Da in Basic die gesamte Struktur nur programmtechnisch existiert, ist natürlich der Einsprung an beliebiger Stelle möglich. Wie sinnvoll dies jedoch ist, muß jeder selbst entscheiden. Es entsteht auf diese Weise sehr leicht ein undurchdringlicher Spaghetti-Code.
Albert Markl, Ismaning

RAM-Disk-Driver
Zunächst möchte ich Ihnen zu der wirklich gelungenen Konzeption des „Peeker" gratulieren. Eine Zeitschrift, die sich an den Bedürfnissen des fortgeschrittenen Apple-Programmierers orientiert und nicht nur auf einigen Seiten „Tips & Tricks" bringt, fehlte meines Erachtens schon lange auf dem Zeitschriftenmarkt. Neben Ihrer Serie über „Double Lores"- und „Double Hires"-Grafiken war ich besonders über den RAM-Disk-Driver für die Language-Card (Heft 1/2 '85) begeistert. Als Anregung möchte ich an dieser Stelle folgendes nennen: Damit die in der RAM-Disk abgelegten Files schnell auf einer Floppy abgelegt werden können, wäre eine kurze, von BASIC aufrufbare Utility wünschenswert, die den Inhalt der RAM-Disk auf Diskette kopiert.
Alle, die wie ich einen Apple II Plus ohne Kleinschrift besitzen, haben sich sicherlich über die merkwürdigen Zeichen in der Frage „Ramdisk mit Init J/N ?" des Drivers gewundert. Kleinbuchstaben werden vom „Großschrift-Apple" bekanntlich als Sonderzeichen interpretiert. Ein kleiner Patch sorgt nach Laden des Drivers für Abhilfe: $6151:AØ CD C9 D4 AØ C9 CE C9 D4 AØ BF. Die Meldung erscheint nun in Großbuchstaben.
Georg Bang, Oldenburg

AUGE
Ich kann Ihnen für Ihren Peeker nur gratulieren. Das ist genau die Informationsquelle, die ernsthafte Apple-Hobbyprogrammierer gesucht haben, die man aber vergeblich in der Apple User Group sucht. Was ich besonders gut finde, ist die Tatsache, daß Ihre Programme auch funktionieren. Das liegt wohl daran, daß Sie, der Chefredakteur, den Apple in- und auswendig kennen, aus der Branche kommen, und die Sorgen und Nöte der Freaks aus eigener Erfahrung kennen. Bitte machen Sie nur weiter so. *Rudolf Rötering, Soest*

Assembler-Grundlagen
Als relativer Neuling im Programmieren des Apple IIe komme ich zwar mit dem Programmieren in Applesoft ganz gut zuwege; doch schaue ich in den „Peeker", so begegnen mir die Programmdarstellungen überwiegend in hexadezimaler und Assembler-Schreibweise. Nun habe ich mir gestern den Assembler-Kurs „mister micro" gekauft, der speziell für den Apple II/IIe geschrieben wurde (Sybex-Verlag) und dem eine Assembler-Diskette beigefügt ist; doch wenn ich glaubte, mit diesem Rüstzeug endlich den Zugang zu den inneren Geheimnissen in der Handhabung meiner Konfiguration gefunden zu haben, so sehe ich mich entäuscht. Vielleicht könnten Sie mir da einige Hinweise geben, wie ich mit Sicherheit zu dem ersehnten Durchblick komme, damit ich den „Peeker" in Zukunft auch mit Gewinn verstehen und nutzen kann. *Ernst Kaiser, Titisee-Neustadt*

Jubeltests
Erstmal herzlichen Glückwunsch zu dieser Zeitschrift. Wie ich hoffe, bleibt sie so interessant und frei von „Jubeltests", ansonsten finde ich ihre jetzige Mischung aus Software- und Hardware-Infos sehr gelungen. Bitte weiter so.
Ich besitze seit 1979 einen Apple II europlus und bin naturgemäß besonders an Beiträgen zu diesem Thema interessiert. Insbesondere gute Assemblerprogramme interessieren mich, oder wie wäre es mal mit einem Beitrag zu Star-Gemini 10X; dort wäre die Character-Download-Procedure sehr interessant. Auch Z80-Assembler wäre bestimmt ein ergiebiges Thema, somit gibt es bestimmt noch viel zu tun. *Wilfried Röll, Walldorf*

