Nr. 10/85 Oktober
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

PEEKER

MAGAZIN FÜR APPLE-COMPUTER

Grafik-Editor
Double Hires
EPROM-Gerät
P-Code-Cruncher
Pascal 1.2
Speedemon
IIe-Kompatible

10 Seiten Sonderteil
Die neuen IIe-ROMs

Hüthig
PUBLIKATION

Systemintegratoren, Entwickler und Techniker für professionellen Mini- und Mikrocomputereinsatz:
Mit dieser Zeitschrift sichern Sie sich einen

Informations-Vorsprung

Denn **mini Micro magazin** hilft Ihnen jetzt, die richtigen Hard- und Software-Entscheidungen zu treffen. Testen Sie **mini Micro magazin**, den sicheren Helfer des Systemintegrators. Machen Sie eine kompetente Redaktion zu Ihrem persönlichen Ratgeber.
Sie erhalten **mini Micro magazin** zum Subskriptionspreis von nur DM 96,– für 12 Ausgaben (incl. Mwst. und Porto). Dieser Preis gilt für Bestellungen bis 31. 12. 1985, danach zahlen Sie DM 132,–. Und dies absolut ohne Risiko. Sie können – sollte **mini Micro magazin** Ihren Erwartungen nicht entsprechen – ohne Angabe von Gründen jeweils zum 1. eines Quartals abbestellen.

Coupon

Ja, ich bin an Ihrem neuen Fachmagazin interessiert.

☐ Schicken Sie mir die Erstausgabe von **mini Micro magazin** und die folgenden Hefte zum Subskriptionspreis von DM 96,–

☐ Ich möchte **mini Micro magazin** zunächst einmal kennenlernen bevor ich mich festlege. Liefern Sie mir deshalb die Erstausgabe und die folgenden Hefte. Sollte ich eine Woche nach Erhalt der 2. Ausgabe nicht schriftlich abbestellt haben, so bleibe ich bis auf weiteres überzeugter **mini Micro magazin**-Leser.

Name, Vorname _____ Firma _____

Straße _____ PLZ/Ort _____

Telefon _____ Datum _____ X Unterschrift _____

Ich habe davon Kenntnis genommen, daß ich 1 Woche nach Erhalt der 2. Ausgabe ohne Angaben von Gründen schriftlich die weitere Belieferung mit **mini Micro magazin** einstellen kann und damit keinerlei Verpflichtungen übernehme. Erfolgt diese Abbestellung nicht, so verlängert sich mein Abonnement jeweils um ein Quartal.

Datum _____ X Unterschrift _____

Nur Bestellungen mit 2 Unterschriften können wir bearbeiten.

Coupon ausschneiden und adressieren an mini Micro magazin, Verlagsgruppe Hüthig, Landsberger Straße 439, 8000 München 60

EDITORIAL

Unter dem Motto "Patch as you patch can" werden in Kürze die „Enhanced ROMs" für den Apple IIe erscheinen. In dem Bemühen, größtmögliche Kompatibilität zu den vorangehenden Apple-II-Modellen zu erzielen, entstand ein bewundernswertes Flickwerk („Patchwork"), das seinesgleichen sucht. Es gibt kaum noch Monitor-Routinen, die an einem Stück programmiert sind. An den „offiziellen" Einsprungadressen wird meist ein Prozessorregister gesetzt, und ab geht die Post zu einer ganz anderen Speicherstelle. Würde man die historischen Hintergründe nicht kennen, so müßte man annehmen, daß hier ein etwas verschrobener Programmierer am Werk gewesen sei.

Es gab und gibt 5 Apple-II-Modelle:
– Ur-Apple II mit Step-und-Trace-Monitor und Integer-BASIC (heute nicht mehr aktuell)
– Apple II+ mit Autostart-ROM und Applesoft-BASIC; ferner entsprechende Kompatible
– Apple IIe bis dato mit neuem Autostart-ROM, CX-ROM und unverändertem Applesoft-BASIC; ferner einige neue Kompatible (siehe Testberichte in diesem Heft)
– Apple IIc mit neuem Autostart-ROM, geändertem CX-ROM und leicht modifiziertem Applesoft-BASIC
– Apple IIe enhanced mit neuem Autostart- und CX-ROM sowie (gegenüber Apple IIc) erneut geändertem Applesoft-BASIC.

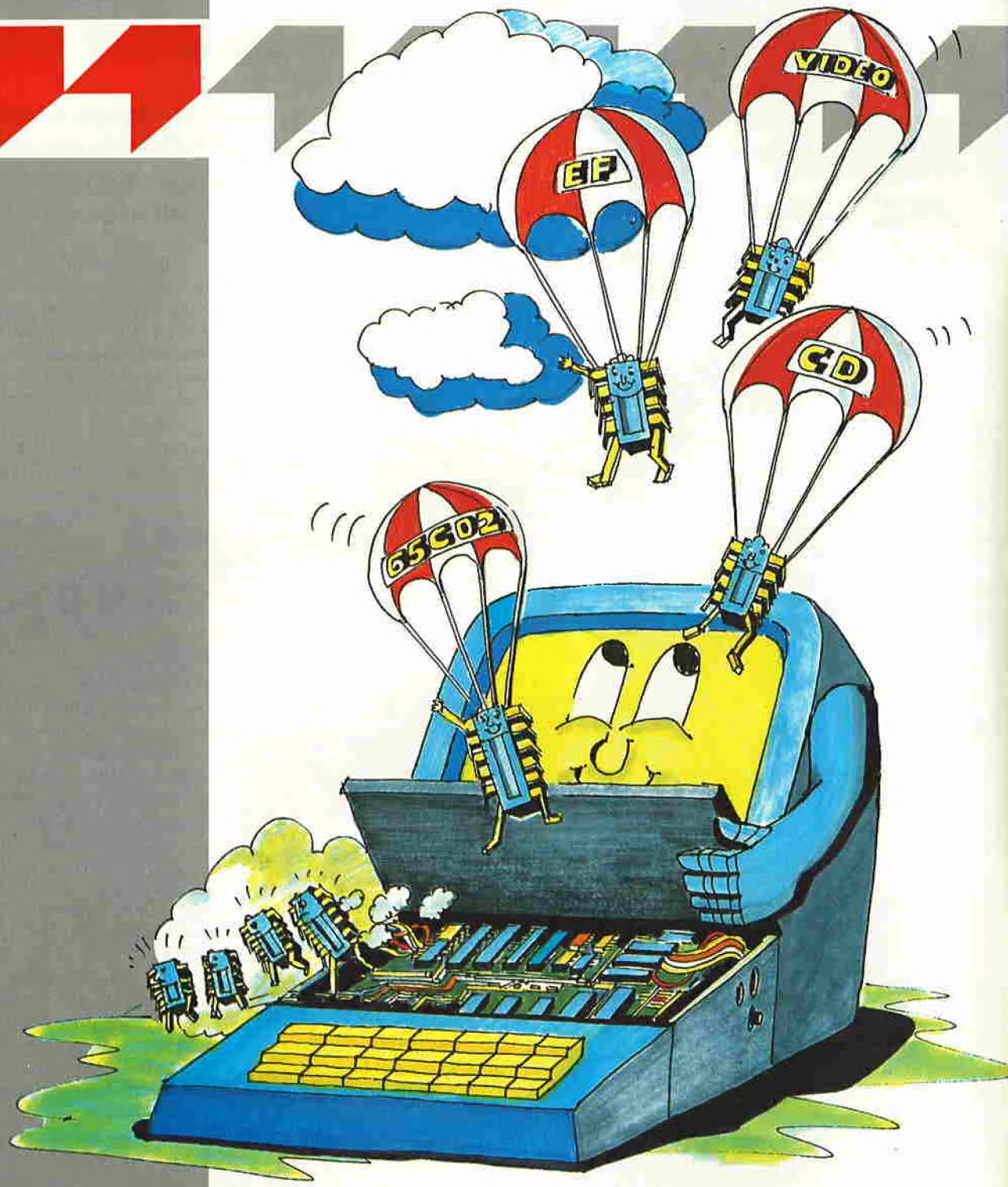
Selbst erfahrene Apple-Programmierer verlieren inzwischen den Überblick, zumal nicht jeder über jedes Modell verfügt. Für unseren Peeker gilt der Grundsatz, daß ein Programm so spezifisch wie nötig und so unspezifisch wie möglich sein sollte. Das heißt konkret:
1. Ein Programm, das spezifische Modell-Eigenschaften ausnutzt, braucht nur auf spezifischen Modellen zu laufen. Beispielsweise kann Double Hires nur auf dem IIe oder IIc funktionieren, so daß man hier auf den II+ verzichten muß. Doch sollte ein Double-Hires-Programm nicht derart „spezifisch" sein, daß es etwa nur noch auf dem IIc läuft, denn hierfür besteht kein sachlicher Grund.
2. Ein Programm, das keine spezifischen Modell-Eigenschaften ausnutzt, sollte auf allen Geräten laufen. Dies gilt im allgemeinen für alle Programme, die keine 80-Zeichenkarte voraussetzen, denn im 40-Zeichen-Modus ist auch heute noch (fast) immer Kompatibilität erreichbar.

Dies liest sich viel leichter, als es in praxi ist. Viele eingereichte Programme sind unnötig „spezifisch". Deshalb an dieser Stelle eine herzliche Bitte an alle Peeker-Autoren: Falls möglich, testen Sie Ihr Programm auch auf anderen Apple-II-Modellen. Auf alle Fälle geben Sie bitte unbedingt immer an, auf welcher Apple-Konfiguration und unter welchem Betriebssystem Sie das Programm entwickelt haben.

Wenn Sie dieses Oktober-Heft mit früheren Peeker-Heften vergleichen, werden Sie merken, daß wir den Satzspiegel um 1 cm nach oben erweitert haben. Damit gewinnen wir gegenüber früher bei gleichem Seitenumfang 2 zusätzliche Druckseiten mit redaktionellem Inhalt. Außerdem sind wir mit den Firmennachrichten und den Leserbriefen mit dem Schriftgrad nach unten gegangen, so daß diese Seiten, die nicht zur kontinuierlichen Lektüre gedacht sind, jetzt ca. 10.000 Zeichen/Seite aufweisen. Da die Listings ohnehin schon immer sehr platzsparend gesetzt worden sind, umfaßt der redaktionelle Teil des Peekers ohne die Abbildungen weit über 300.000 Zeichen/Heft. Dies entspricht etwa einem 160seitigen, konventionell gesetzten Buch oder einem 240seitigen, „modern" produzierten Data-Becker-Buch („Manuskriptbuch").

Ulrich Stiehl

INHALT



Impressum

Pesker
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: H. Wondra

Hüthig
PUBLIKATION

PEEKER

Heft 10/1985

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

Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement DM 72,–, 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 0522656

Zahlungen: an den Dr. Alfred Hüthig Verlag
GmbH, D-6900 Heidelberg 1: Postscheck-
konten: 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

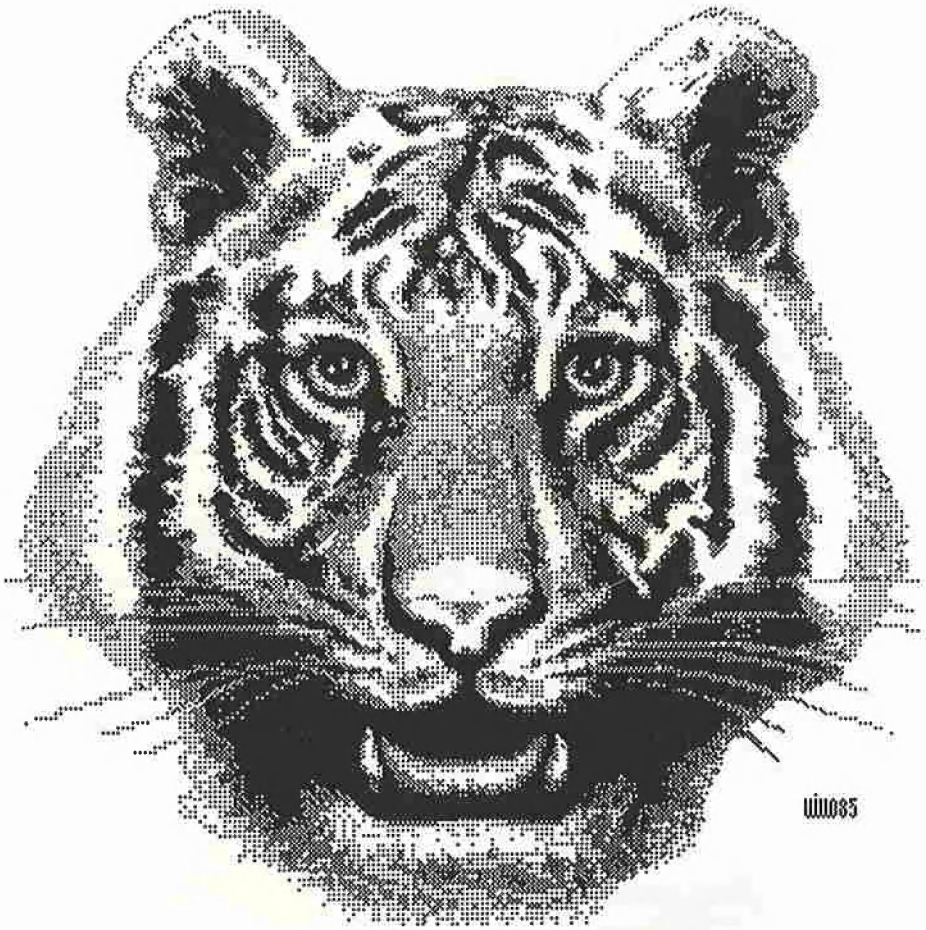
Bankkonten: Landeszentralbank Heidel-
berg 67 207 341; BLZ 672 000 00; Deutsche
Bank Heidelberg 0 2165 041; BLZ
672 700 03; Bezirkssparkasse Heidelberg
204 51, BLZ 672 500 20.

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Graf-quattro

von Nino G. Barbieri

Teil 4: Der Grafik-Editor



Zeige mir Dein BASIC, und ich werde Dir sagen, wer Du bist. Es ist immer interessant, ein BASIC-Programm vom stilistischen Standpunkt aus zu betrachten und Rückschlüsse auf den jeweiligen Autor zu ziehen. Es gibt den Superpingeligen und den totalen Chaoten, den überschwenglichen REM-Benutzer und den „Je-weniger-desto-besser"-Typ. Die stilistischen Eigenarten dieses hier abgedruckten BASIC-Programms sind aber mehr auf die vorhandenen Gegebenheiten als auf den Charakter des Autors zurückzuführen.
Gewiß, BASIC ist langsam. Compilieren ist wegen Platzmangels oft nicht möglich. Der beste Weg ist also, alles was zeitlich möglich ist, in Assembler zu schreiben und von BASIC aus mit PEEK, POKE und CALL zu steuern. Das macht fast immer aus einer Schnecke einen Flitzer. In den vorangegangenen Beiträgen ist eine Ansammlung von Maschinensprache-Routinen abgedruckt worden; wir wollen sie jetzt aus einem Applesoft-Programm heraus steuern.

1. Das Programm

Zwischen Anfang des Applesoft-Programms (2048) und Beginn der ersten Grafikseite (8192) sind genau 6144 Bytes verfügbar. Aus diesem Grund mußte das Programm **GRAFIK.EDITOR** auf REMs verzichten und so gehalten werden, daß ein Maximum an Effizienz mit einem Minimum an Platzbedarf zu erreichen war. Wenn das Programm im Speicher ist und läuft, bleiben nur noch ca. 100 Bytes übrig. Dieses bedeutet, daß nicht sehr viel Platz für Einfügungen oder Verbesserungen des Lesers bleibt, es sei denn, die verhältnismäßig aufwendigen Dialoge der LOAD- und SAVE-Routinen (Zeile 426-488) werden verkürzt oder herausgenonmen. Schließlich kann man einen Grafikseite-SAVE und -LOAD auch „zu Fuß" machen. Die Programmbeschreibung wird jetzt im Telegrammstil folgen, sonst müßte ich den ganzen Peeker in Anspruch nehmen (ich höre auch so unseren Chefredakteur stöhnen: „Zu lang, viel zu lang!"). Ich werde mich daher auf das Erläutern der einzelnen Befehle beschränken, mit knappen Hinweisen auf das Programmgeschehen. Alle Befehle erfolgen per Knopfdruck, mit Ausnahme derjenigen, die die Beantwortung einer Frage verlangen.
Nach Laden des Assembler-Teils (GRAF-.QUATTRO.1, als Zusammenstellung aller bisheriger Routinen auf der Sammeldisk #8 enthalten und hier nochmals als Hex-Dump gelistet), Initialisierung der notwendigen Variablen und Einschalten der Grafik (Zeile 100-110) springt das Programm nach Zeile 294, wo die Hauptsteuerschlei-

fe (bis 332) steht. Achtung: Der Grafikspeicher wird aus gutem Grund nicht gelöscht.

2. Allgemeine Befehle

I, J, K, M – Richtungssteuerung des Cursor 1, für die rechte Hand, in den jeweiligen logischen Richtungen (Zeile 298-304).

W, A, S, Y – Richtungssteuerung des Cursor 2, für die linke Hand (Zeile 306-312). Achtung, diejenigen, die keine deutsche Tastatur angeschlossen haben, müssen in Zeile 310 den Wert 89 (ASCII Y) mit dem Wert 90 (ASCII Z) austauschen!
Die Cursoren und evtl. andere Gebilde werden von den GOSUBs zwischen 164 und 224 gesteuert, die kontinuierlich von der Hauptschleife angesprochen werden.

Ctrl-R = Repetieren – Diese Taste (Zeile 314) bewirkt einen Sprung zu den Zeilen 366-382. Dieser Bereich sorgt für das Kopieren oder Übertragen von einem viereckigen Ausschnitt der Grafik auf die sichtbare Grafikseite 1 oder auf die unsichtbare Grafikseite 2. Beim Drücken der Taste Ctrl-R erscheint, wenn nicht schon vorhanden (s.a. Befehl V), ein blinkendes Viereck, begrenzt durch die Stellung der Cursoren. Dabei wird die Frage gestellt, ob die Übertragung auf Seite 1 oder 2 erfolgen soll. Nach Beantwortung der Frage das blinkende Viereck mit Hilfe der rechten oder linken Handsteuerung auf die gewünschte Übertragungsstelle schieben und erneut Ctrl-R drücken. Während der Positionierungszeit sorgt die Flagge U dafür, daß die Hauptschleife bis Zeile 316 begrenzt ist, so daß alle übrigen Befehle inaktiv werden.

D = Doppelte Cursor-Steuerung – Beim Drücken der Taste D (Zeile 318) wird die Flagge DO ein- und ausgeschaltet. Ist die Flagge auf 1, bewegen sich beide Cursoren synchron, gleichviel ob man mit der linken oder rechten Hand steuert. Die Routinen in den Zeilen 184-194 und in Zeile 218 sorgen dafür, daß beim Erreichen der Grafikränder keine weitere Verschiebung der Cursoren möglich ist.