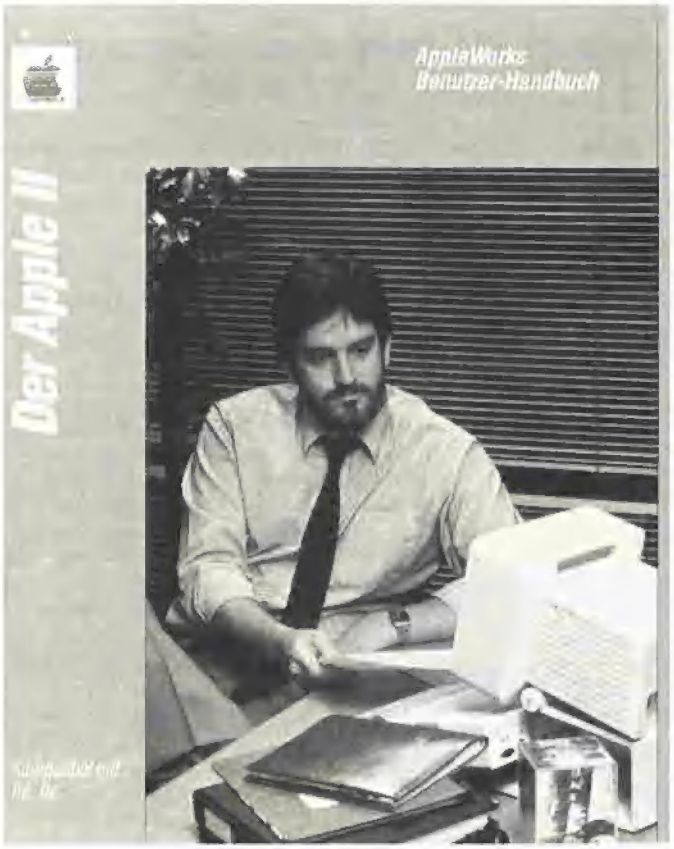
Glücklicher Mac-Besitzer
Als ich Ihr Editorial in Heft 2/84 gelesen hatte, griff ich spontan zur Maus, drehte den Bildschirm auf und möchte Ihnen hiermit mitteilen, daß ich mit meinem neuen Büromitarbeiter Fat Mac 512K bestens zufrieden bin. Er ist das beste für den Preis, was ich probiert habe. Ich bin auf jeden Fall mit meiner Wahl glücklich. Insbesondere ist für mich der Einsatz des Zeichenprogramms eine tolle Alternative zu den technischen Zeichnungen, die ich im Rahmen meiner Arbeit zu machen hatte und ich nunmehr mit MacPaint erledige (MacDraw habe ich noch nicht ausprobieren können).
Ebenso sind meine Damen über MacWrite glücklich und ich, da ich nunmehr auch einmal einen Text selbst tippen und ausdrucken kann. Und mit am besten finde ich die Übereinstimmung des Bildschirmbildes mit dem Schriftbild einer Briefseite.
Der einzige mir von meinen Damen genannte Nachteil der Tastatur: Sie spricht ein bißchen zu leicht an. Wenn jemand noch zwischendrein Schreibmaschine schreiben muß, so wird ein unterschiedlicher Anschlag der verschiedenen Tastaturen immer zu einer höheren Fehlerquote führen. Ich werde Apple vorschlagen, den Anschlag der Tastatur veränderlich zu gestalten. Und: Nennen Sie mir einen formschöneren Computer. *Rechtsanwalt Klaus Mierswa, Mannheim*

Schattenseiten
Als Leser Ihrer Bücher DOS 3.3 und ProDOS war ich, nachdem ich einen Apple-Fachmann als Chefredakteur einer Computerzeitschrift fand, neugierig geworden. Nach der Lektüre der Ausgabe 2/84 sehe ich mich „genötigt", eine weitere Zeitschrift zu beziehen. Zu Ihrer Frage im Leitartikel 2/84, ob denn bei Testbeschreibungen auch über Schattenseiten der jeweiligen Testobjekte geschrieben werden sollte, meine ich, daß gerade dieses eine unabhängige Zeitschrift (Redaktion) auszeichnet. Im übrigen möchte ich Sie ermutigen, so weiterzumachen. Halten Sie Verbindung zu Ihren Lesern. *Rolf Steinfelder, Hildesheim*

Mailmerger und Appleworks

getestet von Ulrich Stiehl

Die Firma INTUS Lern-Systeme AG, 7890 Waldshut-Tiengen, Kaiserstraße 21, Tel. 07751/7920, ist eigentlich auf Schulsoftware spezialisiert, doch hat sie auch einige allgemeine Anwenderprogramme veröffentlicht, z.B. den Mailmerger als Hilfsprogramm zu Appleworks.



Appleworks

Appleworks ist ein von Rupert Lissner (Autor von *Quickfile*) für die Firma Apple entwickeltes dreiteiliges Programmpaket für den privaten und geschäftlichen Bereich (vorwiegend Kleingewerbetreibende), das unter dem ProDOS-Betriebssystem läuft. Um es gleich vorweg zu sagen: Dieses Paket ist unter Berücksichtigung der charakterisierten Zielgruppe in jedem Fall sein Geld wert! Es setzt einen Apple IIc oder einen Apple IIe mit 64K-Karte voraus, läuft also nicht auf dem II Plus. Eine einfache 80-Zeichenkarte ohne 64K-Zusatzspeicher wäre ebenfalls möglich, doch wäre dann der zur Verfügung stehende Arbeitsspeicher viel zu gering. Appleworks wird mit einem knapp 350seitigen, zweifarbig gedruckten Handbuch in Spiralbindung sowie mit 2 Systemdisketten geliefert. Die sogenannte *Start*-Diskette dient zum Booten des ProDOS-Betriebssystems, wobei auch ein „Warmstart" mit „–PRODOS" o.ä. möglich ist. Denn man hat Appleworks nicht kopiergeschützt, damit es uneingeschränkt mit Festplatten (Profile usw.) sowie ggf. auch mit RAM-Disks benutzt werden kann. Die 64K-Karte selbst wird allerdings als Datenspeicher und nicht als RAM-Disk eingesetzt. Wenn man weder die Profile noch größere RAM-Karten besitzt, so empfehlen sich unbedingt 2 Laufwerke, da sonst Programm- und Datendiskette permanent gewechselt werden müßten. Nach dem „Hochfahren" des Betriebssystems (ProDOS Version 1.1.1) von der Start-Diskette muß man die sog. *Programm*-Diskette einlegen, die praktisch voll ist, weil das modular aufgebaute, eigentliche Appleworks-Programm über 200 Blocks einnimmt, die zu einem einzigen „Monster"-File zusammengefaßt wurden. Die diversen Module enthalten nicht nur die Assemblerteilprogramme, sondern auch die jeweils benötigten Menü-Texte.

Appleworks besticht durch den Benutzerkomfort und die Geschwindigkeit, mit der die einzelnen Funktionen (Sortieren, Suchen usw.) ausgeführt werden. Diese hohe Geschwindigkeit kommt jedoch nicht von ungefähr: Appleworks ist ein RAM-orientiertes Programmpaket, d.h. alles findet im Arbeitsspeicher („auf dem Schreibtisch") statt. Eine einzelne Datei oder die Summe aller Dateien darf im Falle einer 64K-Karte ca. 55K nicht überschreiten. Appleworks ist deshalb nur dann optimal, wenn entweder kleinere Datenmassen verwaltet oder größere Datenmassen sinnvoll aufgeteilt werden können.

Erwähnenswert ist die Flexibilität, mit der Fremddaten (ASCII-Files) in Appleworks überführt werden können. Nur so war es mir übrigens möglich, mit „schnell gestrickten" Programmen große Dateien zum Austesten von Appleworks zu erzeugen.

Modul 1: Dateiverwaltung: Dieses Modul, das eine Quickfile-Modifikation darstellt, dient zum Anlegen, Sortieren und Ausdrucken von Dateien aller Art, wobei auch „rechnende Felder" zulässig sind. Eine Datei könnte theoretisch 1350 Datensätze enthalten. Ein Datensatz ist die Summe der Felder (bei Adreßverwaltung: Vorname, Zuname, Straße usw.). Ein Datensatz könnte theoretisch 1024 Zeichen oder bis zu 30 Felder mit jeweils bis zu 76 Zeichen umfassen. Wegen der 55K-Kapazität können jedoch bei einer Adreßverwaltungsdatei, deren Datensätze (mit Briefanrede usw.) erfahrungsgemäß ca. 180 Zeichen groß sein müssen, nur ca. 300 Adressen verwaltet werden. (Zum Vergleich kann bei meinem „DB-Meister" auf 650 180-Zeichen-Datensätze zugegriffen werden.) Die Appleworks-„Datenbank" ist damit mehr im Privat- und weniger im Geschäftsbereich einsetzbar.

Modul 2: Textverarbeitung: Dieses Modul hat mir am besten gefallen. Mit einem 55K-Arbeitsspeicher – dies wären ca. 11 Peeker-Druckseiten oder ca. 25 Schreibmaschinenseiten – kann man sehr große Textmassen bearbeiten. Überraschenderweise ist die Speicherverwaltung besser als beim 47K-Applewriter gelöst, so daß man selbst bei vollem Speicher noch relativ schnell schreiben kann, was beim Applewriter IIe nicht mehr möglich ist. Funktionen wie Suchen und Ersetzen ähneln dem Applewriter. Analog zum Wordstar kann man einen rechten Rand einstellen, doch werden Silbentrennungen nicht angeboten. Wenn man darauf keinen Wert legt, kann man jetzt – im Gegensatz zum Applewriter – ein Dokument immer so sehen, wie es später gedruckt wird. (Der neue und in Deutschland noch nicht erhältliche ProDOS-Applewriter wird ähnliche Features aufweisen.)

Modul 3: Tabellenkalulation: Dieses Modul entspricht funktionell dem altbewährten *Visicalc*. Rein theoretisch könnten über 100.000 Zahlen- bzw. Rechenfelder, unterteilt in bis zu 127 Spalten und bis zu 999 Zeilen, eingerichtet werden. Wegen der 55K-Speicherbegrenzung liegt jedoch die praktische Obergrenze bei 5000-6000 Feldern. Visicalc-DIF-Dateien können von Appleworks übernommen werden.

Mailmerger

Obgleich Appleworks als integriertes Softwarepaket angepriesen wird, hapert es in praxi etwas mit der Integration. Beispielsweise können Adressen (aus Modul 1) nicht mit einem Schemabrief (aus Modul 2) verknüpft werden. Hier hilft der *Mailmerger*, der ursprünglich von der englischen Firma Amber Software entwickelt und von Peter Meyer aus dem Hause INTUS für deutsche Verhältnisse überarbeitet wurde. Der Mailmerger wird mit einer knapp 40seitigen, im Kleinoffset hergestellten Anleitung und 2 Programmdisketten geliefert. Um zu veranschaulichen, wie Mailmerger funktioniert, ein vereinfachtes Beispiel:

a) Mit dem Modul 1 von Appleworks (Dateiverwaltung) erstellt man wie bisher die Adreßdatei. Diese Datei muß jedoch für den Mailmerger in eine ASCII-Datei umgewandelt und auf einer Datendiskette gespeichert werden. Bei vollen Appleworks-Adreßdateien (55K) muß man praktisch immer eine zusätzliche ProDOS-Datendiskette bereithalten.
b) Mit dem Modul 2 von Appleworks (Textverarbeitung) erstellt man den Schemabrief, der auch „rechnende Felder" enthalten darf, in einer vorgeschriebenen Form, z.B.:

↑VORNAME
↑ZUNAME
↑STRASSE
↑PLZ
↑ORT
↑ANREDE
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑

↑VORNAME ↑ZUNAME
↑STRASSE
↑PLZ ↑ORT

↑ANREDE,

(ab hier Briefkörper)

Zunächst müssen also die zu verwendenden Felder mit dem „↑"-Zeichen definiert werden (Definitionskopf, der später selbst *nicht* ausgedruckt wird). Danach folgt der eigentliche Schemabrief mit den Felder-Platzhaltern. Der ganze Text muß auch hier aus Appleworks heraus in eine ASCII-Datei umgewandelt und auf einer Datendiskette gespeichert werden.
Nach dem Booten der Mailmerger-Diskette von Drive 1 – die ASCII-Datendiskette legt man zweckmäßigerweise in Drive 2 – braucht man dann nur noch Adreß- und Briefdateiname einzugeben, und schon kann der automatische Ausdruck beginnen. Die Bildschirm-Menüs erklären sich von selbst, so daß nach kurzer Benutzung die Anleitung nicht mehr benötigt wird. Der Mailmerger ist damit eine nützliche Ergänzung zu Appleworks für den Fall, daß Schemabriefe (Werbebriefe, Rechnungen, Mahnungen usw.) mit Adressen gemischt („gemergt") werden sollen.

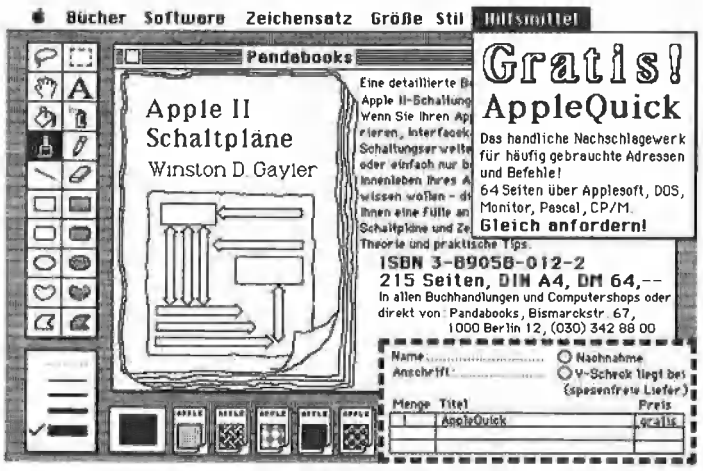
Einige Dinge gefallen mir jedoch nicht: Der Mailmerger ist partiell kopiergeschützt, wobei sich die geschützten Dateien „MM1.5G.T1.TXT" usw. („1.5G" steht für Version 1.5 German) bisweilen selbst nicht lesen können, worauf das Hauptprogramm-Modul erneut geladen wird. Außerdem ist im Gegensatz zu den ProDOS-Richtlinien slotabhängig programmiert worden (Slot 6, Drive 1), womit der Einsatz einer Profile oder RAM-Disk entfällt. Ob die Profile als *Daten*disk verwendet werden kann, konnte ich mangels Hardware nicht feststellen. Die Verwendung der 64K-RAM-Disk (/RAM, S3, D2) in dem Mailmerger-Modul „MM.A", Programmzeilen 20130 und 20240 ff. ist jedoch auf alle Fälle nicht möglich, da mit der dort enthaltenen fehlerhaften Directory-Read-Methode ein (verkürztes) RAM-Disk-Directory nicht ohne End-of-Data-Fehlermeldung gelesen werden kann.

Weiterhin finde ich es schwach, daß der Mailmerger ein Applesoft-Programm ist. Zwar sind einige Ampersand-Routinen an das Basic-Programm angehängt worden, doch ist der Mailmerger insgesamt zu langsam. Mit Hilfe der Balfer-Interface-Karte, die an anderer Stelle in diesem Peeker beschrieben ist, habe ich einen Druckerslot vorgetäuscht und konnte damit ermitteln, daß die interne Aufbereitung selbst bei nicht-rechnenden Briefen ca. 10s pro Brief dauert.

Damit können insbesondere schnelle Matrixdrucker vom Mailmerger nicht optimal bedient werden; dies gilt namentlich für kürzere Schemabriefe.

Schließlich mißfällt mir, daß der Mailmerger, obgleich er speziell für Appleworks gedacht ist, *nicht direkt* auf die ureigenen Appleworks-Dateien zugreifen kann. Durch die vorherige Umwandlung in ASCII-Dateien wird der Mailmerge-Vorgang unnötig umständlich und zeitraubend.
Zur Zeit ist der Mailmerger die einzige Alternative für Appleworks-Besitzer, die Schemabriefe verarbeiten wollen. Der RAM-Disk-Bug soll lt. Rücksprache in Kürze beseitigt werden.