B = Box – Diese Taste bewirkt eine in den vordefinierten Farben gefüllte Box (Zeile 320). Die Begrenzung der Box wird von der diagonal entgegengesetzten Stellung der Cursoren definiert.

9 und 0 = Ellipse und gefüllte Ellipse – (Zeile 322, Routinen in den Zeilen 142-158). Taste 9 bewirkt das Zeichnen einer Ellipse, wobei die X- und Y-Achse von der jeweiligen Stellung der Cursoren definiert werden. Die Ellipse wird in der vordefinierten Farbe gezeichnet. Taste 0 bildet eine gefüllte Ellipse in den vordefinierten Boxfarben. Am einfachsten ist es, Viereck vorzuschalten (s. Befehl V), um Stellung und Dimensionen der Ellipse zu bestimmen. Ist das Viereck ein Quadrat, dann bekommen wir keine Ellipse mehr, sondern einen Kreis.

P = Plotten – Beim Drücken der Taste P wird in der angegebenen Farbe das sichtbare, blinkende Gebilde auf dem Bildschirm gezeichnet.

2 = Cursor-Schrittweite – Bei Betätigung erfolgt die Frage nach der gewünschten Cursor-Schrittweite (Zeile 324, die Befragung in Zeile 292).

Ctrl-D = Disk – Das gleichzeitige Drükken der Tasten Ctrl und D (Zeile 326) bewirkt einen Sprung zu den Zeilen 426-488, wo die LOAD- und SAVE-Routinen für die Grafikseiten ihren Platz finden. Bei Platzbedarf können die Zeilen 326 und 426-488 entfernt werden. Danach muß aber das LOAD und SAVE der Grafik manuell erfolgen.

3. Zirkel-Befehle

Ctrl-K = Zirkel – (Zeile 328) ruft den Übergang auf die sekundäre Steuerschleife hervor (Zeile 334-354), die einen Zirkel simuliert.
Zwischen Zeile 258 und 286 ist eine dritte Steuerschleife, die gemeinsam von der Haupt- und Zirkelschleife benutzt wird.
Die folgenden Befehle werden nach Betätigung von Ctrl-K gültig.

Ctrl-Z = Zurück zur Hauptschleife – Diese Taste (Zeile 356) ist zu betätigen, wenn man mit der Kreisroutine fertig ist. Bei der Kreisroutine bildet einer der Cursoren immer das Zentrum des Kreises, der andere Cursor rotiert in dem gegebenen Abstand bei Betätigen der nachfolgenden Tasten.

J und K = Bogen um Mittelpunkt-Cursor – Das Zeichnen eines Bogens, eines kompletten Kreises oder eines regelmäßigen Polygons ist, je nach Betätigung anderer untenstehender Steuertasten, gegeben. Dabei bewegt die Taste J den Cursor im entgegengesetzten Uhrzeigersinn, die Taste K im Uhrzeigersinn.

P = Plotten – Die Taste P schaltet das Plotten der Kreissegmente in der vorgegebenen Farbe und definiertem Winkelgradschritt ein und aus. Einmal drücken, und der rotierende Cursor hinterläßt eine Linie, nochmals drücken, und der Cursor bewegt sich, ohne zu zeichnen (Zeile 346).

W = Wechseln – Beim Drücken der Taste W wird der jeweilige rotierende Cursor zum Mittelpunkt-Cursor und der ehemalige Drehpunkt-Cursor zum rotierenden Cursor. In dieser Art und Weise kann man sehr einfach Kreisornamente zeichnen (Zeile 348).

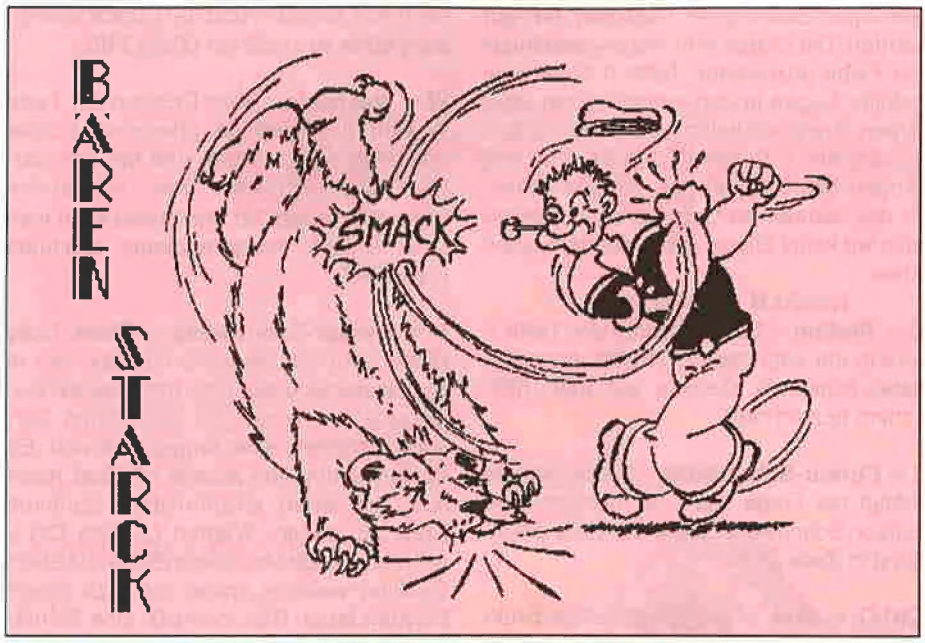
2 = Cursor-Schrittweite – Diese Taste (Zeile 350) hat dieselbe Funktion wie in der Hauptsteuerschleife, nur, daß die Bewegung des jeweiligen rotierenden Cursors in Winkelgraden angegeben wird. Eine Bewegung von jeweils 10 Grad reicht aus, um einen einigermaßen sauberen Kreis zu ziehen. Werden größere Gradschritte angegeben, zeichnet der Zirkel ein Gebilde, welches immer mehr zu einem Polygon neigt. Gibt man z.B. eine Schrittweite von 120 Grad ein, zeichnet unser Zirkel ein gleichschenkliges Dreieck. 90 Grad Schrittweite bildet ein Quadrat, 72 Grad ein Pentagon usw. Bei der Gradeingabe sind Werte mit Dezimalstellen erlaubt.

4. Gemeinsame Steuerbefehle

Sowohl die Hauptschleife als auch die Zirkelschleife benutzen die Utilities dieser gemeinsamen Schleife (Zeile 258-286). Folgende Befehle sind vorgesehen:

X = Kontinuierliches Zeichnen – Die Taste X (Zeile 260) bewirkt das Ein- und Ausschalten des kontinuierlichen Zeichnens. Gezeichnet wird in der vorgegebenen Farbe (mit Ausnahme der Farbe 8 = XOR).

L = Linie – Mit dieser Taste (Zeilen 262 und 264) wird eine blinkende Linie zwischen den zwei Cursoren ein- und ausgeschaltet. Diese Linie verhält sich wie ein Gummiband, d.h. sie verbindet immer die zwei Cursoren, gleichgültig, wie sie bewegt werden. Mit Hilfe der beiden Funktionen P und X kann man einige hübsche Dinge realisieren. Will man z.B. eine dicke Linie zeichnen, bringt man die Cursoren in den gewünschten Abstand und schaltet die Funktion D (Doppel-Cursor-Steuerung) und X ein. Bei Schrittweite 1 wird mit Hilfe der Richtungstasten (Tasten gedrückt lassen) eine Linie in der gewünschten Stärke gezeichnet. Ist die Cursor-Schrittweite eine andere als 1, wird eine Reihe von Linien in regelmäßigen Abständen geplottet. Will man die blinkende Linie nur an bestimmten Stellen zeichnen, ist die Funktion P anstatt X zu benutzen.
Ist die Zirkelroutine eingeschaltet, ist das Ergebnis dasselbe, nur daß anstatt einer



gradlinigen Bewegung eben die Radien des jeweiligen Kreises oder Bogens gezeichnet werden.

V = Viereck – (Zeile 266 und 268). Diese Taste schaltet ein blinkendes Viereck zwischen den zwei diagonal entgegengesetzten Cursoren ein und aus. Das Viereck verhält sich in derselben Art und Weise wie die Linie, d.h. in Verbindung mit den Funktionen D, X und P. Es lohnt sich, mit verschiedenen gleichzeitig eingeschalteten Funktionen ein bißchen zu experimentieren. Probieren Sie z.B. einen größeren Kreis mit eingeschalteter Viereckfunktion (V- und X-Taste) zu zeichnen. Viereck ermöglicht besseres Dimensionieren und Positionieren bei der Benutzung der Funktionen: Ausschnitt übertragen, Box, Ellipse und gefüllte Ellipse.

Leertaste = Textzeilen – Die Leertaste (Zeile 270) schaltet die vier Textzeilen unter der Grafik ein und aus. Sind die Textzeilen eingeblendet, werden automatisch Informationen sichtbar, und das Programm wird langsamer, da bei jedem Tastendruck die neue Situation angezeigt wird (Zeile 384-424). Die Fußnoten sind in drei Spalten aufgeteilt und zeigen folgendes an (immer von oben nach unten):
Linke Spalte:
– XCOL. 1 = die Farbe, die benutzt wird beim Einschalten der Funktion X.
– PCOL. 2 = die Farbe der Funktion P.
– BCOL. 1 und 2 = die Farben, mit denen Box, Ellipse und Hintergrund gefüllt werden.
Mittlere Spalte:
Der Reihe nach: Funktionen L, V, X und D an oder aus. Diese Angaben haben reinen Erinnerungswert.
Rechte Spalte bei der Hauptschleife:
– X-,Y-Position des Cursors 1;
– X-,Y-Position des Cursors 2;
– horizontaler und vertikaler Abstand der zwei Cursoren;
– aktuelle Cursor-Schrittweite.
Rechte Spalte bei Zirkelschleife:
– X-,Y-Position des stehenden Cursors (Kreismitte);
– absolute Winkelangabe zwischen stehendem und rotierendem Cursor (sogar nach DIN, d.h. 0 Grad rechts, 90 Grad oben, 180 Grad links und selbstverständlich 270 Grad unten);
– die jeweilige rotierende Cursor-Bewegung in Grad;
– Angabe, ob die Kreiszeichnerfunktion an oder aus ist.

1 = Cursor-Schrittweite 1/10 – Die Taste 1 (Zeile 272) bewirkt eine Umschaltung der Cursor-Schrittweite von 1 auf 10 und umgekehrt. Bei der Kreisroutine natürlich nicht in Punkten, sondern in Grad.

Ctrl-S = Seiten austauschen (swappen) – Bei Betätigung der Taste Ctrl-S (Zeile 274) werden die Grafikseite 1 und 2 miteinander vertauscht. Das ist bequem, wenn man mit zwei Grafikseiten gleichzeitig arbeiten will oder wenn ein Teil der Grafikseite 2 (Hilfsseite) auf die Grafikseite 1 (Arbeitsseite) zu übertragen ist. Also zuerst swappen, dann übertragen und nochmals swappen.

Ctrl-T = Test 1 – Die Taste Ctrl-T (Zeile 276) bewirkt ein schnelles Hin- und Herschalten der beiden Grafikseiten, so daß sie quasi gleichzeitig zu sehen sind. Diese Funktion ist nützlich, wenn man den Unterschied zwischen der ersten und zweiten Grafik genau betrachten will, z.B. einen Bewegungsschritt bei Animation. Abstellen mit irgendeiner anderen Taste.

T = Test 2 – Diese Funktion (Zeile 278) macht nur die unsichtbare Seite zur Kontrolle sichtbar; irgendeine andere Taste führt zum ursprünglichen Zustand zurück.

Ctrl-H = Hintergrund – Die Taste Ctrl-H (Zeile 280) füllt die Grafikseite mit den angegebenen Farben. Sind die angegebenen Farben beide mit 8 definiert worden, wird die Grafikseite negativiert. Vorsicht: Mit Ausnahme der Negativierungsfunktion zerstört der Befehl Ctrl-H den Inhalt der Grafikseite.

Taste F = Farbe 1 – Beim Drücken der Taste F (Zeile 282 und GOSUB nach 230 und 232) wird die Frage nach der gewünschten Linienfarbe gestellt. Die Farben werden, wie beim Apple üblich, zwischen 0 und 7 definiert. Dieser Farbbereich gilt gleichzeitig für die Funktion X und P. Neu in diesem Grafiksystem ist die Farbe 8 = XOR, d.h. immer negativ zum Hintergrund. Die Farbe 8 gilt nur für die Funktion P, da sie für die Funktion X zu unerwünschten Effekten führen könnte.

Ctrl-F = Farbe 2 – Drückt man Ctrl-F (Zeile 284, GOSUB nach 234), wird zweimal nach einer Farbdefinition gefragt. Hier kann man entweder zweimal dieselbe oder auch verschiedene Farben angeben, einschließlich der Farbe 8. Diese Farbdefinition gilt für Box, gefüllte Ellipse und Hintergrund. Gefüllt wird alternativ in den zwei angegebenen Farben. Auch hier lohnt es sich zu probieren, z.B. alternativ mit Farbe 8 und einer anderen, um durchscheinende Effekte zu erreichen.
Ich habe versucht, die Tastenfunktionen nach einer gewissen Mnemonik zu verteilen (z.B. F für Farbe, L für Linie usw.), so daß sie schnell zu erlernen und einfach zu behalten sind. Wer nicht zufrieden ist, kann selbstverständlich seine eigenen Funktionstasten programmieren: Einfach in der angegebenen Zeile den gewünschten ASCII-Wert einsetzen.
Eine letzte Anmerkung: Zeile 252 muß ohne Leertasten abgetippt werden, sonst paßt sie nicht in den Eingabepuffer.

5. Wie geht's weiter?

Da sich die Veröffentlichung der kompletten Serie noch über viele Monate hinziehen würde, wird sie mit diesem Grafik-Editor zunächst abgeschlossen. Statt dessen soll „Graf-quattro" entweder als Peeker-Sonderheft oder in Buchform erscheinen. Wir werden Sie rechtzeitig informieren.

Auf Profis programmiert:

Mit mc kommen Sie jeden Monat auf neue Ideen, wenn Sie selbst programmieren oder Hardware bauen und verändern.

Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.

Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.

In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.

Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.
Für DM 6,50 bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.

Ein kostenloses Probeheft schicken wir Ihnen gerne.

Das ist Ihr Gutschein:

mc
Die Mikrocomputer-Zeitschrift

GUTSCHEIN für ein kostenloses Probeheft der mc

Bitte schicken Sie mir die neueste Ausgabe der mc kostenlos an meine folgende Anschrift:

Name

Beruf

Straße

PLZ/Ort

Coupon auf Postkarte kleben, mit 60 Pfennig freimachen und ab damit an

Franzis'
Franzis-Verlag, Abt. Service (PE)
Postfach 37 01 20
8000 München 37

GRAF.QUATTRO.1

(Bereits als Quellcode in den vorangehenden Peeker-Folgen 4/85, 6/85 und 8/85 veröffentlicht)