Ulrich Stiehl



Erphi-Controller

Eine kompatible Erweiterung

getestet von H. Grumser

Die Firma erphi electronic GmbH vertreibt über den Fachhandel (also *nicht* direkt, weshalb man sich an Zwischenhändler wie Ueding usw. wenden muß) den Controller AFDC2, der den Anschluß von zwei Diskettenlaufwerken erlaubt, wobei außer zwei üblichen „Apple-Schnittstellen" (20polig) auch eine Standard-Schnittstelle (Shugart-Schnittstelle, 34polig) vorliegt. Somit können sowohl Apple-Laufwerke und Kompatible als auch Standard-Laufwerke betrieben werden.
Beim Auspacken des Controllers erlebt man die erste angenehme Überraschung. Die Karte zeichnet sich nicht, wie viele andere, durch kreuz und quer verteilte Bauteile mit nachträglich eingesetzten Kondensatoren aus, sondern durch höchste Professionalität. Der mitgelieferte Schaltplan zeigt die Einsicht, daß Qualität und Transparenz in engem Zusammenhang stehen.
Die Hardwareanpassung an verschiedene Laufwerke erfogt durch zwei auf der Karte befindlichen Schaltergruppen, mit denen Schnittstellentyp und Aufzeichnungsformat (Spurdichte, einseitig/doppelseitig, 35(70)/40(80) Spuren) vorgegeben werden. Dadurch ist es möglich, Floppy-Disk-Drives mit einer Kapazität bis zu 640K zu betreiben.
Die volle Leistungsfähigkeit des Controllers kann nur ausgeschöpft werden, wenn die verfügbaren Betriebssysteme entsprechend gepatcht und um einige Teile erweitert werden. Dies geschieht durch einen sogenannten Autopatch, der beim Booten des Rechners das Betriebssystem an den erforderlichen Stellen ändert. Der Vorteil dieses Verfahrens liegt darin, daß die Disketten weiterhin auf „normalen" Systemen bootfähig bleiben; das Image des Betriebssystems wird auf der Diskette nicht geändert.
Das Patch-Programm selbst, derzeit Version 5.0, ist Bestandteil der Controller-Firmware, die wegen ihres Umfangs nur noch in Slot 6 lauffähig ist. Dieses Programm ist in der Lage, DOS 3.3, Pascal 1.1 sowie CP/M 2.2 (56K CP/M von Digital Research) zu erkennen und zu modifizieren. Zum Test lag bereits Version 6/3 bzw. 6/5 für Diversi-DOS (2-C und 4-C), Pascal 1.2 (64K und 128K), CP/M 2.20 (B) (56K-Version von Microsoft) sowie ProDOS (1.0.1, 1.1 – gemeint 1.0.2? – und 1.1.1) vor. Man beachte, daß der Patch nur beim Booten erfolgt. Würde man z.B. ProDOS *erneut* mit BRUN PRODOS, A$2000, TSYS einlesen, so käme die Controller-Firmware nicht zum Einsatz und ProDOS würde (mit einem 80-Spur-Laufwerk) nicht mehr laufen.
Die speziellen Besonderheiten der einzelnen Betriebssysteme, insbesonders bezüglich der Verwaltung größerer Datenmengen, sind in dem mitgelieferten Handbuch, das auf über 50 Seiten sehr detailliert alle Hard- und Softwareprobleme erläutert, ausführlich beschrieben. Da nicht alle Initialisierungs- und Kopierprogramme, die im allgemeinen beim Erwerb eines Betriebssystems mitgeliefert werden, höhere Kapazitäten verwalten können, liefert die Firma erphi eine Multi-Disk, die entweder eigene Routinen enthält oder bereits vorhandene Systemroutinen entsprechend modifiziert.
Bei der Anpassung der Betriebssysteme an neue Hardware-Gegebenheiten hat man sich bei erphi die Kompatibilität zum alten System zur höchsten Maxime gewählt. So wurde z.B. beim DOS 3.3 die größere Datenmenge nicht durch eine trickreiche Erweiterung der Catalogspur und der VTOC erreicht, sondern durch die Einrichtung eines zweiten Catalogs, der virtuell über Drive 3 und 4 angesprochen wird (DOS 3.3 kann maximal 400K pro Volume verwalten). Soll der Apple ausschließlich mit Laufwerken höherer Kapazität betrieben werden, so stellt sich das Problem, käuflich erworbene Software, die fast immer auf 35-Spur-Disketten vertrieben wird, auf eine Diskette höherer Kapazität zu kopieren. Hierzu befindet sich auf der Multi-Disk eine Utility, die ein bestimmtes Laufwerk des erweiterten Floppy-Disk-Systems per Software zu einem 35-Spur-Drive erklärt.
Beim Test wurde ein 640K-Doppellaufwerk der Firma erphi benutzt, wobei unter allen Betriebssystemen die erhöhte Kapazität ohne Schwierigkeiten ansprechbar war. Der Vorteil zeigte sich besonders unter Pascal, da sich hier nun die Möglichkeit bietet, alle 4 System-Disketten auf einer einzigen unterzubringen.
Das Handbuch in Ringbuchform, das im übrigen auch einen Teil für Assemblerprogrammierer enthält, rundet den durchweg positiven Eindruck ab. Der Preis von ca. DM 350,– steht gänzlich in Relation zur Qualität dieses Produkts.

Erphi-Controller mit Doppellaufwerk

INTERTEXT

Internationale Textautomation mit dem Apple IIe

INTERTEXT ist ein von der Firma Sauer entwickeltes und auf dem Applewriter-TKS-System (TKS = Textkommandosprache; WPL = Word Processing Language) basierendes Programm für automatisierte Textverarbeitung mit dem Apple IIe. Das System wurde mir vor kurzem in der Peeker-Redaktion vorgeführt.

Je 4.000 Textbausteine in den Sprachen Deutsch, Englisch, Französisch und Spanisch umfassen den gesamten Bereich der Geschäftskorrespondenz und ermöglichen die Rationalisierung und Automatisierung des deutschen und internationalen Schriftverkehrs.

Die Verarbeitung der Bausteine erfolgt am Bildschirm durch Eingabe entsprechender Satz-Nummern (Codes). Die Einarbeitungszeit ist relativ kurz. Da die Bausteine in allen vier Sprachen identisch codiert sind, ist es möglich, Auslandskorrespondenz ohne perfekte Fremdsprachenkenntnisse zu schreiben. An jeder Stelle der Briefe können Variablen oder eigene Sätze per Tastatur eingefügt werden. Eine Verknüpfung mit Adreßdateien ist möglich, falls diese mit dem Applewriter erstellt wurden. INTERTEXT kann jederzeit durch eigene Bausteine erweitert werden. Dies wird auch unumgänglich sein, da die Textbausteine nicht branchen-, geschweige denn firmenspezifisch sind. Hinzu kommt, daß zumindest das Englisch, das mir in Form einiger Mustersätze vorlag, nicht immer idiomatisch korrekt ist, z.B. „Our customers place very much importance on first-class quality only."

4.000 Textbausteine in 1 Sprache kosten DM 1.990,-, 2 Sprachen (z.B. Deutsch/Englisch) DM 3.600,-, und alle 4 Sprachen mit insgesamt ca. 16.000 Bausteinen DM 5.900,-. Zum Vergleich kostet ein zwei- oder mehrsprachiges Satzwörterbuch für Auslandskorrespondenz mit 5.000 bis 10.000 Mustersätzen DM 50,- bis DM 100,-. So kostet etwa mein 560seitiges „Satzwörterbuch des Buch- und Verlagswesens" mit 10.000 Beispielsätzen nur DM 88,-. Dies zeigt wieder einmal, daß eine „Buch-Datenbank" erheblich preiswerter als eine „Computer-Datenbank" ist. Das TKS-System allein kostet übrigens DM 790,-, ist also teurer als der Applewriter IIe selbst (ca. DM 500,-). Das Textverarbeitungssystem setzt einen Apple IIe mit 64K-Karte und 2 Drives voraus. Für Firmen mit viel schablonenmäßiger Korrespondenz ist INTERTEXT empfehlenswert, wobei man allerdings tief in die Tasche greifen muß.

Ulrich Stiehl

SCREEN80.SAVER

Speichert den 80-Z/Z-Bildschirm von IIc/IIe als Textfile auf Diskette. Funktioniert unter DOS 3.3 und ProDOS. Nach dem Poken der Maschinenroutine SAVER aufrufen, wie in Zeile 230 angegeben. Dateiname beliebig. Zum späteren Einlesen des Textfiles die INALL-Routine verwenden (s. S. 70).

```
100 REM SCREEN80.SAVER von U.Stiehl/85
110 DATA 169,0,173,80,3,160,0,140,81,3
120 DATA 173,80,3,32,193,251,172,81,3,141
130 DATA 85,192,32,58,3,172,81,3,141,84
140 DATA 192,32,58,3,172,81,3,200,192,40
150 DATA 208,221,169,141,32,68,3,238,80,3
160 DATA 173,80,3,201,24,144,204,96,177,40
170 DATA 9,128,201,160,176,2,105,64,141,84
180 DATA 192,32,75,3,96,41,255,108,54,0,0
190 RESTORE : FOR X = 768 TO 848: READ Y:
    POKE X,Y: NEXT
200 REM Demo-Anwendung
210 PRINT CHR$ (4) "PR#3": LIST
230 PRINT CHR$ (4) "OPEN SCHIRM":
    PRINT CHR$ (4) "WRITE SCHIRM":
    CALL 768: PRINT CHR$ (4)"CLOSE"
```

Akustikkoppler AK 300

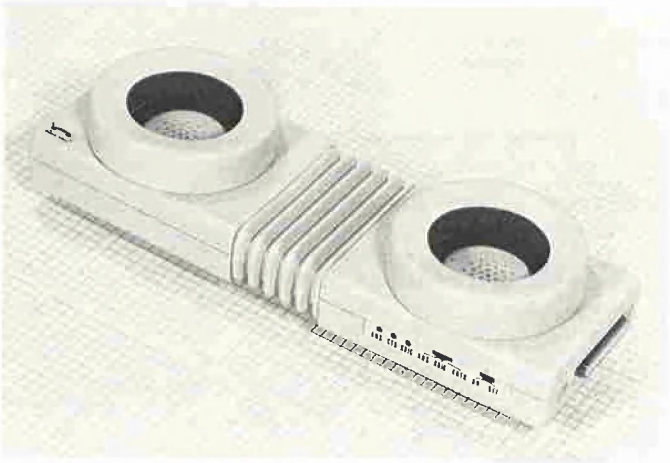
Der AK 300 der Firma Software Express ist ein postalisch zugelassener Akustikkoppler (mit FTZ-Nummer), der mit einer normalen V24-Schnittstelle ausgestattet ist und sich deshalb an den Apple IIc (V24 bereits eingebaut) oder den Apple IIe mit der Super-Serial-Card anschließen läßt. Die Datenübertragung erfolgt wie üblich mit 300 Baud, also ca. 30 Zeichen/Sekunde. Neben Netzanschluß besteht auch die Möglichkeit des Batteriebetriebs mit den normalen 1,5-Volt-Batterien, so daß sich der AK 300 besonders für den IIc empfiehlt. Einige Apple-Händler haben den AK 300 im Lieferprogramm.