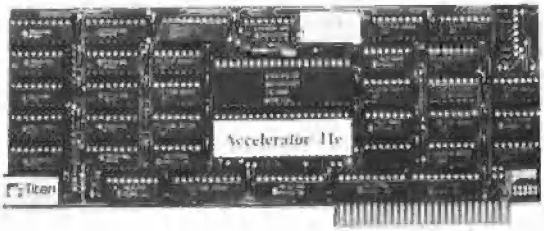
```
BSAVE GRAF.QUATTRO1, A$6000 L$0466
$6000: A9 3A 20 C0 DE A9 93 20
$6008: C0 DE 20 B7 00 C9 C1 F0
$6010: 22 20 B9 F6 48 20 B7 00
$6018: C9 C1 F0 0B 68 20 11 F4
$6020: A5 30 29 7F 4C 60 F4 68
$6028: 20 11 F4 20 B7 00 C9 C1
$6030: F0 01 60 20 C0 DE 20 B9
$6038: F6 84 9D A8 8A A6 9D 48
$6040: 38 E5 E0 48 8A E5 E1 85
$6048: D3 B0 0A 68 49 FF 69 01
$6050: 48 A9 00 E5 D3 85 D1 85
$6058: D5 68 85 D0 85 D4 68 85
$6060: E0 86 E1 98 18 E5 E2 90
$6068: 04 49 FF 69 FE 85 D2 84
$6070: E2 66 D3 38 E5 D0 AA A9
$6078: FF E5 D1 85 1D A4 E5 B0
$6080: 05 0A 20 65 F4 38 A5 D4
$6088: 65 D2 85 D4 A5 D5 E9 00
$6090: 85 D5 A5 30 29 7F 51 26
$6098: 91 26 E8 D0 04 E6 1D F0
$60A0: 8A A5 D3 B0 DC 20 D3 F4
$60A8: 18 A5 D4 65 D0 85 D4 A5
$60B0: D5 65 D1 50 DB AE CA 03
$60B8: AC CB 03 AD CC 03 20 D2
$60C0: 60 A9 3A 20 C0 DE 20 B9
$60C8: F6 8E CA 03 8C CB 03 8D
$60D0: CC 03 20 11 F4 A2 DE A0
$60D8: 60 A9 00 4C 5D F6 32 07
$60E0: C1 C1 67 21 55 3A 07 00
$60E8: AE CD 03 AC CE 03 AD CF
$60F0: 03 20 05 61 A9 3A 20 C0
$60F8: DE 20 B9 F6 8E CD 03 8C
$6100: CE 03 8D CF 03 20 11 F4
$6108: A2 11 A0 61 A9 00 4C 5D
$6110: F6 1C 1C 4D F1 1E 1E 1E
$6118: 4D 39 C1 07 00 20 46 61
$6120: AE CA 03 AC CB 03 AD CC
$6128: 03 20 11 F4 AE CE 03 AC
$6130: CF 03 AD 00 03 F0 09 AD
$6138: CD 03 20 3A F5 4C 46 61
$6140: AD CD 03 20 3F 60 AE CA
$6148: 03 AC CB 03 AD CC 03 20
$6150: D2 60 AE CD 03 AC CE 03
$6158: AD CF 03 20 05 61 60 20
$6160: 46 61 AE CA 03 AC CB 03
$6168: AD CC 03 20 11 F4 AC CC
$6170: 03 AE CE 03 AD CD 03 20
$6178: A1 61 AC CF 03 AE CE 03
$6180: AD CD 03 20 A1 61 AC CF
$6188: 03 AE CB 03 AD CA 03 20
$6190: A1 61 AC CC 03 AE CB 03
$6198: AD CA 03 20 A1 61 4C 46
$61A0: 61 48 AD 00 03 F0 04 68
$61A8: 4C 3A F5 68 4C 3F 60 20
$61B0: 46 61 20 3F 62 A9 00 85
$61B8: 01 AD CC 03 AE CF 03 8D
$61C0: C6 03 8E C9 03 CD CF 03
$61C8: D0 14 AD C6 03 AE C9 03
$61D0: 8D CC 03 8E CF 03 A9 00
$61D8: 8D 00 03 4C 46 61 B0 0E
$61E0: 48 AD CF 03 8D CC 03 68
$61E8: 8D CF 03 AD CC 03 38 ED
$61F0: CF 03 85 00 E6 00 A5 01
$61F8: D0 1A C6 01 A5 02 C9 08
$6200: D0 08 A9 00 8D 00 03 4C
$6208: 1E 62 85 E4 A9 01 8D 00
$6210: 03 4C 1E 62 E6 01 A5 03
$6218: C9 08 F0 E6 D0 EC AE CA
$6220: 03 AC CB 03 AD CC 03 20
$6228: 11 F4 AE CE 03 AC CC 03
$6230: AD CD 03 20 A1 61 CE CC
$6238: 03 C6 00 F0 8D D0 B7 AE
$6240: 02 03 BD 50 62 85 02 AE
$6248: 03 03 BD 50 62 85 03 60
$6250: 00 2A 55 7F 80 AA D5 FF
$6258: 08 20 46 61 20 3F 62 A9
$6260: 00 85 01 A9 C0 85 00 C6
$6268: 00 A5 01 D0 1A C6 01 A5
$6270: 02 C9 08 D0 08 A9 00 8D
$6278: 00 03 4C 91 62 85 E4 A9
$6280: 01 8D 00 03 4C 91 62 E6
$6288: 01 A5 03 C9 08 F0 E6 D0
$6290: EC A2 00 A0 00 A5 00 20
$6298: 11 F4 A2 01 A4 00 A9 17
$62A0: 20 A1 61 A5 00 D0 C0 4C
$62A8: 46 61 A9 40 85 09 A9 20
$62B0: 85 07 A9 00 85 06 85 08
$62B8: A8 B1 06 AA B1 08 91 06
$62C0: 8A 91 08 C8 D0 F3 E6 07
$62C8: E6 09 A5 07 C9 40 D0 E9
$62D0: 60 AD C6 03 CD CC 03 D0
$62D8: 06 A9 02 85 09 D0 0C B0
$62E0: 06 A9 01 85 09 D0 04 A9
$62E8: 00 85 09 A5 FF D0 27 AD
$62F0: C5 03 CD CB 03 F0 04 B0
$62F8: 1D 90 15 AD C4 03 CD CA
$6300: 03 D0 0B A5 09 C9 02 F0
$6308: 19 85 03 4C 1A 63 B0 06
$6310: A9 01 85 03 D0 04 A9 00
$6318: 85 03 AD C6 03 CD C9 03
$6320: D0 05 A9 20 85 E6 60 90
$6328: 0C 38 ED C9 03 85 1F A5
$6330: 09 F0 0F D0 16 38 AD C9
$6338: 03 ED C6 03 85 1F A5 09
$6340: F0 09 AE C9 03 AC CF 03
$6348: 4C 51 63 AE C6 03 AC CC
$6350: 03 86 02 84 08 E6 1F AD
$6358: CA 03 CD CD 03 AD CB 03
$6360: ED CE 03 B0 17 38 AD CD
$6368: 03 ED CA 03 85 1D AD CE
$6370: 03 ED CB 03 85 1E A5 03
$6378: D0 17 F0 28 38 AD CA 03
$6380: ED CD 03 85 1D AD CB 03
$6388: ED CE 03 85 1E A5 03 D0
$6390: 13 AE C7 03 AC C8 03 86
$6398: 00 84 01 AE CD 03 AC CE
$63A0: 03 4C B4 63 AE C4 03 AC
$63A8: C5 03 86 00 84 01 AE CA
$63B0: 03 AC CB 03 86 06 84 07
$63B8: E6 1D D0 02 E6 1E A6 00
$63C0: A4 01 86 F9 84 FA A6 06
$63C8: A4 07 86 FB 84 FC A6 1D
$63D0: A4 1E 86 FD 84 FE A9 20
$63D8: 85 E6 A6 00 A4 01 A5 02
$63E0: 20 11 F4 B1 26 25 30 29
$63E8: 7F F0 04 A9 7F D0 02 A9
$63F0: 00 85 E4 A5 FF F0 04 A9
$63F8: 40 85 E6 A6 06 A4 07 A5
$6400: 08 20 57 F4 C6 1D D0 06
$6408: A5 1E F0 27 C6 1E A5 03
$6410: D0 0E E6 00 D0 02 E6 01
$6418: E6 06 D0 BA E6 07 D0 B6
$6420: A5 00 D0 02 C6 01 C6 00
$6428: A5 06 D0 02 C6 07 C6 06
$6430: 4C D6 63 C6 1F F0 2A A5
$6438: 09 F0 07 C6 02 C6 08 4C
$6440: 46 64 E6 02 E6 08 A6 F9
$6448: A4 FA 86 00 84 01 A6 FB
$6450: A4 FC 86 06 84 07 A6 FD
$6458: A4 FE 86 1D 84 1E 4C D6
$6460: 63 A9 20 85 E6 60
```

GRAFIK.EDITOR

Dieses Programm läuft auf Apple II+/e/c unter DOS 3.3, nicht unter ProDOS. Starten mit
RUN GRAFIK.EDITOR

```
100 REM Grafik-Editor
102 REM von N.G. Barbieri
104 PRINT CHR$ (4);"BLOAD GRAF.QUATTRO.1": REM s. Hex-Dump
106 HIMEM: 8191: POKE 227,0: HCOLOR= 3:C = 3: POKE 769,3:
    POKE - 16297,0: POKE - 16304,0: POKE - 16300,0: POKE -
    16302,0: POKE 230,32:PI = ATN (1) * 4:DI = 180 / PI
108 T = - 16384:S = - 16368:LI = 24861:VI = 24927:V = 0:L
    = 0:K1 = V:K2 = V:D0 = V: POKE 768,0: POKE 769,3: POKE
    770,8: POKE 771,8: POKE 34,20: HOME
110 C1 = 24757:C2 = 24808:X1 = 20:Y1 = X1:X2 = 10:Y2 = X2:
    SCALE= 1: CALL C1 + 12:X1,Y1: CALL C2 + 12:X2,Y2:FA =
    10:S1 = 1: POKE - 16302,0: GOTO 294
112 W = W + FA: IF W > 360 THEN W = W - 360
114 GOTO 118
116 W = W - FA: IF W < 0 THEN W = W + 360
118 XO = X:YO = Y: IF K3 THEN CALL C2 + 12:X2,Y2: GOTO 122
120 CALL C1 + 12:X1,Y1
122 WI = W / 180 * PI:Y = INT (YM + XW * SIN (WI) + .5):X
    = INT (XM + XW * COS (WI) + .5): GOSUB 208: IF P AND
    PEEK (769) < > 8 THEN HPLOT XO,YO TO X,Y: GOTO 126
124 IF P THEN CALL 24576: HPLOT XO,YO TO X,Y
126 IF K3 THEN CALL C2 + 12:X,Y:X2 = X:Y2 = Y: RETURN
128 CALL C1 + 12:X,Y:X1 = X:Y1 = Y: RETURN
130 XW = INT ( SQR ((XM - X) ↑ 2 + (YM - Y) ↑ 2) + .5)
132 IF XM = X AND YM = Y THEN RETURN
134 IF XM = X AND YM > Y THEN W = 270: RETURN
136 IF XM = X AND YM < Y THEN W = 90: RETURN
138 W = DI * ATN ((YM - Y) / (XM - X)): IF XM > X THEN W =
    180 + W
140 RETURN
142 XM = (X1 + X2) / 2:YM = (Y1 + Y2) / 2:XW = ABS (X2 -
    X1) / 2:YW = ABS (Y2 - Y1) / 2: IF F = 48 THEN 148
144 IF PEEK (769) = 8 THEN CALL 24576: HPLOT XM + XW,YM:
    FOR I = 1 TO 36:F = 6.28 * I / 36:XO = XM + XW * COS
    (F):YO = YM + YW * SIN (F): CALL 24576: HPLOT TO XO +
    .55,YO + .55: GOTO 158
146 HPLOT XM + XW,YM: FOR I = 1 TO 36:F = 6.28 * I / 36:XO
    = XM + XW * COS (F):YO = YM + YW * SIN (F): HPLOT TO
    XO + .55,YO + .5: GOTO 158
148 YW = ABS (Y2 - Y1) + 1: IF INT (YW / 2) * 2 < > YW
    THEN YW = YW + 1
150 FOR I = 1 TO INT (YW / 2):XW = (YW / 2) - I:XO = .5 +
    ABS (X2 - X1) * SQR (.25 - (XW * XW) / (YW * YW)):F =
    1 * (F = 0):J = PEEK (770 + F): IF J = 8 THEN POKE
    768,0: GOTO 156
152 POKE 768,1: HCOLOR= J
154 HPLOT ABS (XM - XO) + 1,YM + XW TO XM + XO,YM + XW:
    HPLOT ABS (XM - XO) + 1,YM - XW TO XM + XO,YM - XW:
    GOTO 158
156 CALL 24576: HPLOT ABS (XM - XO) + 1,YM + XW TO XM +
    XO,YM + XW: IF XW THEN CALL 24576: HPLOT ABS (XM - XO)
    + 1,YM - XW TO XM + XO,YM - XW
158 NEXT : POKE 768,0: HCOLOR= C: RETURN
160 CALL C1 + 12:X1,Y1: CALL C2 + 12:X2,Y2: IF KR THEN
    RETURN
162 GOTO 178
164 IF L THEN CALL LI
166 IF V THEN CALL VI
168 IF DO AND K1 OR D0 AND K2 THEN 184
170 IF K1 THEN X = X1:Y = Y1: GOSUB 198:X1 = X:Y1 = Y
172 CALL C1:X1,Y1
174 IF K2 THEN X = X2:Y = Y2: GOSUB 198:X2 = X:Y2 = Y
176 CALL C2:X2,Y2
178 IF L THEN CALL LI
180 IF V THEN CALL VI
182 K1 = 0:K2 = 0: RETURN
184 X = X2:Y = Y2: GOSUB 198:X2 = X:Y2 = Y
186 X1 = X2 + XU:Y1 = Y2 + YU
188 IF X1 < 0 THEN X1 = 0:X2 = X1 + ABS (XU)
190 IF Y1 < 0 THEN Y1 = 0:Y2 = Y1 + ABS (YU)
192 IF X1 > 279 THEN X1 = 279:X2 = X1 - XU
194 IF Y1 > 191 THEN Y1 = 191:Y2 = Y1 - YU
196 CALL C1:X1,Y1: GOTO 176
198 ON Z GOTO 200,202,204,206
200 X = X + FA: GOTO 208
202 X = X - FA: GOTO 208
204 Y = Y + FA: GOTO 208
206 Y = Y - FA
```

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



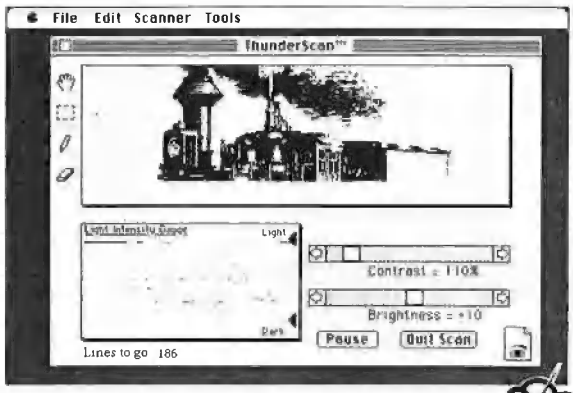
Jetzt laufen VisiCalc®, Apple Writer, PASCAL, BASIC, Datenbanken usw. endlich ohne langen Zeitverlust. Stecken Sie einfach die ACCELERATOR IIe Karte in irgendeinen Slot und beobachten Sie, wir Ihr Apple loslegt!
ACCELERATOR IIe besitzt seinen eigenen schnellen 6502 Prozessor und 80 K-Byte Hochgeschwindigkeitsspeicher, 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

ThunderScan.™

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



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

ThunderScan

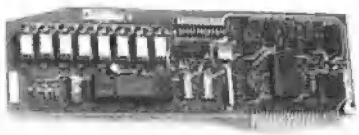
pandasoft Dr.-Ing. Eden

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

Orange™ Printer Interface **Druckerinterfaces** für Apple II+/e/c/IIi Interfaces auf dem **neuesten Stand der Technik. Kompatibel** mit allen gängigen Druckern wie: APPLE, EPSON, STAR, NEC, OKIDATA usw. Passende Treibersoftware wird über Dip-Switch ausgewählt.

Grappler + Printer Interface **Grafikfähiges Druckerinterface** das keine Wünsche mehr offen läßt. Über **2 Dutzend Kommandos** ermöglichen die volle Kontrolle über alle Möglichkeiten Ihres Druckers. Jetzt auch mit **IIe Features: Double Hires Grafics** und **80 Zeichen Dump** mittels Druckerpuffer nachrüstbar über Bufferboard.

Offered Grappler + Besitzt alle Vorzüge des Grappler +, hat aber zusätzlich einen integrierten **16 K Druckpuffer,** der auf **32 oder 64 K aufrüstbar** ist.



Serial Grappler Printer Interface **Serielles Druckerinterface** speziell für den **Apple Imagewriter.**

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

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

pandasoft Dr.-Ing. Eden

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

Sie haben einen Apple...

wir haben die Software...

und die Hardware...

wir haben die Bücher...

und die Zeitschriften...

Fordern Sie unseren Gratiskatalog an!
ALLES FÜR DEN APPLE II+, IIe, IIc UND MACINTOSH