Akustikkoppler AK 300

Akustikkoppler dataphon s-21-d

Die Kommunikation mit Datenbanken, Mailboxen, Großrechnern und anderen Mikrocomputern über das Telefonnetz ermöglicht der seit kurzem auf dem Markt erhältliche Akustikkoppler dataphon s-21-d, ein Gerät deutscher Herstellung (Firma HIB). Der dataphon ist FTZ-zugelassen und kann Daten mit der üblichen Geschwindigkeit von 300 Baud übertragen. Die Verbindung zum Apple erfolgt über eine V24-Standardschnittstelle wie bei dem AK 300.
Akustisch dicht schließende Gummi-Manschetten zur Aufnahme von Sprech- und Hörmuschel des Telefons gewährleisten eine einwandfreie Verbindung.

Trotz des niedrigen Preises bietet der Akustikkoppler einige Besonderheiten, die sonst teureren Geräten vorbehalten sind, z.B. außer Answer- und Originate-Modus die Möglichkeit der automatischen Kanalwahl, die den Aufbau einer Verbindung auch dann ermöglicht, wenn nicht bekannt ist, in welchem Übertragungsmodus die Gegenstation arbeitet. Die Stromversorgung (9–15V) erfolgt wahlweise über ein geregeltes Netzteil, Akku, Batterie oder über den Schnittstellenstecker vom Computer aus. Für den Apple II+, IIe (und IIc?) hat HIB entsprechene Software erstellt.

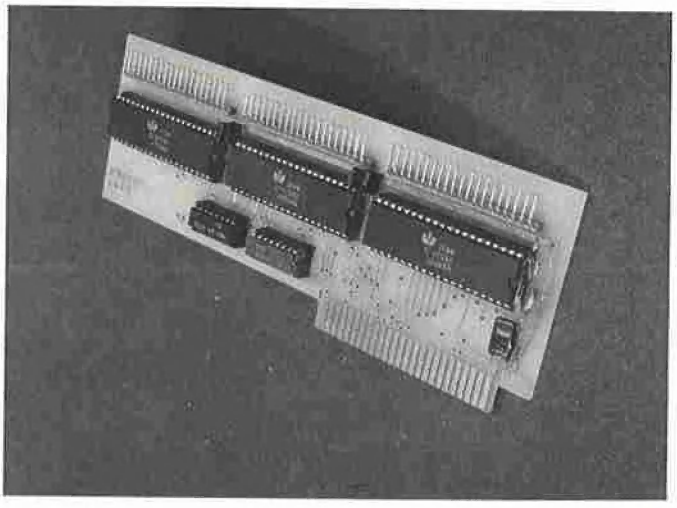


Akustikkoppler dataphon S-21-d

Parallelportkarte für Apple II

Diese Karte der Firma Brecht enthält drei VIA Bausteine R6522. Es ergeben sich dadurch 60 TTL-kompatible Ein-/Ausgabeleitungen, davon sind 12 interruptfähig. Sechs 16-Bit-Timer für Impulserzeugung, Zählen von Impulsen und Erzeugen von Verzögerungszeiten sowie drei Schieberegister lassen sich für die seriell-parallele bzw. parallel-serielle Wandlung verwenden.
Durch die Vielseitigkeit der VIAs ergeben sich universelle Möglichkeiten. Durch Kombination mit Filter- und Leistungskarten des gleichen Herstellers kann der Apple II, II+, IIe oder Kompatible in der Steuerungstechnik oder z.B. der automatischen Qualitätskontrolle eingesetzt werden.

Die Karte kann in jedem Slot installiert werden, wobei die vergoldeten Kontakte eine optimale Sicherheit in der Kontaktgabe gewährleisten. Zum Anschluß der E/A-Leitungen stehen 60 Molexfingerkontakte zur Verfügung.
Jeder Lieferung liegt eine Beschreibung der Arbeitsweise der Karte bei, aus der die Konfigurierung und Anwendung der Register zu entnehmen ist. Außerdem erhält der Käufer die (englischen) Datenblätter des VIA 6522. Der Preis der Karte beträgt knapp DM 200,-.
Vom gleichen Hersteller ist eine zweifach RS-232C-Schnittstelle mit EPROM/RAM ($CX00-$CXFF) und eine 16-Bit-A/D-Wandlerkarte für den Apple erhältlich.



Parallelportkarte

Hüthig-FACHBUCH-TIP

„Apple Assembler" wendet sich an alle, die bereits Anfängerkenntnisse der 6502-Programmierung haben 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 sowie Angaben zu den apple-spezifischen Zahlenformaten (Integer- und Fließkommazahlen). Im zweiten Teil werden neben einer Kurzwiederholung der Monitor-Befehle alle Routinen und Adressen des Monitors zusammengestellt, die für Assemblerprogrammierer von Nutzen sein können. Darüber hinaus findet der Leser Unterroutinen für Vorwärts- und Rückwärts-Moven, hexadezimale Addition/Subtraktion/Multiplikation/Division, Binär-, Hex-und ASCII-Umwandlung, Dumpen/Disassemblieren, Aufwärts-Scrollen, Reset u. a.
Der dritte Teil befaßt sich mit der Speicherverwaltung der Language-Card und der IIe-64K-Karte und enthält Test- und Move-Programme zum Verschieben von Daten in die und aus der Language Card sowie der 64K-Karte, wobei besonders ausführlich auf die Softswitches eingegangen wird.

Der vierte Teil ist dem Applesoft-ROM gewidmet und beschreibt die interne Struktur von Applesoft-Programmen, die Methoden der Parameterübergabe mittels CALL, USR, &, PEEK und POKE und listet dann eine große Anzahl nützlicher Interpreter-Adressen. Bei den Utility-Programm liegt das Schwergewicht auf Fließkommamathematik einschließlich Print Using.

Der letzte Teil behandelt den Text- und Grafikspeicher. Neben einem professionellen Maskengeneratorprogramm findet der Leser hier auch erstmals Routinen zur Double-Lores- und Double-Hires-Grafik des Apple IIe.

Apple Assembler
- Tips und Tricks -

von U. Stiehl

232 S., 40 Programm-Listings,
3 Abb., kart., DM 34,—
ISBN 3-7785-1047-9

Begleitdiskette zum Buch DM 28,—
ISBN 3-7785-1048-7

BESTELLCOUPON

Bitte ausfüllen und an Hüthig Vertriebs-service, Postfach 102869 · 6900 Hei-delberg schicken.

Buchtitel

Name

Straße

Ort

Unterschrift

Wir vertreten unter anderem folgende Firmen in Deutschland:

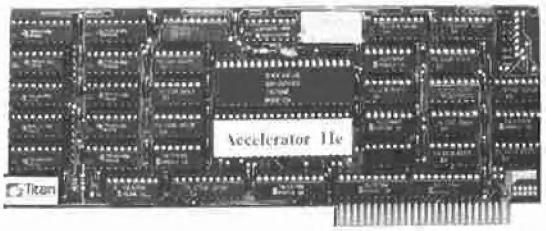
PASSPORT DESIGNS INC
TG PRODUCTS
THUNDERWARE, INC.
QUORUM
Mountain™ Computer Inc.
INTRA COMPUTER INC.
Titan TECHNOLOGIES, INC.
KENSINGTON MICROWARE
California Computer Systems
COMPUTER STATION
CENTRAL POINT Software, Inc.

Händleranfragen erwünscht.

pandasoft Dr.-Ing. Eden

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

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



Jetzt laufen VisiCalc®, Apple Writer, PASCAL, BASIC, Datenbanken usw. endlich ohne langen Zeitverlust.
Stecken Sie einfach die ACCELERATOR IIe Karte in irgendeinen Slot und beobachten Sie, wir Ihr Apple loslegt!
ACCELERATOR IIe besitzt seinen eigenen schnellen 6502 Prozessor und 80 K-Byte Hochgeschwindigkeitsspeicher, einschließlich einer eingebauten schnellen Sprachkarte und schnellem RAM-Speicherplatz für die ROM-Sprache.
Direkt von Pandasoft (Titan Distributor für Deutschland) oder bei Ihrem Applehändler.

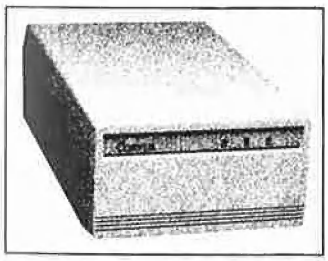
pandasoft Dr.-Ing. Eden

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

CORVUS OMNIDRIVE™

5, 11, 16 oder 45 MB
für Apple Macintosh.

Single user Version* incl. Omninet-Transporter.
Für Entfernungen bis zu 1 km,
problemlos über RS 422 Zweidrahtleitung.



CORVUS Omnidrive™
Winchester Disk

Omnidrive
Massenspeicher und Netzwerkanschluß in einem Gerät.

* ab 3. Quartal '85 aufrüstbar für Multiuser-Betrieb zum Vernetzen von bis zu 63 Mac's.

Omninet Constellation II Multiuser Version für gemischten Betrieb von Apple II +, IIe, Corvus Concept, DEC Rainbow, TI Professional, Zenith Z 100
IBM-PC sofort lieferbar

pandasoft Dr.-Ing. Eden

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

Sie haben einen Apple...

wir haben die Software...

und die Hardware...

wir haben die Bücher...

und die Zeitschriften...

Fordern Sie unseren Gratiskatalog an!

ALLES FÜR DEN APPLE II, 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: .....

intelligente Peripherie für Ihren Apple®-Rechner in Profi-Qualität, made in Germany

- Laufwerke
- RAM-Karten
- ★ Controller
- komplette Sub-Systeme

Information und Verkauf über den Fachhandel

erphi electronic GmbH, Dammweg 3, 8011 Großhelfendorf, tx 528 021 erphi d