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

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

Hinweis: Beachten Sie bitte auch die Cursor-Berichtigung, S. 63

```
208 IF X < Ø THEN X = Ø
210 IF X > 279 THEN X = 279
212 IF Y < Ø THEN Y = Ø
214 IF Y > 191 THEN Y = 191
216 RETURN
218 IF DO THEN XU = X1 - X2:YU = Y1 - Y2: RETURN
220 X1 = PEEK (97Ø) + PEEK (971) * 256:Y1 = PEEK (972):X2
    = PEEK (973) + PEEK (974) * 256:Y2 = PEEK (975):
    RETURN
222 IF PEEK (769) < > 8 THEN POKE 768,1: GOSUB 164: POKE
    768,Ø
224 GOTO 178
226 F = PEEK (T) - 176: IF F < Ø OR F > 8 THEN 226
228 POKE S,Ø: RETURN
230 HOME : POKE - 16301,Ø: PRINT " Linie-Farbe (1/8) ? ";:
    GOSUB 226: HOME : POKE - 16302,Ø: POKE 769,F: IF F = 8
    THEN POKE 768,Ø:F = Ø: RETURN
232 C = F: HCOLOR= C:F = Ø: RETURN
234 HOME : POKE - 16301,Ø: PRINT "1. Box-Farbe (1-8) ? ";:
    GOSUB 226: POKE 77Ø,F: PRINT : PRINT "2. Box-Farbe
    (1-8) ? ";: GOSUB 226: POKE 771,F:F = Ø: HOME : POKE -
    16302,Ø: RETURN
236 POKE - 16302,Ø:S1 = 1
238 POKE - 16299,Ø: FOR I = 1 TO 5Ø: NEXT : POKE -
    16300,Ø: FOR I = 1 TO 5Ø: NEXT : IF PEEK (T) < 128
    THEN 238
240 POKE S,Ø: RETURN
242 POKE - 16299,Ø: IF PEEK (T) < 128 THEN 242
244 POKE - 16300,Ø: GOTO 240
246 GOSUB 178: CALL 25177: HCOLOR= C: POKE 768,Ø: GOSUB
    178: RETURN
248 POKE - 16301,Ø: HOME : PRINT " Übertragen auf Seite
    (1-2) ? ";: GOSUB 226:UE = F - 1: IF UE < Ø OR UE > 1
    THEN 248
250 POKE - 16302,Ø: RETURN
252 CALL 24576: HPLOT PEEK (964) + PEEK (965) * 256, PEEK
    (966) TO PEEK (964) + PEEK (965) * 256, PEEK (969) TO
    PEEK (967) + PEEK (968) * 256, PEEK (969) TO PEEK
    (967) + PEEK (968) * 256, PEEK (966) TO PEEK (964) +
    PEEK (965) * 256, PEEK (966): RETURN
254 X1 = PEEK (964) + PEEK (965) * 256:Y1 = PEEK (966):X2
    = PEEK (967) + PEEK (968) * 256:Y2 = PEEK (969):
    RETURN
256 IF S1 THEN RETURN
258 IF F = 8Ø THEN GOSUB 222
260 IF F = 88 THEN POKE 768,1 * ( PEEK (768) = Ø): IF NOT
    PEEK (768) AND NOT KR THEN F = 8Ø: GOTO 258
262 IF F = 76 THEN CALL LI:L = 1 * (L = Ø)
264 IF F = 76 AND KR THEN CALL LI
266 IF F = 86 THEN CALL VI:V = 1 * (V = Ø)
268 IF F = 86 AND KR THEN CALL VI
270 IF F = 32 THEN S1 = 1 * (S1 = Ø): POKE - 16301 - S1,Ø
272 IF F = 49 THEN FA = 1 + 9 * (FA = 1)
274 IF F = 19 THEN GOSUB 16Ø: CALL 25258: GOSUB 16Ø
276 IF F = 2Ø THEN GOSUB 236:F = Ø
278 IF F = 84 THEN GOSUB 242:F = Ø
280 IF F = 8 THEN GOSUB 246: IF KR THEN GOSUB 178
282 IF F = 7Ø THEN GOSUB 23Ø
284 IF F = 6 THEN GOSUB 234
286 RETURN
288 HOME : POKE - 16301,Ø: PRINT "Winkel-Schritt (Max.
    12Ø) ";: INPUT FA: IF FA < Ø OR FA > 12Ø THEN 288
290 RETURN
292 HOME : POKE - 16301,Ø: INPUT "Cursor-Schritt (1-191) ?
    ";FA: POKE - 16302,Ø: IF FA < 1 OR FA > 191 THEN 292
294 IF NOT S1 THEN GOSUB 164: GOSUB 384
296 GOSUB 164:F = PEEK (T) - 128: IF F < Ø THEN 296
298 POKE S,Ø: IF F = 75 THEN Z = 1:K1 = Z
300 IF F = 74 THEN Z = 2:K1 = Z
302 IF F = 77 THEN Z = 3:K1 = Z
304 IF F = 73 THEN Z = 4:K1 = Z
306 IF F = 83 THEN Z = 1:K2 = Z
308 IF F = 65 THEN Z = 2:K2 = Z
310 IF F = 89 THEN Z = 3:K2 = Z
312 IF F = 87 THEN Z = 4:K2 = Z
314 IF F = 18 THEN 366
316 IF U THEN GOSUB 256: GOTO 294
318 IF F = 68 THEN DO = 1 * (DO = Ø): GOSUB 218
320 IF F = 66 THEN GOSUB 178: CALL 2500?: GOSUB 178
322 IF F = 48 OR F = 57 THEN GOSUB 178: GOSUB 142: GOSUB
    178
324 IF F = 5Ø THEN 292
326 IF F = 4 THEN 426
328 IF F = 11 THEN KR = 1: GOTO 334
330 GOSUB 258
332 GOTO 294
334 GOSUB 178:P = Ø:K3 = Ø: GOSUB 36Ø: GOTO 352
336 GOSUB 164:F = PEEK (T) - 128: IF F < Ø THEN 354
338 POKE S,Ø: GOSUB 26Ø
340 IF F = 26 THEN 356
342 IF F = 75 THEN GOSUB 112
344 IF F = 74 THEN GOSUB 116
346 IF F = 8Ø THEN P = 1 * (P = Ø)
348 IF F = 87 THEN GOSUB 36Ø
350 IF F = 5Ø THEN GOSUB 288
352 IF NOT S1 THEN GOSUB 384
354 GOTO 336
356 GOSUB 178: IF DO THEN GOSUB 218
358 KR = Ø:FA = 1Ø: GOTO 33Ø
360 IF K3 THEN 364
362 XM = X1:YM = Y1:X = X2:Y = Y2: GOSUB 13Ø:K3 = 1:
    RETURN
364 XM = X2:YM = Y2:X = X1:Y = Y1: GOSUB 13Ø:K3 = Ø:
    RETURN
366 IF U THEN 372
368 DX = DO:DO = 1: GOSUB 218: POKE 255,1: GOSUB 248: IF
    NOT UE THEN 376
370 GOSUB 382: CALL 25258: GOSUB 16Ø: GOTO 294
372 IF NOT UE THEN 376
374 U = Ø: GOSUB 16Ø: CALL 25258: POKE 23Ø,32: POKE -
    16299,Ø: CALL 25297: GOSUB 16Ø: GOSUB 254: POKE -
    163ØØ,Ø:DO = DX: GOTO 294
376 IF U THEN 38Ø
378 DX = DO:DO = 1: GOSUB 218: POKE 255,Ø: GOSUB 382:
    GOSUB 252: GOSUB 16Ø: GOTO 294
380 U = Ø: GOSUB 16Ø: GOSUB 252: CALL 25297: GOSUB 16Ø:
    GOSUB 254:DO = DX: GOTO 294
382 U = 1: FOR I = 964 TO 969: POKE I, PEEK (I + 6): NEXT
    : GOSUB 16Ø:V = 1: RETURN
384 HOME : PRINT "XCOL. 1=";C: PRINT "PCOL. 2="; PEEK
    (769): PRINT "BCOL. 1="; PEEK (77Ø): PRINT "BCOL. 2=";
    PEEK (771);
386 VTAB 21: HTAB 12: PRINT "Lin.  ";: IF L THEN PRINT
    "an": GOTO 39Ø
388 PRINT "aus"
390 VTAB 22: HTAB 12: PRINT "Vier. ";: IF V THEN PRINT
    "an": GOTO 394
392 PRINT "aus"
394 VTAB 23: HTAB 12: PRINT "Plot  ";: IF PEEK (768) THEN
    PRINT "an": GOTO 398
396 PRINT "aus"
398 VTAB 24: HTAB 12: PRINT "2XCu. ";: IF DO THEN PRINT
    "an";: GOTO 4Ø2
4ØØ PRINT "aus";
4Ø2 IF KR THEN 414
4Ø4 VTAB 21: HTAB 22: PRINT "Cur. 1: X=";X1;" Y="Y1
4Ø6 VTAB 22: HTAB 22: PRINT "Cur. 2: X=";X2;" Y="Y2
4Ø8 VTAB 23: HTAB 22: PRINT "HLIN.="; ABS (X1 - X2);"
    VLIN.="; ABS (Y1 - Y2);
41Ø VTAB 24: HTAB 22: PRINT "Cur.-Step =";FA;
412 RETURN
414 VTAB 21: HTAB 22: PRINT "Mitte : X=";XM;" Y="YM;
416 VTAB 22: HTAB 22: PRINT "Winkel : "; INT (36Ø - W);
418 VTAB 23: HTAB 22: PRINT "Step : ";FA;
42Ø VTAB 24: HTAB 22: PRINT "Kr.-Plot : ";: IF P THEN
    PRINT "an";: GOTO 424
422 PRINT "aus";
424 RETURN
426 F = FRE (Ø): GOSUB 16Ø
428 HOME : POKE - 16301,Ø: PRINT "S = SAVE": PRINT "L =
    LOAD": PRINT "Z = Zurück";
430 F = PEEK (T): IF F < 128 THEN 430
432 POKE S,Ø
434 IF F < > 218 THEN 438
436 POKE - 16302,Ø: GOTO 1Ø6
438 ONERR GOTO 1Ø6
440 IF F < > 2Ø4 THEN 462
442 HOME : PRINT "Laden v. Disk (Name der Grafik) ?"
444 PRINT : INPUT N$
446 HOME : PRINT "Laden : ";N$;: PRINT : PRINT "auf Seite
    (1-2) ";
448 F = PEEK (T) - 176: IF F < 1 OR F > 2 THEN 448
450 HOME : PRINT "Lade ";N$;" auf Seite ";F
452 PRINT : PRINT "OK (J/N) ? ";:SE = F * 2ØØØ
454 F = PEEK (T): IF F < 128 THEN 454
456 POKE S,Ø: IF F = 2Ø6 THEN 436
458 IF F < > 2Ø2 THEN 454
460 PRINT CHR$ (13) + CHR$ (4);"BLOAD" + N$ + ",A$" + STR$
    (SE): GOTO 436
462 IF F < > 211 THEN 436
464 HOME : PRINT "Speichern (Name der Grafik) ?"
466 PRINT : INPUT N$
468 HOME : PRINT "Speichern : ";N$;: PRINT : PRINT "von
    Seite (1-2) ";
470 F = PEEK (T) - 176: IF F < 1 OR F > 2 THEN 470
472 SE = F * 2ØØØ: IF F = 2 THEN CALL 25258
474 HOME : PRINT "Speichere ";N$;" von Seite ";F
476 PRINT : PRINT "OK (J/N) ? ";
478 F = PEEK (T): IF F < 128 THEN 478
480 POKE S,Ø: IF F = 2Ø6 THEN 436
482 IF F < > 2Ø2 THEN 478
484 PRINT CHR$ (13) + CHR$ (4);"BSAVE" + N$ +
    ",A$2ØØØ,L$1FF8"
486 IF SE = 4ØØØ THEN CALL 25258
488 GOTO 436
```

Applesoft-Interpreter-Erweiterungen für Double-Hires

von Dr. Wolfgang Braun

Die normalen Applesoft-Hires-Routinen für hochauflösende Grafik sind bekanntlich nicht in der Lage, die Fähigkeit der Apple IIe und IIc zur Darstellung doppelt-hochauflösender Grafik mit 560 * 192 Bildpunkten auszunutzen. Die normalen Hires-Befehle können nur Grafiken mit 280 * 192 Bildpunkten erzeugen.

Warum sollte es aber nicht möglich sein, die normalen Hires-Routinen so zu modifizieren, daß sie auch im doppelt-hochauflösenden Modus funktionieren? Da dem Apple die Fähigkeiten zum Zeichnen von Punkten, Geraden und Linienfolgen (Shapes) in farbiger Darstellung von Hause aus bereits mitgegeben sind, müßte man nur das Vorhandene nutzen und ergänzen und dadurch den erforderlichen Programmieraufwand und den Speicherbedarf gering halten.

Die Aufgabe kann allerdings erst dann als befriedigend gelöst gelten, wenn folgende Bedingungen erfüllt sind:

1. Es müssen sämtliche Applesoft-Befehle, die sich auf die normale Hires-Grafik beziehen, auch für die Double-Hires-Grafik verfügbar sein.
2. Die Befehle müssen in ihrer ursprünglichen Form, d.h. ohne Ampersand-Vektor (&), in Applesoft-Programmen angewendet werden können.

Meine Bemühungen resultierten in einem Assembler-Programm von 700 Bytes Länge, wobei davon nur 480 Bytes für die Erweiterungen der normalen Grafikbefehle im Speicher verfügbar sein müssen. Die restlichen 220 Bytes werden nur zur Initialisierung der Erweiterungen benötigt und danach wieder für Applesoft freigegeben. Das im folgenden abgedruckte Assembler-Programm **AS.DOUBLE.HIRES** leistet das Gewünschte, wenn man es mit „BRUN AS.DOUBLE.HIRES“ aufruft, bevor ein Applesoft-Programm eingegeben oder gestartet wird.

Danach meldet sich Applesoft zurück, und für den Benutzer hat sich nichts verändert, außer daß sich in den Hires-Befehlen der zulässige Bereich für die horizontale X-Koordinate von 0-279 auf 0-559 verdoppelt hat. Jedes Programm, das die für die normale Hires-Grafik ausgelegten Befehle benutzt, wird auch jetzt noch laufen, allerdings werden die Bilder auf die halbe Breite zusammengeschrumpft sein. Um die Erweiterungen voll zu nutzen, sollten die Werte der X-Koordinaten verdoppelt werden.

Es müssen jedoch folgende Einschränkungen gemacht werden:
Die bisherige HGR-Seite 2 ist im Double-Hires-Modus prinzipiell nicht mehr nutzbar, weil die Software-Schalter $C054 und $C055 zum Umschalten zwischen den HGR-Seiten 1 (Speicherbereich $2000-$3FFF) und 2 (Speicherbereich $4000-$5FFF) ihre Funktionen geändert haben. Daher ist der Befehl HGR2 nicht mehr verfügbar.
Unter ProDOS sind die Erweiterungen nicht lauffähig, weil sie die Language-Card benutzen und daher mit ProDOS kollidieren würden. Alle Grafik-Programme, die auf die LC zugreifen (z.B. mit in die LC geschobenem DOS) laufen ebenfalls nicht.

1. Die Grundlagen der Erweiterungen

Um das gesteckte Ziel zu erreichen, war es erforderlich, die normalen Hires-Routinen, die sich im Speicherbereich $F3D8-$F772 des Interpreters befinden, zu modifizieren. Dies geht natürlich nicht im ROM, sondern nur im RAM. Daher wird der gesamte Speicherbereich von $D000-$FFFF, der den Interpreter einschließlich der Hires-Routinen und auch das Monitorprogramm enthält, in die LC kopiert und diese danach aktiviert. Jetzt können an den Hires-Routinen Änderungen vorgenommen werden, ohne daß sich, von der Grafikfähigkeit abgesehen, für den Benutzer etwa ändert.
Um die Hires-Routinen zu manipulieren, mußten zunächst einmal diejenigen Stellen ausfindig gemacht werden, an denen der normale Programmablauf unterbrochen und „Umleitungen“ zu Hilfsroutinen im Arbeitsspeicher angebracht werden konnten. Eine sehr gute Darstellung der Funktionsweise der Applesoft-Hires-Routinen (1) erleichterte mir diese Arbeit. Es ergaben sich insgesamt zehn Ansatzpunkte, an denen drei oder mehr Bytes durch je einen JSR- bzw. JMP-Befehl und wenn nötig noch durch andere Befehle ersetzt wurden. Ein JSR-Befehl leitet den Programmablauf aus der normalen Hires-Routine heraus zu einer Hilfsroutine, nach deren Ausführung das Programm wieder in die Hires-Routine zurückkehrt.

Von zentraler Bedeutung für die Realisierung der doppelt-hochauflösenden Grafik ist die Kenntnis der Software-Schalter, mit deren Hilfe der Double-Hires-Modus eingeschaltet werden kann (2, 3). Im normalen Hires-Modus ist der Bildschirm in 40 Spalten unterteilt, und in jeder Spalte können mit Hilfe eines einzelnen Bytes sieben Bits (Bildpunkte) gesetzt werden. Damit ergeben sich 7 * 40 = 280 Bildpunkte in der Horizontalen.

Im Double-Hires-Modus gehören zu jeder der 40 Spalten zwei adreßgleiche Speicherseiten: eine im Hauptspeicher (MAIN-RAM) und die andere im Hilfsspeicher der 80-Zeichenkarte (Auxiliary-RAM oder AUXRAM). Damit können in jeder der 40 Spalten 14 Bildpunkte gesetzt werden, links die sieben im AUXRAM, rechts daneben die sieben im MAINRAM. Das ergibt 14 * 40 = 560 Bildpunkte in der Horizontalen. Die vertikale Punktanzahl von 192 bleibt unverändert.

Ob ein Punkt in die linke oder in die rechte Hälfte einer der 40 Spalten gesetzt wird, läßt sich mit den bereits erwähnten Soft-

ware-Schaltern bestimmen. Mit z.B. „LDA $C055“ wird die AUXRAM-Speicherseite aktiviert und zugleich die MAINRAM-Speicherseite desaktiviert. Wird danach ein Bit im Bildschirmspeicher gesetzt, so erscheint der Punkt in der linken Hälfte der entsprechenden Spalte. Entsprechendes gilt mit „LDA $C054“ für die rechte Hälfte der Spalte.

Den normalen Hires-Routinen fehlt nun die Fähigkeit, diese Software-Schalter zu betätigen, weil es letztere zum Zeitpunkt des Erscheinens von Applesoft noch nicht gab. Man kann die Hires-Routinen jedoch dadurch überlisten, daß man sie vor der Übergabe der Koordinaten eines im Double-Hires-Modus zu zeichnenden Punktes unterbricht, die X-Koordinate in den normalen Hires-Modus umrechnet, die richtige Speicherseite selektiert und erst dann die nunmehr „normalen“ Koordinaten an die Hires-Routinen zur weiteren Verarbeitung übergibt. Der zu zeichnende Punkt wird dann in der richtigen Spaltenhälfte und an der richtigen Position erscheinen. Die normalen Hires-Routinen „merken“ also gar nicht, daß sie im Double-Hires-Modus arbeiten.
Die Transformation der X-Koordinate eines Punktes vom doppelten zum einfachen Hires-Modus kann durch folgenden einfachen Ausdruck, in dem XDIV7 der ganzzahlige Anteil von X, dividiert durch 7, und REST der Rest bei dieser Division (X MOD 7) bedeutet, geschehen:

normale X-Koordinate = 7 * INT (XDIV7 / 2) + REST

Ist XDIV7 geradzahlig, so muß AUXRAM, ist XDIV7 ungeradzahlig, so muß MAINRAM aktiviert werden. Der Wert von REST, eine Zahl zwischen 0 und 6, bestimmt die Position des gesetzten Bits im Speicher-Byte. Im Programm wird allerdings ein etwas kürzerer Algorithmus verwendet.

Etwas komplizierter liegen die Dinge beim Zeichnen von Geraden mit HPLOT TO oder von Shapes mit DRAW AT, weil im ersten Fall nur die Koordinaten der Start- und Zielpunkte und im zweiten Fall sogar nur die Koordinaten des Startpunktes vorgegeben sind und die Koordinaten der dazwischenliegenden bzw. nachfolgenden Punkte nicht.
In beiden Fällen wird zunächst der vorgegebene Startpunkt geplottet. Danach berechnet eine normale HGR-Routine die Spaltennummer S des nächsten zu zeichnenden Punktes. Das Ergebnis ist nur von der Steigung der Geraden abhängig. Hat sich S im Vergleich mit dem zuletzt gezeichneten Punkt um 1 erhöht bzw. erniedrigt, so wird im normalen Hires-Modus der nächste Punkt in die danebenliegende nächste Spalte gezeichnet.
Im Double-Hires-Modus würde dieser Fall jedoch dazu führen, daß der zuletzt gezeichnete Punkt und der als nächstes zu zeichnende Punkt um 7 Bildpunkte auseinanderliegen, denn die eine Hälfte der Spalte wurde wegen der fehlenden Speicherseitenumschaltung übersprungen.
Es muß also jedesmal, bevor der nächste Punkt gezeichnet werden kann, überprüft werden, ob sich dessen Spaltennummer im Vergleich zu der des Vorgängers geändert hat. Für die Differenz DS der Spaltennummern benachbarter Punkte gibt es 5 mögliche Fälle (erste Spalte = $00, letzte Spalte = $27):

DS = $00 – Beide Punkte liegen in derselben Spaltenhälfte.
DS = $01 – Der nächste Punkt liegt in der benachbarten Spaltenhälfte rechts.
DS = $FF – Der nächste Punkt liegt in der benachbarten Spaltenhälfte links.
DS = $D9 – Der nächste Punkt liegt auf dem linken Rand (Wrap-around, Spalte $00, AUXRAM).
DS = $27 – Der nächste Punkt liegt auf dem rechten Rand (Wrap-around, Spalte $27, MAINRAM).

Nur im ersten Fall ist keine Seitenumschaltung erforderlich; in allen anderen Fällen muß vor dem Zeichnen des nächsten Punktes von einer Seite auf die andere umgeschaltet werden. Zusätzlich muß in den Fällen DS = $01 und DS = $FF die durch die Hires-Routinen bewirkte Erhöhung der Spaltennummer genau dann unterdrückt werden, wenn der Übergang von der einen Spaltenhälfte zur anderen innerhalb derselben Spalte stattfindet.

2. Erläuterungen zum Programm

Der Initialisierungsteil reicht von Zeile 48 bis 188. Die eigentliche Erweiterung für die Applesoft-Hires-Routinen beginnt in der Zeile 208.
In den Zeilen 48 bis 66 wird der ROM-Speicherbereich $D000-$FFFF in die LC kopiert und diese aktiviert. In den Zeilen 71 bis 133 werden die JSR-Befehle, die in den Zeilen 163 bis 188 zu finden sind, an den unmittelbar ablesbaren Stellen in die Hires-Routinen eingetragen. Mit den Zeilen 138 bis 141 wird der zulässige Wertebereich der X-Koordinaten auf 0-559 erweitert, und schließlich wird in den Zeilen 146 bis 154 zum Schutz der Erweiterungsroutinen vor Überschreiben durch Stringvariablen HIMEM auf $9421 gesetzt und die 80-Zeichenkarte aktiviert. Damit ist der Speicherbereich des Initialisierungsteils wieder für Applesoft-Programme zugänglich, und die Hilfsroutinen für die Hires-Befehle mit einigen Variablen stehen ab $9421 bereit.

3. Hinweise zu den einzelnen Befehlen

3.1. HGR

Sobald der Interpreter in einem Applesoft-Programm den Befehl HGR erkennt, springt er zu der Adresse $F3E2. Die ersten 4 Bytes dort wurden ersetzt durch die beiden Befehle

```
JSR  PHGR
     RTS.
```

Demnach springt das Programm aus dem Interpreter heraus nach PHGR (Zeile 395 des Assembler-Programms). Dort werden die Software-Schalter für den Double-Hires-Modus bedient und dann zum Löschen des Bildschirmes nacheinander MAINRAM und AUXRAM aktiviert und jeweils mit Hilfe der Bildschirmlöschroutine HCLR ($F3F2) gelöscht. Danach kehrt das Programm in den Interpreter zurück. In der Zeile 395 kann man „LDA MIX“ ($C053) durch „LDA $C052“ ersetzen, um mit HGR den Bildschirm auf Grafik ohne Text umzuschalten.

Die Anweisungen HCOLOR und TEXT behalten ihre ursprüngliche Bedeutung bei.

3.2. HPLOT X, Y

Nach einem HPLOT-Befehl springt der Interpreter zur Adresse $F6FE. In der Folge würde dann bei Adresse $F705 die normale Routine HPLOT ($F457) zum Zeichnen eines Punktes aufgerufen. Der Befehl „JSR $F457“ ist jetzt jedoch ersetzt durch „JSR PPLOT“. Daher verzweigt das Programm jetzt nach Zeile 208 mit den Koordinaten des zu zeichnenden Punktes in den Registern X, Y und A. Die nächste Routine XTRANSF transformiert die X-Koordinate von doppelter auf einfache Hires-Grafik, und die nachfolgende Routine PAGER aktiviert dann die richtige Speicherseite. Danach werden die Register – jetzt mit den normalen X-Koordinaten und der unveränderten Y-Koordinate – geladen, und die normale Plot-Routine HPLOT zum Zeichnen eines Punktes wird aufgerufen.

3.3. HPLOT X1, Y1 TO X2, Y2

Die Einsprungadresse ist dieselbe wie oben. Die Routine zum Zeichnen einer Geraden zwischen zwei Punkten, HLINE ($F53A), würde bei $F71B aufgerufen. Hier verzweigt das Programm statt dessen zur Hilfsroutine PHLINE in der Zeile 252, die sich selbst erklärt. Die Routine zur

Berechnung der Spaltennummer des nächsten zu zeichnenden Punktes ist INTX ($F465), die von HLINE in $F57D aufgerufen würde. Dort wird statt dessen zur Hilfsroutine PINTX (Zeile 244) verzweigt, wo die Spaltennummer des zuletzt gezeichneten Punktes abgespeichert und die normale Routine INTX aufgerufen wird. Danach steht die neue Spaltennummer im Y-Register. Die Routine SWITCH vergleicht dann die neue und die alte Spaltennummer, aktiviert die entsprechende Speicherseite und kehrt nach HLINE zurück. Daraufhin wird der nächste Punkt gezeichnet wird.
Die Hilfsroutinen PDECRX (Zeile 223) und PINCRX (Zeile 231) verhindern, daß die letzten 7 Bildpunkte vor dem rechten und linken Rand des Bildschirmes nicht geplottet werden.
Auch der Befehl „HPLOT TO X, Y" arbeitet im Double-Hires-Modus in der gleichen Weise, vorausgesetzt daß vorher ein Startpunkt gezeichnet worden ist.

3.4. DRAW/XDRAW N AT X, Y

Die Einsprungstellen für diese beiden Befehle sind $F769 bzw. $F76F. Die Abläufe sind bei beiden Befehlen bis auf die eigentlichen Zeichenroutinen identisch.
Ab Speicherstelle $F763 steht der Befehl „JSR HPOSN" ($F411). HPOSN setzt an Hand der Koordinaten X, Y die internen Cursor-Daten (Adresse des Bildschirm-Bytes, Spaltennummer sowie die Bit-Position) des Startpunktes. Vor dem Aufruf von HPOSN muß daher die X-Koordinate transformiert und die richtige Speicherseite selektiert werden. Dies übernimmt die Routine PDRAW1 (Zeile 314), die jetzt HPOSN ersetzt. Danach kehrt das Programm wieder nach $F766 in die Hires-Routine zurück.
Bis dahin ist auf dem Bildschirm allerdings noch nichts geschehen. Bei den Befehlen DRAW bzw. XDRAW sind LRUD1/LRUD2 ($F4B3/$F4B4) bzw. LRUDX1/LRUDX2 ($F49C/$F49D) die Routinen, die die Punkte auf dem Bildschirm setzen. Vor dem Setzen des Bildschirm-Bytes ist (wie bei HPLOT TO) die richtige Speicherseite zu selektieren. Dazu werden an den Stellen $F49D bzw. $F4B4 Sprünge zu der Hilfsroutine PDRAW (Zeile 331) eingetragen, wo dann das Erforderliche durch SWITCH veranlaßt wird.

Die Befehle ROT und SCALE bleiben in ihren Funktionen unverändert.

4. Schlußbemerkung

Die erforderliche Hardware ist ein Apple IIe mit erweiterter 80-Zeichenkarte, auf der die Drahtbrücke installiert sein muß, oder ein IIc.
Zur Demonstration der Funktionsfähigkeit der beschriebenen Erweiterungen ist ein Applesoft-Programm **AS.DOUBLE.HIRES.DEMO** abgedruckt, in dem die wesentlichen Hires-Befehle im Double-Hires-Modus aufgerufen werden.

Literatur:

(1) C. K. Mesztenyi, All About Applesoft Nr.1, 1981, Seite 92.
(2) K.-W. Bott, Peeker, Heft 2/84, Seite 24.
(3) U. Stiehl, Apple Assembler, Hüthig-Verlag, 1984.

Kurzhinweise

1. Zweck:
Erweiterung der Applesoft-HGR-Befehle für doppelt-hochauflösende Grafik.
2. Konfiguration:
Apple IIe mit 80-Zeichenkarte oder IIc; DOS 3.3 (48K, da LC benutzt wird).
3. Test:
RUN AS.DOUBLE.HIRES.DEMO
4. Sammeldisk:
AS.DOUBLE.HIRES.DEMO
(Applesoft-Demoprogramm)
AS.DOUBLE.HIRES
(Maschinenprogramm)
T.AS.DOUBLE.HIRES
(Big-Mac-Quelltext)

AS.DOUBLE.HIRES setzt einen Apple IIc oder Apple IIe mit 64K-Karte und DOS 3.3 voraus. Kein ProDOS, kein gemovtes DOS 3.3 verwenden!

AS.DOUBLE.HIRES.DEMO

```
10Ø REM AS.DOUBLE.HIRES.DEMO
11Ø PRINT CHR$ (4);"BRUN AS.DOUBLE.HIRES"
12Ø HGR : HCOLOR= 3
13Ø POKE - 16302,Ø: REM NOMIX
14Ø REM Shape-Tabelle ab $3ØØ = 768
15Ø FOR I = 1 TO 57: READ Z: POKE 767 + I,Z: NEXT
16Ø REM Anfangsadresse der Tabelle in $ØØE8/$E9
17Ø POKE 232,Ø: POKE 233,Ø3
18Ø SCALE= 1: ROT= 1
19Ø REM Das Wort 'Text' schreiben
2ØØ FOR I = 2 TO 5
21Ø DRAW I AT 5Ø + I * 7,86: NEXT
22Ø REM Rahmen zeichnen
23Ø HPLOT Ø,Ø TO 559,Ø TO 559,191
24Ø HPLOT TO Ø,191 TO Ø,Ø
25Ø REM Bewegung
26Ø SCALE= 85
27Ø FOR K = 1 TO 64: ROT= K
28Ø XDRAW 1 AT 279,96: NEXT
29Ø REM Mit beliebiger Taste abbrechen
3ØØ IF PEEK (49152) < 127 THEN GOTO 27Ø
31Ø TEXT : POKE 49168,Ø
32Ø DATA 5,Ø,14,Ø,16,Ø,23,Ø,33,Ø,48,Ø,57,Ø,4,Ø
33Ø DATA 45,45,159,54,54,38,Ø
34Ø DATA 146,146,32,1ØØ,173,62,183,45,4,Ø
35Ø DATA 146,146,13,24,14,88,4Ø,4Ø,248,27,21,85,17Ø,4,Ø
36Ø DATA 137,54,54,46,12,24,248,4,Ø
```

AS.DOUBLE.HIRES

BSAVE AS.DOUBLE.HIRES, A$9344, L$Ø2BC

```
1    ********************************
2    *                              *
3    *       AS.DOUBLE.HIRES        *
4    *                              *
5    * Modifikation der Applesoft- *
6    * Hires-Routinen für doppelt- *
7    * hochauflösende Grafik mit   *
8    *   56Ø * 192 Bildpunkten      *
9    *                              *
1Ø   * von Dr.W. Braun, Juni 1985  *
11   *                              *
12   ********************************
13
14            ORG  $9344
15
16   HIMEM    EQU  $73
17   DX       EQU  $DØ
18   QDRNT    EQU  $D3
19   E        EQU  $D4
2Ø   XØ       EQU  $EØ
21   HNDX     EQU  $E5
22   HPAG     EQU  $E6
23
24   * Software-Schalter
25
26   STORE8Ø  EQU  $CØØ1
27   COL8Ø    EQU  $CØØD
28   STAT     EQU  $CØ1C
29   GRAFIK   EQU  $CØ5Ø
3Ø   MIX      EQU  $CØ53
31   MAINRAM  EQU  $CØ54
32   AUXRAM   EQU  $CØ55
33   HIRES    EQU  $CØ57
34   AN3      EQU  $CØ5E
35
36   * Applesoft-HGR-Routinen
37
38   INTX     EQU  $F465
39   HPLOT    EQU  $F457
4Ø   HCLR     EQU  $F3F2
```

```
                        41  HLINENEU EQU  $F55A
                        42  HPOSN    EQU  $F411
                        43
                        44  * Move-Routinen
                        45
                        46  * ROM $D000-$FFFF in LC moven *
                        47
9344: 8D 89 C0          48          STA  $C089
9347: 8D 89 C0          49          STA  $C089
934A: A9 00             50          LDA  #$00
934C: 85 E0             51          STA  X0
934E: A9 D0             52          LDA  #$D0
9350: 85 E1             53          STA  X0+1
9352: A0 00             54          LDY  #$00
9354: B1 E0             55  NEXT    LDA  (X0),Y
9356: 91 E0             56          STA  (X0),Y
9358: C8                57          INY
9359: D0 F9             58          BNE  NEXT
935B: E6 E1             59          INC  X0+1
935D: A5 E1             60          LDA  X0+1
935F: D0 F3             61          BNE  NEXT
                        62
                        63  * Read/Write LC Bank 1
                        64
9361: 8D 8B C0          65          STA  $C08B
9364: 8D 8B C0          66          STA  $C08B
                        67
                        68  * Hilfsroutinen in die Apple-
                        69  * soft-HGR-Routinen eintragen
                        70
9367: A0 04             71          LDY  #$04       ;PDECRX
9369: B9 F8 93          72  LOOP1   LDA  P1,Y
936C: 99 71 F4          73          STA  $F471,Y
936F: 88                74          DEY
9370: 10 F7             75          BPL  LOOP1
                        76
9372: A0 02             77          LDY  #$02       ;PINCRX
9374: B9 FD 93          78  LOOP2   LDA  P2,Y
9377: 99 93 F4          79          STA  $F493,Y
937A: 88                80          DEY
937B: 10 F7             81          BPL  LOOP2
                        82
937D: A0 02             83          LDY  #$02       ;PINT
937F: B9 00 94          84  LOOP3   LDA  P3,Y
9382: 99 7D F5          85          STA  $F57D,Y
9385: 88                86          DEY
9386: 10 F7             87          BPL  LOOP3
                        88
9388: A0 02             89          LDY  #$02       ;PPLOT
938A: B9 03 94          90  LOOP4   LDA  P4,Y
938D: 99 05 F7          91          STA  $F705,Y
9390: 88                92          DEY
9391: 10 F7             93          BPL  LOOP4
                        94
9393: A0 09             95          LDY  #$09       ;PHLINE
9395: B9 06 94          96  LOOP5   LDA  P5,Y
9398: 99 15 F7          97          STA  $F715,Y
939B: 88                98          DEY
939C: 10 F7             99          BPL  LOOP5
                        100
939E: A9 0C             101         LDA  #$0C       ;RTS
93A0: 8D 0E F7          102         STA  $F70E
                        103
93A3: A0 03             104         LDY  #$03       ;PHGR
93A5: B9 10 94          105 LOOP6   LDA  P6,Y
93A8: 99 E2 F3          106         STA  $F3E2,Y
93AB: 88                107         DEY
93AC: 10 F7             108         BPL  LOOP6
                        109
93AE: A0 02             110         LDY  #$02       ;PDRAW1
93B0: B9 14 94          111 LOOP7   LDA  P7,Y
93B3: 99 63 F7          112         STA  $F763,Y
93B6: 88                113         DEY
93B7: 10 F7             114         BPL  LOOP7
                        115
93B9: A0 03             116         LDY  #$03       ;PDRAW
93BB: B9 17 94          117 LOOP8   LDA  P8,Y
93BE: 99 9D F4          118         STA  $F49D,Y
93C1: 99 B4 F4          119         STA  $F4B4,Y
93C4: 88                120         DEY
93C5: 10 F4             121         BPL  LOOP8
                        122
93C7: A0 02             123         LDY  #$02       ;AUSDRAW
93C9: B9 1B 94          124 LOOP9   LDA  P9,Y
93CC: 99 5A F6          125         STA  $F65A,Y
93CF: 88                126         DEY
93D0: 10 F7             127         BPL  LOOP9
                        128
93D2: A0 02             129         LDY  #$02       ;AUSXDRAW
93D4: B9 1E 94          130 LOOP10  LDA  P10,Y
93D7: 99 B6 F6          131         STA  $F6B6,Y
93DA: 88                132         DEY
93DB: 10 F7             133         BPL  LOOP10
                        134
                        135 * X-Bereich von (0..279)
                        136 * auf (0..559) erweitern
                        137
93DD: A9 02             138         LDA  #>560
93DF: 8D C4 F6          139         STA  $F6C4
93E2: A9 30             140         LDA  #<560
93E4: 8D CA F6          141         STA  $F6CA
                        142
                        143 * HIMEM zum Schutz der Hilfs-
                        144 * routinen neu setzen
                        145
93E7: A9 21             146         LDA  #<XSTARTL
93E9: 85 73             147         STA  HIMEM
93EB: A9 94             148         LDA  #>XSTARTL
93ED: 85 74             149         STA  HIMEM+1
                        150
                        151 * 80-Zeichenkarte über Outport aktivieren
                        152
93EF: A9 03             153         LDA  #03
93F1: 20 95 FE          154         JSR  $FE95
                        155
                        156 * LC-Schreibschutz
                        157
93F4: 8D 88 C0          158         STA  $C088
93F7: 60                159         RTS
                        160
                        161 * Hilfsroutinen
                        162
93F8: 20 4F 94          163 P1      JSR  PDECRX
93FB: EA                164         NOP
93FC: EA                165         NOP
                        166
93FD: 20 5C 94          167 P2      JSR  PINCRX
                        168
9400: 20 6C 94          169 P3      JSR  PINTX
                        170
9403: 20 2D 94          171 P4      JSR  PPLOT
                        172
9406: 20 76 94          173 P5      JSR  PHLINE
9409: 4C 08 F7          174         JMP  $F708
940C: 8D 54 C0          175         STA  MAINRAM
940F: 60                176         RTS
                        177
9410: 20 72 95          178 P6      JSR  PHGR
9413: 60                179         RTS
                        180
9414: 20 E4 94          181 P7      JSR  PDRAW1
                        182
9417: 20 0B 95          183 P8      JSR  PDRAW
941A: EA                184         NOP
                        185
941B: 4C F0 95          186 P9      JMP  AUSDRAW
                        187
941E: 4C F7 95          188 P10     JMP  AUSXDRAW
                        189
                        190 * Hierher springen die Apple-
                        191 * soft-HGR-Routinen
                        192
9421: 00                193 XSTARTL HEX  00
9422: 00                194 XSTARTH HEX  00
9423: 00                195 XZIELL  HEX  00
9424: 00                196 XZIELH  HEX  00
9425: 00                197 XDIV7   HEX  00
9426: 00                198 REST    HEX  00
9427: 00                199 SEITE   HEX  00
9428: 00                200 XL      HEX  00
9429: 00                201 XH      HEX  00
942A: 00                202 YSTART  HEX  00
942B: 00                203 YZIEL   HEX  00
942C: 00                204 YOLD    HEX  00
                        205
                        206 * Plotten eines einzelnen Punktes
                        207
942D: 8E 21 94          208 PPLOT   STX  XSTARTL
9430: 8C 22 94          209         STY  XSTARTH
9433: 8D 2A 94          210         STA  YSTART
9436: 8E 28 94          211 PLOTS   STX  XL
9439: 8C 29 94          212         STY  XH
943C: 20 15 95          213         JSR  XTRANSF
943F: 20 98 95          214         JSR  PAGER
9442: AE 28 94          215         LDX  XL
9445: AC 29 94          216         LDY  XH
9448: AD 2A 94          217         LDA  YSTART
944B: 20 57 F4          218         JSR  HPLOT
944E: 60                219         RTS
                        220
                        221 * Randspalte plotten
                        222
944F: 88                223 PDECRX  DEY
9450: 10 09             224         BPL  LBL14
```

```
9452: AØ FF     225           LDY  #$FF
9454: AD 1C CØ  226           LDA  STAT
9457: 1Ø Ø2     227           BPL  LBL14
9459: AØ 27     228           LDY  #$27
945B: 6Ø        229 LBL14     RTS
                23Ø
945C: C8        231 PINCRX    INY
945D: CØ 28     232           CPY  #$28
945F: DØ ØA     233           BNE  LBL12
9461: 48        234           PHA
9462: AD 1C CØ  235           LDA  STAT
9465: 1Ø Ø3     236           BPL  LBL7
9467: AØ 28     237           LDY  #$28
9469: 18        238           CLC
946A: 68        239 LBL7      PLA
946B: 6Ø        24Ø LBL12     RTS
                241
                242 * Speicherseite schalten
                243
946C: 8C 2C 94  244 PINTX     STY  YOLD
946F: 2Ø 65 F4  245           JSR  INTX
9472: 2Ø A6 95  246           JSR  SWITCH
9475: 6Ø        247           RTS
                248
                249 * Differenz der X-Koordinaten von
                25Ø * Start- und Zielpunkt berechnen
                251
9476: 8E 23 94  252 PHLINE    STX  XZIELL
9479: 8C 24 94  253           STY  XZIELH
947C: 8D 2B 94  254           STA  YZIEL
947F: 8E 28 94  255           STX  XL
9482: 8C 29 94  256           STY  XH
9485: 38        257           SEC
9486: AD 23 94  258           LDA  XZIELL
9489: ED 21 94  259           SBC  XSTARTL
948C: 48        26Ø           PHA
948D: AD 24 94  261           LDA  XZIELH
9490: ED 22 94  262           SBC  XSTARTH
9493: 85 D3     263           STA  QDRNT
9495: BØ ØA     264           BCS  LBL2Ø
9497: 68        265           PLA
9498: 49 FF     266           EOR  #$FF
949A: 69 Ø1     267           ADC  #$Ø1
949C: 48        268           PHA
949D: A9 ØØ     269           LDA  #$ØØ
949F: E5 D3     27Ø           SBC  QDRNT
94A1: 85 D1     271 LBL2Ø     STA  DX+1
94A3: 85 D5     272           STA  E+1
94A5: 68        273           PLA
94A6: 85 DØ     274           STA  DX
94A8: 85 D4     275           STA  E
                276
                277 * Startpunkt zeichnen
                278
94AA: AE 21 94  279           LDX  XSTARTL
94AD: AC 22 94  28Ø           LDY  XSTARTH
94BØ: AD 2A 94  281           LDA  YSTART
94B3: 2Ø 36 94  282           JSR  PLOTS
                283
                284 * X-Koordinaten des Ziel-
                285 * punktes transformieren
                286
94B6: AD 23 94  287           LDA  XZIELL
94B9: AE 24 94  288           LDX  XZIELH
94BC: 8D 28 94  289           STA  XL
94BF: 8E 29 94  29Ø           STX  XH
94C2: 2Ø 15 95  291           JSR  XTRANSF
                292
                293 * Zielpunkt -> Startpunkt
                294 * für nächstes TO
                295
94C5: AD 23 94  296           LDA  XZIELL
94C8: AE 24 94  297           LDX  XZIELH
94CB: AC 2B 94  298           LDY  YZIEL
94CE: 8C 2A 94  299           STY  YSTART
94D1: 8D 21 94  3ØØ           STA  XSTARTL
94D4: 8E 22 94  3Ø1           STX  XSTARTH
                3Ø2
                3Ø3 * Register laden
                3Ø4 * und Linie zeichnen
                3Ø5
94D7: AD 28 94  3Ø6           LDA  XL
94DA: AE 29 94  3Ø7           LDX  XH
94DD: AC 2B 94  3Ø8           LDY  YZIEL
94EØ: 2Ø 5A F5  3Ø9           JSR  HLINENEU
94E3: 6Ø        31Ø           RTS
                311
                312 * Interne Cursordaten für DRAW setzen
                313
94E4: 8E 21 94  314 PDRAW1    STX  XSTARTL
94E7: 8C 22 94  315           STY  XSTARTH
94EA: 8E 28 94  316           STX  XL
```

```
94ED: 8C 29 94  317           STY  XH
94FØ: 8D 2A 94  318           STA  YSTART
94F3: 2Ø 15 95  319           JSR  XTRANSF
94F6: 2Ø 98 95  32Ø           JSR  PAGER
94F9: AE 28 94  321           LDX  XL
94FC: AC 29 94  322           LDY  XH
94FF: AD 2A 94  323           LDA  YSTART
95Ø2: 2Ø 11 F4  324           JSR  HPOSN
95Ø5: A5 E5     325           LDA  HNDX
95Ø7: 8D 2C 94  326           STA  YOLD
95ØA: 6Ø        327           RTS
                328
                329 * Speicherumschaltung bei DRAW, XDRAW
                33Ø
95ØB: Ø8        331 PDRAW     PHP
95ØC: 2Ø A6 95  332           JSR  SWITCH
95ØF: 28        333           PLP
951Ø: A5 D1     334           LDA  DX+1
9512: 29 Ø4     335           AND  #$Ø4
9514: 6Ø        336           RTS
                337
                338 * X-Koordinate auf einfache
                339 * HIRES-Grafik transformieren
                34Ø *
                341 * Nach XTRANSF befindet sich
                342 * die transformierte X-Koor-
                343 * dinate in XL/XH
                344
9515: AØ ØØ     345 XTRANSF   LDY  #$ØØ
9517: 8C 27 94  346           STY  SEITE
951A: 38        347           SEC
951B: AD 28 94  348 LOOP3Ø    LDA  XL
951E: E9 Ø7     349           SBC  #$Ø7
952Ø: 8D 28 94  35Ø           STA  XL
9523: AD 29 94  351           LDA  XH
9526: E9 ØØ     352           SBC  #$ØØ
9528: 8D 29 94  353           STA  XH
952B: C8        354           INY
952C: FØ Ø3     355           BEQ  LBL6
952E: BØ EB     356           BCS  LOOP3Ø
953Ø: 88        357           DEY
9531: 8C 25 94  358 LBL6      STY  XDIV7
9534: 18        359           CLC
9535: AD 28 94  36Ø           LDA  XL
9538: 69 Ø7     361           ADC  #$Ø7
953A: 8D 26 94  362           STA  REST
                363
                364 * XL = (XKOOR1 + REST) * Ø,5
                365 * oder
                366 * XL = (XKOOR1 + REST - 6) * Ø,5
                367
953D: AD 25 94  368           LDA  XDIV7
954Ø: 4A        369           LSR
9541: 2E 27 94  37Ø           ROL  SEITE
9544: AD 26 94  371           LDA  REST
9547: 18        372           CLC
9548: 6D 21 94  373           ADC  XSTARTL
954B: 8D 28 94  374           STA  XL
954E: A9 ØØ     375           LDA  #$ØØ
955Ø: 6D 22 94  376           ADC  XSTARTH
9553: 8D 29 94  377           STA  XH
9556: AC 27 94  378           LDY  SEITE
9559: FØ ØE     379           BEQ  NULL
955B: 38        38Ø           SEC
955C: AD 28 94  381           LDA  XL
955F: E9 Ø6     382           SBC  #$Ø6
9561: 8D 28 94  383           STA  XL
9564: AD 29 94  384           LDA  XH
9567: E9 ØØ     385           SBC  #$ØØ
9569: 18        386 NULL      CLC
956A: 4A        387           LSR
956B: 8D 29 94  388           STA  XH
956E: 6E 28 94  389           ROR  XL
9571: 6Ø        39Ø           RTS
                391
                392 * Simulation des HGR-Befehles für
                393 * doppelt-hochauflösende Grafik
                394
9572: AD 53 CØ  395 PHGR      LDA  MIX
9575: AD 57 CØ  396           LDA  HIRES
9578: AD 5Ø CØ  397           LDA  GRAFIK
957B: 8D Ø1 CØ  398           STA  STORE8Ø
957E: 8D ØD CØ  399           STA  COL8Ø
9581: AD 5E CØ  4ØØ           LDA  AN3
9584: A9 2Ø     4Ø1           LDA  #$2Ø
9586: 85 E6     4Ø2           STA  HPAG
9588: 8D 54 CØ  4Ø3           STA  MAINRAM
958B: 2Ø F2 F3  4Ø4           JSR  HCLR
958E: 8D 55 CØ  4Ø5           STA  AUXRAM
9591: 2Ø F2 F3  4Ø6           JSR  HCLR
9594: 8D 54 CØ  4Ø7           STA  MAINRAM
9597: 6Ø        4Ø8           RTS
```

```
                409
                410  * Speicherumschaltung für das
                411  * Plotten eines Punktes
                412
9598: AD 27 94  413  PAGER    LDA  SEITE
959B: F0 05     414          BEQ  LBL1
959D: 8D 54 C0  415          STA  MAINRAM
95A0: D0 03     416          BNE  LBL2
95A2: 8D 55 C0  417  LBL1     STA  AUXRAM
95A5: 60        418  LBL2     RTS
                419
                420  * Speicherumschaltung für das
                421  * Zeichnen von Geraden mit
                422  * HPLOT, DRAW oder XDRAW
                423
95A6: 98        424  SWITCH   TYA
95A7: 38        425          SEC
95A8: ED 2C 94  426          SBC  YOLD
95AB: C9 00     427          CMP  #$00
95AD: F0 40     428          BEQ  LBL3
95AF: C9 01     429          CMP  #$01
95B1: D0 14     430          BNE  LBL4
95B3: AD 1C C0  431          LDA  STAT       ;DY = 1
95B6: 10 09     432          BPL  LBL5
95B8: AC 2C 94  433          LDY  YOLD       ;AUXRAM on
95BB: 8C 54 C0  434          STY  MAINRAM
95BE: 4C EC 95  435          JMP  LBL10
95C1: 8D 55 C0  436  LBL5     STA  AUXRAM
95C4: 4C EC 95  437          JMP  LBL10
```

```
95C7: C9 FF     438  LBL4     CMP  #$FF
95C9: D0 14     439          BNE  LBL8
95CB: AD 1C C0  440          LDA  STAT       ;DY = -1
95CE: 30 09     441          BMI  LBL9
95D0: AC 2C 94  442          LDY  YOLD
95D3: 8D 55 C0  443          STA  AUXRAM
95D6: 4C EC 95  444          JMP  LBL10
95D9: 8D 54 C0  445  LBL9     STA  MAINRAM
95DC: 4C EC 95  446          JMP  LBL10
95DF: C9 D9     447  LBL8     CMP  #$D9
95E1: D0 06     448          BNE  LBL11
95E3: 8D 55 C0  449          STA  AUXRAM     ;Spalte $27->$0
95E6: 4C EC 95  450          JMP  LBL10
95E9: 8D 54 C0  451  LBL11    STA  MAINRAM    ;Spalte $0->$27
95EC: 8C 2C 94  452  LBL10    STY  YOLD
95EF: 60        453  LBL3     RTS
                454
                455  * Rücksprung in Interpreter
                456
95F0: F0 03     457  AUSDRAW  BEQ  LB1
95F2: 4C 26 F6  458          JMP  $F626
95F5: F0 05     459  LB1      BEQ  LB2
                460
95F7: F0 03     461  AUSXDRAW BEQ  LB2
95F9: 4C 82 F6  462          JMP  $F682
95FC: 8D 54 C0  463  LB2      STA  MAINRAM
95FF: 60        464          RTS

700 Bytes
```

Tips zur Konvertierung der Adreß-verwaltungs-Dateien von DOS 3.3 nach CP/M

(s. Peeker, Heft 8/85, S. 40)

Da einige Benutzer der Peeker-Sammeldisketten Schwierigkeiten bei der o.g. Konvertierung und dem darauffolgendem „Anwerfen" des Programms hatten, möchte ich an dieser Stelle ein paar Tips verraten. Verfährt man nach diesem Rezept, dürfte eigentlich nichts mehr schief gehen.

1. Man formatiert sich im CP/M mit dem FORMAT-Programm, welches sich auf der CP/M Masterdisk befindet, eine Leerdiskette. Danach bringt man mit dem COPY-Programm (ebenfalls auf der Masterdisk) mit dem Befehl COPY B:=A:/S das CP/M-Betriebssystem auf die formatierte Diskette.
   (In Laufwerk A: befindet sich die Masterdiskette, in Laufwerk B: die Leerdiskette.)
2. Man ruft das PIP-Programm von der Masterdiskette auf und bringt mit dem Befehl B:APDOS.COM=A:APDOS-.COM das DOS-CP/M-Konvert-Programm auf die Leerdiskette.
   (Gleiche Laufwerksbelegung wie oben).
3. Man ruft von der Leerdiskette, auf der sich jetzt das Konvert-Programm befindet, APDOS auf. Mit dem Befehl CAT D2 werden die DOS-Programme auf der Diskette in Laufwerk B: angezeigt.
   (In Laufwerk A: befindet sich jetzt die CP/M-Leerdiskette mit APDOS und in Laufwerk B: die DOS-Diskette.)
4. Man könnte nun die Command-Files der Reihe nach von DOS nach CP/M konvertieren. Leider läßt das Konvert-Programm zur Benennung der CP/M-Files nur max. 7 Buchstaben zu. Ein kleiner Trick hilft weiter. Man benennt die Textdateifiles auf der CP/M-Seite einfach der Reihe nach mit A, B, C usw., also
   A.CMD=AD-START. CMD;
   B.CMD=HMENUE.CMD;
   C.CMD=AUFNAHME.CMD usw.
   Dabei steht links vom Gleichheitszeichen der CP/M-Filename und rechts der DOS-Filename. Hat man auf diese Art alle CMD-Dateien von der DOS-Diskette konvertiert, so benennt man mit dem RENAME-Kommando die CP/M-Dateien wieder in ihren ursprünglichen Namen um, also
   REN AD-START.CMD=A.CMD;
   REN HMENUE.CMD=B.CMD usw.
   Nach dieser Prozedur sind die Command-Dateien wieder für das Adreß-verwaltungsprogramm lesbar.

Nun muß auf die so vorbereitete Diskette natürlich noch Ihr dBASE kopiert werden. Anschließend muß, wie in Heft 8/85 auf Seite 41 (Aufbau der Datenbank) beschrieben, auf einer zweiten CP/M-Diskette der File ADRESSEN.DBF kreiert werden. Wenn Sie das ADV zum ersten Mal anwerfen, dürfen Sie nicht mit AD-START.CMD starten, da das Programm sonst die Index- und Memorydateien sucht, nicht findet und sich mit einer Fehlermeldung verabschiedet. Diese Dateien werden ja erst bei der Eingabe der Daten erzeugt. Der „jungfräuliche" Start muß mit HMENUE erfolgen. Sind Daten erstmals eingegeben, so erfolgt der Start mit AD-START. Vom CP/M-System erfolgt der Start mit DBASE AD-START. Bequemer ist es, das Programm mit diesem Kommando selbststartend zu machen.

Nachtrag zum Adreß-verwaltungsprogramm

(Fehler im Programm und deren Beseitigung)

Leider haben sich beim längeren Einsatz des Programms zwei Fehler gezeigt. Im Programm SCHREIBA.CMD muß es unter der Zeile CASE ANR='5' anstelle von STORE (TRIM(VOR)+' + '+TRIM(VORE)) TO NA richtig heißen: STORE (TRIM (VOR)+' + '+TRIM(VORE)+' '+FAM) TO NA. Es erscheint sonst bei der Option 5 (Anrede „Herr+Frau") nicht der gemeinsame Familienname.
Im Programm SUCH.CMD muß über der Zeile DO WHILE #=0 die Zeile STORE F TO ENDE und unter der Zeile IF MFAM=' ' die Zeile STORE T TO ENDE eingeschoben werden. In allen Programmen, welche SUCH.CMD benutzen (EDITFNAM.CMD, SUCHFNAM.CMD, SCHREIBA.CMD und LOESCH.CMD), sind unter der Zeile DO SUCH die folgenden Zeilen einzufügen:

```
IF ENDE
   RETURN
ENDIF
```

Ohne diese Änderungen verhält sich das Programm in einigen Situationen merkwürdig.

Ernst Fischer

Die neuen ROMs für den Apple IIe

von Ulrich Stiehl

Teil 1: Grundlagen

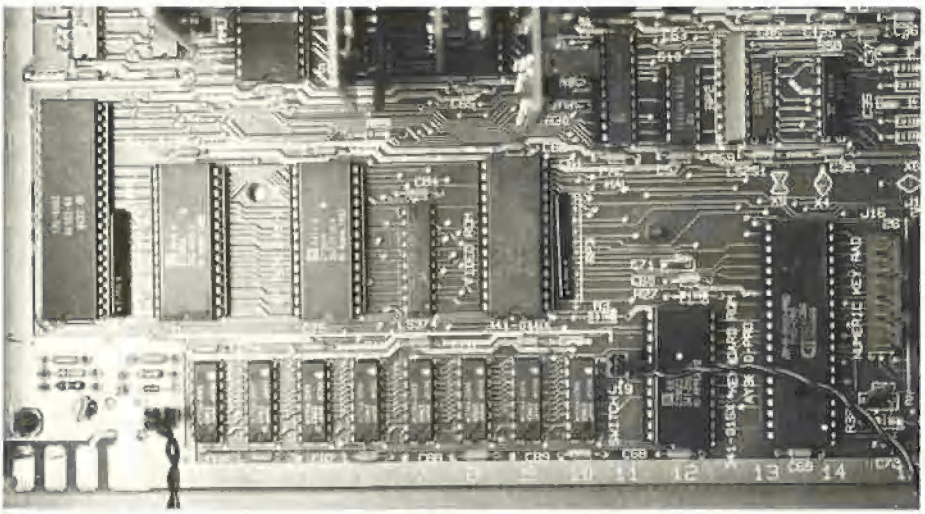


Abb. 1: Apple-IIe-Platine

Die neuen ROMs für den Apple IIe (= sog. „Enhanced Apple IIe" oder neuer IIe) werden erst ab etwa Dezember 1985 in Deutschland ausgeliefert. Deshalb sollte eigentlich diese Serie erst mit dem Novemberheft beginnen. Da jedoch in der „Apple's", Heft 9/85, S. 20 ff. ein völlig irreführender Bericht erschien, der zu unnötigen Mißverständnissen und Rückfragen führen dürfte, habe ich die Serie vorziehen müssen. In der „Apple's" wird beispielsweise behauptet:

„Ohne auch nur einen Steckplatz einzubüßen, verwandelt sich der IIe damit zu einem IIc". – Richtig ist, daß die Routinen in den Bereichen $C100-$CFFF (Internes ROM), $D000-$F7FF (Interpreter-ROM) und $F800-$FFFF (Monitor-ROM) beim IIc und neuen IIe verschieden sind.

„ESC-R wandelt alle Buchstaben, die über die Tastatur eingegeben werden, in Großbuchstaben um." – Richtig ist, daß ESC-R und ESC-T beim neuen IIe eliminiert wurden.

„Besitzer eines 65C02-IIe können alle IIc-Routinen nutzen, die mit mausgesteuerter Grafik arbeiten." – Richtig ist, daß die im IIc ab $C400 ff. enthaltenen Mausroutinen nicht beim neuen IIe vorhanden sind. Es ist lediglich der Mauszeichensatz implementiert.

„Er (d.h. der Mini-Assembler) unterstützt auch die neuen Instruktionen des 65C02." – Richtig ist, daß der Mini-Assembler dies nicht tut.

„Im Find-Modus des Monitors erkennt der Rechner diese Zeichen (gemeint „HALLO", die die ASCII-Codes ersetzen." – Richtig ist, daß die neue Suchroutine nicht für Zeichenfolgen gedacht ist und konkret „'H 'A 'L 'L 'O<300.3FFS" nie gesucht werden könnte. Die diesbezüglichen Angaben sind auch in den amerikanischen Originalunterlagen „About Your Enhanced Apple IIe. Programmer's Guide", S. 14, falsch dargestellt worden.

Außerdem wird in dem zitierten Aufsatz permanent der Eindruck erweckt, als seien die diversen Verbesserungen, z.B. die Erhöhung der Scroll-Geschwindigkeit um 30%, u.a. auf die zusätzlichen Befehle des 65C02-Prozessors zurückzuführen. – Richtig ist, daß in den gesamten ROMs kein einziger spezifischer 65C02-Befehl (wie PHY, PLY, PHX, PLX, TSB, TRB usw.) vorkommt und deshalb weder der Disassembler noch der Mini-Assembler diesbezüglich erweitert wurden.

Die unreflektierte Wiedergabe von Werbeunterlagen hat schon so manchen Autor aufs Glatteis geführt. Deshalb bringe ich nachfolgend nur das, was ich selbst ermittelt und ausprobiert habe.

1. Apple IIe/c/+

Zum „Enhanced Apple IIe" werden neben einer Bedienungsanleitung („Programmer's Guide") vier neue Bauelemente für den Apple IIe geliefert:

1 65C02-Prozessor
1 Video-ROM
1 CD-ROM
1 EF-ROM

Um Mißverständnisse auszuschließen: Die neue CD- und EF-ROMs (CD = $C100-$DFFF; EF = $E000-$FFFF) würden nicht im Apple IIc funktionieren, weil der sog. CX-Bereich ($C100-$CFFF) beim IIc mangels Slots nicht doppelt vorkommt. Auch würden sie nicht im Apple II+ funktionieren, weil beim II+ der interne CX-Bereich fehlt. Die CD- und EF-ROMs sind also einzig und allein für den Apple IIe konzipiert.

2. Installation

Die Bauelemente sollten von einem autorisierten Apple-Händler ausgewechselt werden. Wer den Austausch selbst vornehmen will, gehe wie folgt vor (s.a. **Abb. 1:** Apple-IIe-Platine):

1. Apple IIe ausschalten, Deckel abnehmen und diejenigen Interface-Karten entfernen, die über die Mitte der Platine hinausragen.
2. Direkt hinter der Tastatur findet man in der ersten Reihe die 8 RAMs für den 64K-RAM-Speicher. Wenn man über die Tastatur auf die Platine schaut, sieht man dann in der zweiten Reihe *von links nach rechts* mit der folgenden Beschriftung in Großbuchstaben:
   – 6502A
   – CD-ROM
   – EF-ROM
   – VIDEO-ROM

(Zwischen EF-ROM und VIDEO-ROM ist ein weiteres, kleines Bauelement, das auf der Platine verbleibt.)

3. Man nehme die alten Bauelemente mit einem flachen Messer oder besser mit einer IC-Zange vorsichtig aus den Fassungen. Es empfiehlt sich, die alten ROMs zu beschriften und für eine eventuelle, zukünftige Verwendung auf einer Antistatikunterlage aufzubewahren.

4. Dann setze man die neuen Bauelemente mit den *Einkerbungen in Richtung zur Tastatur* in die leeren Fassungen. Fest eindrücken, aber dabei keine Beinchen abbrechen! Interface-Karten wieder einsetzen und dann den Strom einschalten. Jetzt müßte der Apple IIe mit „Apple //e" booten.

Das geschilderte Verfahren bezieht sich auf den europäischen Apple IIe (erkennbar am Auxiliary Slot in der Mitte der Platine), denn der amerikanische Apple IIe ist teils abweichend konstruiert.

Der 65C02-Prozessor, der übrigens ein GTE-65SC02 und kein NCR-65C02 ist, obwohl im Handbuch das Datenblatt von NCR abgedruckt wurde, läßt sich bei einem älteren Apple IIe möglicherweise nur extrem schwer auswechseln. Zumindest saß der alte 6502A in meinem Apple IIe förmlich wie festgeklebt in der Fassung. Als ich ihn dann zu Hause mangels IC-Zange schließlich doch mit einem kleinen Schraubenzieher herausgehebelt hatte, bekam die Platine unter der Fassung einen Haarriß mit der Folge, daß nunmehr weder der alte noch der neue Prozessor funktionierte. Dank meiner Accelerator-IIe-Karte kann ich jedoch auch ohne Prozessor auf der Platine booten. So konnte ich noch vor der Reparatur mit dem Testen beginnen.

Es besteht übrigens keine Notwendigkeit, *alle* vier Bauelemente auszutauschen. Es ist vielmehr jede Kombination zulässig mit der Ausnahme, daß *entweder* die alten *oder* die neuen CD-EF-ROMs verwendet werden müssen. Beispiele:

1. Alle 4 neuen Bauelemente einsetzen als Normalfall.

2. Nur die neuen Bauelemente 65SC02, CD-ROM und EF-ROM einsetzen und auf das neue Video-ROM verzichten, wenn man den Mauszeichensatz nicht verwenden kann.

3. Nur die neuen CD- und EF-ROMs einbauen und damit zusätzlich den alten 6502A in der Fassung belassen. Dies ist möglich, weil die neuen CD- und EF-ROMs keine 65C02-Befehle verwenden.

4. Nur das Video-ROM einbauen, weil man den Mauszeichensatz benötigt.

3. 65C02-Prozessor

Im Peeker, Heft 1/84, S. 6-25 haben wir uns bereits ausführlich mit den neuen Befehlen des 65C02 befaßt. Außerdem sind bereits Macros für die 65C02-Spezialbefehle veröffentlicht worden (Heft 4/85, S. 30-33), so daß wir an dieser Stelle auf den erweiterten Instruktionssatz nicht einzugehen brauchen. Zwei Dinge sind jedoch wichtig zu wissen:
1. Der GTE-65SC02 ist nur mit 1 MHz getaktet, so daß von daher keine Geschwindigkeitsvorteile entstehen.
2. Soweit ich nicht irgendeine Stelle übersehen habe, wird im *gesamten* System-ROM, d.h. $C100-$CFFF (= internes CX-ROM), $D000-$F7FF (= Applesoft-Interpreter) und $F800-$FFFF (= Monitor), niemals von einem der zusätzlichen 65C02-Befehle wie PHY, PHX, PLY, PLX, TSB, TRB usw. Gebrauch gemacht. Sinngemäß wurden auch weder ein 65C02-Disassembler noch ein 65C02-Mini-Assembler implementiert. Dies steht im krassen Widerspruch zum Apple IIc, bei dem es im System-ROM an 65C02-Sonderbefehlen wimmelt. Über den Grund kann man nur Mutmaßungen anstellen. Wie dem auch sei, wenn Ihnen Ihr Apple-Händler erzählt, daß der neue Apple IIe schneller läuft, *weil* er einen 65C02-Prozessor hat, so werden Sie jetzt nur noch müde lächeln können. Einige ROM-Routinen, insbesondere die Bildschirmsteuerung, sind tatsächlich nunmehr schneller, doch nicht wegen des 65C02, sondern weil ein gewisser Amateur-Programmierer namens Rick Auricchio („Our Hero") bei den neuen ROMs nicht mehr die Federführung innehatte. Hätte man indessen nicht nur eleganter programmiert, sondern zusätzlich die neuen Befehle benutzt, dann wäre der neue Apple IIe zweifellos noch schneller geworden, als er jetzt ist. Wenn Sie also nicht *selbst* mit den zusätzlichen 65C02-Befehlen programmieren und wenn Sie auf den geringeren Stromverbrauch des CMOS-Prozessors (früher 1-Megahertz-6502: 500 Milliwatt, jetzt 1-MHz-65C02: 20 mW; Angaben laut Rockwell) verzichten wollen, so können Sie auf den Einbau des neuen Prozessors verzichten. Ferner kann man auf den Einbau verzichten, wenn man eine Accelerator-IIe- oder Speedemon-Karte besitzt, die beide einen 65C02C-Prozessor aufweisen.

4. Video-ROM

Das Video-ROM umfaßt zwei Zeichensätze in jeweils deutsch und amerikanisch, die mit den Zeichensätzen des Apple IIc übereinstimmen. Der neue *Erstzeichensatz* (Primary or Main Character Set) entspricht dem alten IIe-Erstzeichensatz, während sich der neue Zweitzeichensatz (Second or Alternate Character Set) vom alten IIe-Zweitzeichensatz wie folgt unterscheidet (Versalien = Großbuchstaben; Gemeine = Kleinbuchstaben; Mauszeichen = diverse Fenstertechnik-Ikonen wie Apfelsymbole, Kästchen, Pfeile u.a.):

$00-$1F: Versalien invers
$20-$3F: Sonderzeichen invers
$40-$5F: Mauszeichen (neu!)
$60-$7F: Gemeine invers

$80-$9F: Versalien normal
$A0-$BF: Sonderzeichen normal
$C0-$DF: Versalien normal
$E0-$FF: Gemeine normal

Im Bereich $40-$5F waren früher im alten Zweitzeichensatz erneut inverse Versalien (wie im Bereich $00-$1F). Leider gibt es eine ganze Reihe kommerzieller Programme, die nichts von den Mauszeichen wissen und deshalb anstelle inverser Versalien die Mauszeichen anzeigen. Startet man beispielsweise mit der Microsoft-Premium-Card das Turbo-Pascal, so sieht man anstelle des Cursors und der inversen Meldungen nur noch „wirres Zeug".

Ähnlich ergeht es einem beim Applewriter IIe usw. Der Grund liegt darin, daß die Cursor- sowie allgemein die Normalinvers-Umkehrung in Assembler mit

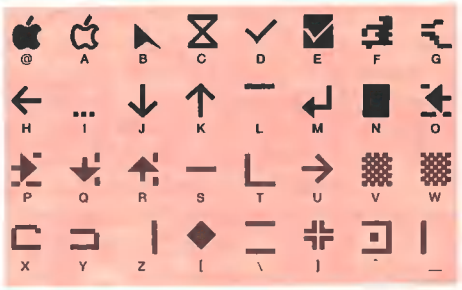
EOR #%10000000

bewerkstelligt wurde. Dies funktioniert bei den Mauszeichen nicht mehr.

Umgekehrt gibt es neuere Programme, die die Mauszeichen zwingend voraussetzen, insbesondere zur Simulation der Fenstertechnik. Hätte man die Mauszeichen in den Bereich $00-$1F gelegt, so wären die Probleme mit älteren Anwenderprogrammen wahrscheinlich geringer gewesen.

Die Bildqualität wird durch das neue Video-ROM übrigens nicht verbessert. Außerdem ist weiterhin nicht das Problem gelöst, daß nach dem Umschalten zwischen deutschem und amerikanischem Zeichensatz die Tastenbelegung der Sonderzeichen wechselt, was insbesondere Pascal-Programmierer stört. Hierzu ist eine zusätzliche Hardware-Änderung erforderlich, auf die manche Apple-Händler bereits eingerichtet sind.

Ein Beispielprogramm (s. **Demo 1:** CHARSET) veranschaulicht die Belegung von Erst- und Zweitzeichensatz, zwischen denen man per Tastendruck hin- und herschalten kann. CHARSET kann sowohl im 40- als auch im 80-Z/Z-Modus gestartet werden.



Maus-Zeichensatz
(Entnommen aus „Apple IIc Technical Reference" Manual)

5. CD- und EF-ROMs

Die neuen CD- und EF-ROMs beziehen sich auf den ROM-Bereich $C100-$CFFF und $D000-$FFFF. Da es ohne den hier nicht mehr charakterisierten Uralt-Apple-II (mit Integer-BASIC statt Applesoft-BASIC) mittlerweile vier verschiedene Apple-II-Typen gibt, wird es selbst einem Apple-II-Kenner bezüglich der Feinheiten immer schwieriger, den Überblick zu behalten. Bei allen Apple-II-Typen muß man – adreßmäßig von unten nach oben gesehen – zwischen unterem RAM, I/O-Adressen, externem und internem Slot-Bereich sowie oberem Interpreter- und Monitor-ROM unterscheiden:

1. Der **RAM-Bereich $0000-$BFFF** ist bei *allen* Typen derselbe, wenn man von den Bauelementen absieht (16K-Bit-Bausteine beim Apple II+, 64K-Bit-Bausteine beim Apple IIe und IIc).

Für Anfänger: RAM bedeutet Random Access Memory = Speicher mit direktem Zugriff = Lese/Schreibspeicher = Speicher, der durch Beschreiben verändert werden kann, dessen Inhalt jedoch beim Ausschalten des Apple verlorengeht. 64K-Bit = 8K, weil 1 Byte = 8 Bits, 1K = 1024 Bytes = 8192 Bits, 8K = 8192 Bytes = 65536 Bits = 64K-Bits)

2. Der **I/O-Bereich $C000-$C0FF** enthält die Input-Output-Adressen = Eingabe/Ausgabe-Adressen = Softswitches für Tastatur, DOS-Controller, Videosteuerung, Speicherverwaltung usw.

Für Anfänger: Der programmäßige Zugriff = „Soft Access" auf die Schalter = „Switches" bewirkt eine hardwaremäßige Veränderung: Bits werden von Diskette gelesen, Tasten abgefragt, Grafik eingeschaltet usw.

3. Der **externe Slot-Bereich $C100-$C7FF** – kurz **Slots** genannt – enthält die Einsteckkarten (Drucker-Interface, DOS-Controller, 80-Zeichenkarte, RAM-Karte, Modem usw.), die meist Slot-ROMs und seltener Slot-RAMs aufweisen. Manche Interface-Karten verfügen über ein zusätzliches ROM oder RAM im Bereich $C800-$CFFF. Der externe Slot-Bereich fehlt beim Apple IIc.

Für Anfänger: Damit ein Drucker, dessen Schnittstelle in Slot 1 steckt, gesteuert werden kann, braucht man nicht nur die Softswitches zur Übergabe der Zeichen, sondern auch ein Treiberprogramm, das die Zeichen konvertiert und die Softswitches betätigt. Dazu liegt im Falle des Slots 1 der Driver = Treiber im ROM-Bereich $C100-$C1FF. Grafik-Druckertreiber befinden sich bei entsprechenden Karten meist in dem zusätzlichen Bereich $C800-$CFFF (= **Zusatz-Slot-ROM/RAM**). Jede Slot-Karte kann diesen 2K-Zusatz-Bereich beanspruchen, so daß bei 7 Slots theoretisch insgesamt 7 mal 2K = 14K Zusatz-RAMs oder -ROMs vorhanden sein können. Es kann jedoch zu einem bestimmten Zeitpunkt immer nur ein einziger Zusatz-Bereich aktiv sein, nämlich derjenige des jeweils aktiven Slots. In dem Moment, da beispielsweise ein Grafik-Bitmuster an den Drucker gesendet wird, ist das Slot-1-Zusatz-ROM aktiv, und alle anderen Zusatz-Bereiche sind „tot". Wenn umgekehrt der interne Slot-Bereich aktiv ist, sind alle externen Slot-ROMs einschließlich möglicher Zusatz-ROMs „tot".

4. Der **interne Slot-Bereich $C100-$CFFF** – kurz **CX-ROM** genannt – existiert nicht beim Apple II+ (und auch nicht beim Uralt-Apple-II).

Beim *Apple IIc* ersetzt das interne CX-ROM die entsprechenden externen Slot-ROMs und Zusatz-Slot-ROMs des Apple IIe und Enhanced IIe. Das heißt für den IIc, daß beispielsweise der V.24-Treiber bei $C100, die 80-Zeichenkartenroutinen bei $C300, die Maussoftware bei $C400 und die DOS-Controller-Routinen bei $C600 *beginnen*. Im übrigen gibt es jedoch keine exakte Zuordnung der CX-Routinen des Apple IIc zu den externen Slot-Routinen des Apple IIe.

Beim *Apple IIe* (alt und neu) enthält das CX-ROM *zusätzliche* Routinen zur Erweiterung des Monitor-ROMs, insbesondere erweiterte Tastatureingabe- und Bildschirmausgaberoutinen (s.u. Ctrl-Zeichentabellen usw.). Dagegen findet man im IIe-CX-ROM keine V.24-, Maus- und DOS-Controller-Routinen; diese befinden sich vielmehr in den betreffenden Einsteckkarten, sofern der Apple IIe entsprechend ausgerüstet ist. Mit den Slot- und Zusatz-Slot-ROMs hat der Apple IIe bei maximaler Belegung der n = 7 Slots theoretisch ca. 16K mehr ROM als der Apple IIc, nämlich (7 * $Cn00-$CnFF) + 7 * C800-$CFFF. Deshalb verwundert es nicht, daß beispielsweise der Druckertreiber beim IIc keine Grafik-Druckroutinen enthält.

5. Der **obere ROM-Bereich** gliedert sich in das Applesoft-**Interpreter-ROM** ($D000-$F7FF) und das **Monitor-ROM** ($F800-$FFFF). Der Monitor (Monitor Program = „Überwachungsprogramm") enthält die elementaren Steuerungsroutinen, die allerdings beim IIe (alt und neu) sowie beim IIc durch das CX-ROM ergänzt werden. Die grundlegende Bedeutung der Monitor-Routinen kann man daran ersehen, daß der Monitor niemals in den Interpreter, aber umgekehrt der Interpreter häufig in den Monitor springt. Auch Betriebssysteme wie DOS 3.3 oder ProDOS benutzen ständig Monitor-Routinen.

Für Anfänger: ROM bedeutet Read-Only-Memory oder Nur-Lese-Speicher, d.h. man kann aus dem ROM lesen, aber nichts in das ROM schreiben. Nach dem Ausschalten des Apple bleibt der ROM-Inhalt erhalten.

Wenn man die verschiedenen Apple-II-Typen unterscheiden will, so muß man die folgenden vier Bereiche berücksichtigen, wobei sich die Ziffern 1-4 auf die **Abb. 2:** Apple-II-Typen beziehen:
1 Monitor-ROM
2 Interpreter-ROM
3 Externe Slots (ROM/RAM)
4 Internes CX-ROM

Im einzelnen gilt, wobei sich die Buchstaben A-D auf die Abb. 1 beziehen:
A Der Apple II+ verfügt über Monitor-ROM, Interpreter-ROM und externe Slots, aber über kein internes CX-ROM.
B Der Apple IIe (alt) verfügt über Monitor-ROM, Interpreter-ROM, externe Slots und internes CX-ROM.
C Der neue Apple IIe („enhanced") verfügt über Monitor-ROM, Interpreter-ROM, externe Slots und internes CX-ROM.
D Der Apple IIc verfügt über Monitor-ROM, Interpreter-ROM und internes CX-ROM, aber über keine externen Slots.

Die vier Komponenten sind bei den vier Apple-II-Typen unterschiedlich. Im einzelnen gilt:
a Die Monitor-ROMs unterscheiden sich bei *allen* vier Apple-II-Typen, d.h. kein Monitor-ROM des Modells X ist mit dem Monitor-ROM des Modells Y identisch, also A1 <> B1 <> C1 <> D1
b Die Interpreter-ROMs sind beim Apple II+ und alten Apple IIe identisch, dagegen beim neuen Apple IIe und Apple IIc untereinander und vom II+/IIe verschieden.

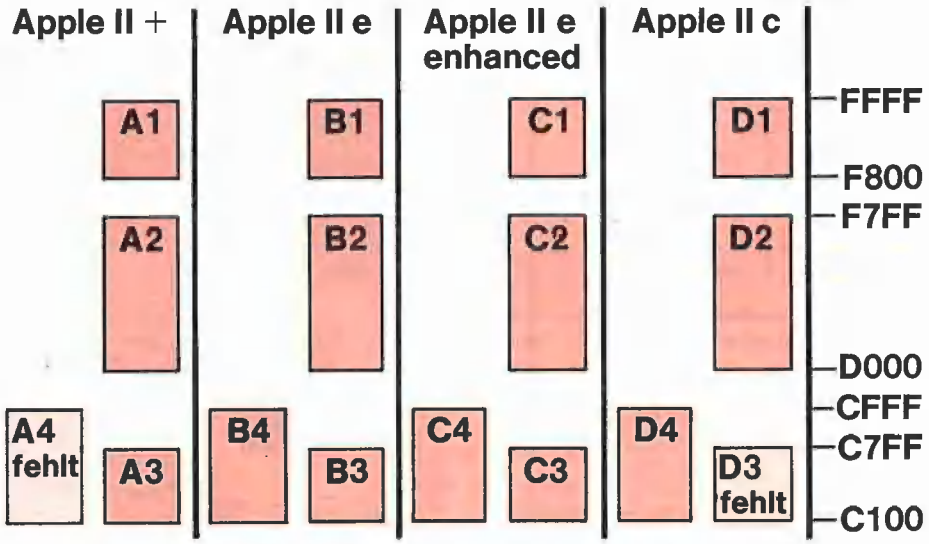


Abb. 2: Apple-II-Typen

1 = Monitor-ROM
2 = Interpreter-ROM
3 = Slots (ROM/RAM)
4 = Internes CX-ROM
A = II+
B = IIe
C = IIe enhanced
D = IIc

A1 <> B1 <> C1 <> D1
(A2 = B2)<> C2 <> D2
(A3 = B3 = C3)<> [D3]
[A4] <> B4 <> C4 <> D4

Das heißt also, daß der Interpreter des Apple IIc nicht mit dem Interpreter der neuen IIe-ROMs identisch ist. Formelmäßig ausgedrückt
(A2 = B2) <> C2 <> D2
c Beim Apple IIc fehlen die Slots, während sie bei den anderen Apple-II-Typen theoretisch funktionsgleich sein können, wenn man gleiche Interface-Karten benutzt und von dem zusätzlichen Slot 3 beim Apple IIe (neu und alt) abstrahiert. Formelmäßig ausgedrückt:
(A3 = B3 = C3) <> [D3]
d Beim Apple II+ fehlt das CX-ROM, während es bei den drei anderen Apple-II-Typen vorhanden ist. Allerdings gleicht kein CX-ROM dem anderen. Das heißt also, daß das CX-ROM des Apple IIc weder mit dem alten noch mit dem neuen CX-ROM des Apple IIe identisch ist. Formelmäßig ausgedrückt:
[A4] <> B4 <> C4 <> D4

6. Markante CD/EF-Neuerungen

Sie werden sich jetzt fragen, worin die Unterschiede im einzelnen bestehen. Dies kann nicht in einem einzelnen Aufsatz, sondern nur in einer Aufsatzserie geschildert werden. Im nächsten Peeker bringe ich eine bereits ausgearbeitete Übersicht über die bei *allen* vier Apple-II-Typen noch gleichermaßen verwendbaren Monitor-Routinen, die für Programmierer von größter Relevanz sind, denn es wird zunehmend schwieriger, Programme zu schreiben, die auf allen vier Typen uneingeschränkt lauffähig sind. Die den Applesoft-Interpreter betreffenden Abweichungen, die glücklicherweise nur geringfügig sind, werden von Harald Grumser im Teil 3 geschildert werden. Außerdem werden wir das Thema Interrupt-Bearbeitung in einem gesonderten Aufsatz zur Sprache bringen.

Für diejenigen jedoch, die jetzt schon einiges erfahren wollen, greife ich nachfolgend einige markante Besonderheiten der neuen (gegenüber den alten) Apple-IIe-CD/EF-ROMs heraus:

1. Der **Applesoft-Interpreter** läßt jetzt Kleinschreibung bei Befehlswörtern zu. Mit GET können jetzt alle Ctrl-Zeichen einschließlich ESC und Rechtspfeil eingegeben werden. Dies gilt übrigens auch für den Monitor. Die Double-Hires-Grafik wird nicht unterstützt. PR#0 schaltet jetzt die 80-Zeichenkarte ab (wie beim IIc). Der LIST-Befehl ist leicht modifiziert, was daran erkennbar ist, daß *vor* der Zeilennummer eine zusätzliche Leertaste ausgegeben wird (wie beim IIc). Die Kassettenroutinen sind erhalten geblieben (im Gegensatz zum IIc). Die HTAB- und Komma-TAB-Routinen wurden wegen der 80-Zeichenkarte umgeschrieben (s.u. Teil 2). Dies wirkt sich jedoch auf DOS 3.3 und ProDOS bei der Abspeicherung von Strings mit Kommafeldbegrenzern aus; s. Zeile 35 des nachfolgenden Listings. Auf der Diskette befinden sich dann keine Einzelstrings, sondern es werden in dem Listing beispielsweise 3 Strings zu einem Gesamtstring in der Form
TEST–TEST–TEST
vereint, wobei „–" für eine bestimmte Anzahl „harter" Leertasten = Hard Spaces steht. Das Programm würde in der vorliegenden Form nicht auf dem alten Apple IIe laufen. Probieren Sie's aus!

DOS-ProDOS-Kommatest

```
10 D$ = CHR$ (4)
15 PRINT D$"OPEN TEST"
20 PRINT D$"WRITE TEST"
25 X$ = "TEST"
30 FOR X = 1 TO 100
35 PRINT X$,X$,X$: REM Hier!
40 NEXT
45 PRINT D$"CLOSE"
50 PRINT D$"OPEN TEST"
55 FOR X = 1 TO 100: REM 100!
60 PRINT D$"READ TEST"
65 INPUT X$
70 PRINT D$: PRINT X$
75 NEXT
80 PRINT D$"CLOSE"
```

2. Der **Monitor** sowie der interne CX-Bereich wurden völlig umgeschrieben, wobei jedoch erfreulicherweise fast alle früheren (Spezial)einsprungadressen im Bereich $F800-$FFFF erhalten blieben. Dies gilt nicht für das CX-ROM, bei dem lediglich $C300 für das Einschalten der 80-Zeichenkarte sowie die 64K-Karte-MOVE-Routine ($C311) und die 64K-Karte-XFER-Routine ($C314) weiterhin benutzt werden können. Alle anderen CX-Adressen sind tabu, da sie bei allen Apple-II-Typen sehr stark abweichen. Neue Softswitches kommen natürlich durch Einstekken neuer ROMs nicht hinzu, also auch keine Maus-Softswitches.
Der 6502-Disassembler ist unverändert geblieben, obwohl Teile in den CX-Bereich ausgelagert wurden. Der früher im Uralt-Integer-BASIC-Apple-II enthaltene Mini-Assembler ist leicht modifiziert zum Leben erweckt worden und beginnt bei $C4C8 mit Fragmenten im F8-ROM-Bereich (s. **Demo 2:** Mini-Assembler). Eine genauere Erläuterung sowie weitere Übungsbeispiele werden zu einem späteren Zeitpunkt veröffentlicht.
Die alten Monitorbefehle (L, Ctrl-E usw.) wurden durch einen Suchbefehl S (beginnt im Speicher ab $FED7) für wahlweise 1 Byte (LL) oder 2 Bytes = Adressen (HHLL-Format) ergänzt. Beispiele (vorher CALL -151 ausführen):

Für Ihre Unterlagen

Abonnement bestellt

am: _____

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

peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

peeker
MAGAZIN FÜR APPLE-COMPUTER

Abo-Karte

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

Ich wünsche jährliche Berechnung durch:

☐ Verlagsrechnung

☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank / PschA _____

Bankleitzahl _____ Kto.-Nr. _____

Datum _____ Unterschrift _____

peeker
MAGAZIN FÜR APPLE-COMPUTER

Buch-Shop

Bitte senden Sie mir gegen Rechnung folgende Bücher:

Menge	Autor, Titel	à DM	gesamt DM

Datum _____ Unterschrift _____

peeker
MAGAZIN FÜR APPLE-COMPUTER

Software-Karte

Bitte senden Sie mir
gegen Rechnung folgende Apple-Programme:

☐ Peeker-Sammeldiskette, einzeln
Disk# _____, Disk# _____
Disk# _____, Disk# _____
Preis je Disk DM 28,– (einzeln)

☐ Peeker Sammeldiskette,
im Fortsetzungsbezug
ab Disk # _____
(Mindestbezug 6 Disketten)
Preis je Disk DM 20,–

☐ Apple DOS 3.3, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 1, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 2, 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,–
☐ Superquick, Programm, DM 48,–

Datum _____ Unterschrift _____

Abo-Karte

Name

Firma

Abteilung

Straße

PLZ/Ort

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

Datum

Unterschrift

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

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

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

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

„Input 2.0" liegt wahlweise in der Bank 1 oder Bank 2 der Language Card und wird durch einen kurzen Driver in den unteren 48K aufgerufen.
Für jedes Feld der Bildschirmmaske lassen sich u. a. definieren: Feldlänge (bis zu 255 Zeichen) – Vtab – Htab – Datentyp (insgesamt 8 Typen) – Scrollflag (starre oder dynamische Maske) – Ctrlflag – Füllflag – Löschflag – Bildschirmflag (40- oder 80-Z-Darstellung). Innerhalb eines Eingabefeldes besteht jeder denkbare Redigierkomfort (Insert, Delete, Rubout, Restore usw.).
Gerätevoraussetzung: Apple IIe oder IIc; ferner Apple II+ im 40-Zeichenmodus

MMU 2.0

Memory Managements Utilities

für die Apple IIe 64K-Karte DOS 3.3 (und ProDOS)

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7787-1023-1

Insgesamt enthält die neue „MMU 2.0"-Diskette über 25 Programme, die neue Einsatzmöglichkeiten für die Extended 80 Column Card (erweiterte 80-Z-Karte = 64K-Karte für den Apple IIe) erschließen. Ein Teil der Programme laufen auch auf dem Apple II Plus, doch ist „MMU 2.0" primär für 64K-Karte-Besitzer gedacht.
Gerätevoraussetzung: Apple IIe mit 64K-Karte oder IIc

Softbreaker 1.0

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

von U. Stiehl

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

Softbreaker ist ein Assemblerprogramm, mit dessen Hilfe Programme, die sich von der 64K-Karte (= Extended 80 Column Card für den Apple IIe) starten lassen, unterbrochen, gespeichert, geladen und exakt an der Stelle der Unterbrechung fortgeführt werden können. Dadurch ist es auch möglich, Sicherungskopien von sogenannten kopiergeschützten Programmen herzustellen.
Mit Softbreaker unterbrochene Programme werden komplett, d. h. die ganzen 64K einschließlich Language Card, in nur ca. 11 Sekunden auf einer formatierten Diskette gesichert.
Gerätevoraussetzung: Apple IIe mit 64K-Karte

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

PEEKER Börse

Verkauf Software

Astrologie: Berechn. u. Graphik. Info n. voreins. 1 DM in Briefm., C. Landscheidt, Im Dorfe 14, D-2804 Lilienthal.

– – – STOCKMASTER II – – – Das Apple-Programm für echte Börsengewinne. Diskette 485,– DM. Beschreibung P10 anfordern bei Töngi, COMPUTER-PRAXIS, Aspeltstr. 4, D-6500 Mainz 1

PIRATE DEFENCE 2.0 Kopierschutz. Die Antwort auf die neuen Nibble-Kopierer. Ab jetzt neben DOS 3.3 auch für ProDOS und Diversi-DOS. Info (50 Pf.) bei: Chr. Bregler, Tulpenstr. 2, 7519 Eppingen. Händleranfragen erwünscht.

Apple, Kompatible: Lehrerprogr., Public Domain Software u. a. Billig! Gratisinfo: Fa. Waltraud Muhle, Waldwinkel 3, 2105 Seevetal 3

APPLE IIe: Wegen Systemwechsel: (Originaldisketten u. Handbücher) Apple II Business Graphics 250,–, Pascal 1.2 (nutzt 128KB) 350,–, Fys-Forth, ProDos 1.01 je 70,–, Tel. 02 28 / 22 24 79

DFÜ-Programm APPLE ACCESS 2e/2c, Original verpackt DM 170,–, Weil-Seemann 02 11 / 34 74 11

Computer-Literatur: Liste bei: **Lindemanns Buchhandl.** kostenlos. Nadlerstr. 10, 7000 Stuttgart-1

80 Track Disk Utilities f. Erphi-Controller: Copy, Dos Copy. Verify Deutsch! Tel. 0 70 23 / 65 85. 14h-18h. 29 DM + Disk. Erphi-Controller AFDC 2

Verkauf Hardware

Verkaufe 100% APPLE komp. Comp. 64K, abgesetzte Tastatur, Z80, 80-Zeichen, Uhr, 2 Teac-Laufw. mit Contr., Metall-Gehäuse, Drucker. Preis auf Anfrage. Tel. 0 65 71 / 81 79

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

APPLE-INTERFACES: original PAL-Karte, PAL-Karte, Accelerator II (3,5 MHZ), Musik-Karte gstimmig/Stereo. Preise VHS. Tel. 0 62 21 / 7 48 32 o. 07 21 / 37 39 14

Schweiz: Neuwertiger Apple Dot-Matrix-Drucker inkl. Original-Parallel-Interface und viel Software: total nur 890,–!! Tel. 01 / 715 12 10

Neuw. Macintosh 512K + Software DM 7000, 1 J. Gt., Tel. 07 61 / 28 11 45

APPLE II (komp) 64K, 2 Floppy, Z80, 80Z, Monitor, Joyst., Epromer Karte Epson RX 80 F/T, Grafikinterface div. Software; kompl. od. einzeln Preis VB, Tel. 0 26 78 / 10 01

APPLE IIe, DISK + Contr., Monitor, Joyst. (Org.) + div. Softw. + 4 Bücher 3400 DM, W. Gross, Heinrichstr. 111, 61 Darmstadt, Tel. 0 61 51 / 42 22 68

IBM-Comp.: MICROMINT TIEFSTPREIS-GARANT, Tel. 0 21 04 - 3 94 71 (24 Std.)

Speedemon-Karte fast neu für DM 700. Chiffre P 1004

Zusatzkarten f. APPLE (II, II+, IIe) + Compatible (auch IBM PC), z. B. 192K RAM Disk, PC/APPLE Transfer Card, usw. supergünstig! Info von Klaus Oldigs, Landsbergstr. 435, 8000 München 60.

APPLE 256K-RAM-Karte 298,– DM/Geh. + Cherrytast. 120,– DM, Tel. 0 29 31 / 8 52 04

Macintosh 512 mit 2 Laufw., Maus, Imagewriter, M. Paint, M. Write, Filevision, Address, Draw, MS. Basic, u. a. Neupr. 18.482 DM, wenig gebraucht, absolut neuw., cpl. für 9000 VB zu verkaufen. Tel. 0 78 26 / 351

Apple II + (11 Mon.) + 80-Zeichen + Language Card + Z80 + Controller + Laufwerk + DOS 3.3 + Assembler zus. DM 1280,–, Tel. 0 64 28 / 18 28

MODEM für RS-232 Schnittstellen CCITT-Norm 300 Baud Vollduplex sofort anschließbar für 99,– DM, 0 61 27 / 47 80 oder 24h 0 61 21 / 7 31 33

MACINTOSH + IMAGEWRITER Garantie VHB 6000 + 1500 DM, Tel. 0 61 51 / 2 71 47

Verkaufe APPLE-Nachbau mit Z 80 KA, Super-S-Card, 80-Z-KA PARALLEL-INTERFACE. Tel. 0 21 92 / 42 39

APPLE MACINTOSH 512KB incl. Softw. wie neu, originalverpackt, auf Wunsch mit Drucker mit Garantie DM 7200,– VHB. Tel. 07 61 / 6 68 43

Neue RomPlus + Karte und Spracheingabe (Mikrofon + Zubehör) für Apple II. Tel. 0 64 61 / 8 94 16

Verkaufe Apple IIe mit Duodisk, 128 KB, 80-Zeichenkarte, zus. Disk-Controller, Monitor, DOT-Matrix-Drucker und div. Software – nur komplett – VB 5100,–. Tel. 02 08 / 60 51 37

APPLE IIc, dtsch., eingeb. Diskiwk. + Orig. Monitor + Ständer + Maus + Softw., neuwertig, VB 3000,– DM, Tel. 0 60 47 / 18 22

Apple IIe (!)-komp. DM 1250,00 bitte Liste anfordern, ComSoft, Waldstr. 96, 6078 Neu-Isenburg, Tel. 0 61 02 / 1 73 02

Ankauf Software

Suche Pascal- und Assemblerprogramme zur Arithmetik endlicher Körper u. zu zahlentheoretischen Funktionen. Tel. 0 21 63 / 4 72 02.

Kontakte

APPLE-Spezialist löst Ihre großen und kleinen Software-Probleme kostengünstig. B. Rüter, Rahdener Str. 65, 4955 Hille

Hilfe! Wer kann reine Zufallszahlen auf dem APPLE IIe produzieren oder entsprechende Adressen nennen? Tel. 0 62 24 / 29 48 (Rückruf)

Suche Händler o. ä. zum Vertrieb meiner selbsterstellten Profi-Software für APPLE II. E. Heinz, Waldgürtel 7, 5060 Berg. Gld. 1

Verschiedenes

APPLE REPARATUREN
(auch compatible M-boards, z. B. **Atlas, Arca, CES, Datastar, Dipa, Lasar, Mewa, PC-48 + 64, Plato, Radix, o. ae.) sowie Zusatzkarten und Disk-Drives** führt unser Spezialistenteam mit mehr als 5-jähriger Kunden- und Reparatur-Dienst-Erfahrung, garantiert zuverlässig und besonders kostengünstig aus.
Bitte genaue Fehlerangabe sowie Tel. Nr. für evtl. Rückfragen nicht vergessen.
Auf Wunsch Kostenvoranschlag.
aaa-electronic gmbh
Habsburgerstr. 134, 7800 Freiburg, Tel. 0761/276864, Tx. 772642aaad

Neue EPROM's zum Superpreis:

Menge	1-4	5-9	10-19	20-49
27256	29.00	26.50	24.00	22.00
27128	11.45	10.45	9.50	9.00
2764	7.55	6.90	6.25	5.95
2732	16.00	15.00	14.50	14.00
2732	8.00			(PROM)
2532	7.50	7.00	6.80	6.50
2716	7.50	6.00	6.50	6.00

Preiskorrekturen n. unten möglich. Auslandslieferung geg. Vorauskasse.
Computertechnik Ingo Klepsch, 5828 Ennepetal 1, Tel. 0 23 33 / 8 02 02

SHARP PC-1350: Taschenterminal für APPLE IIc! Tel. 0 42 21 / 6 28 63

Epson Interfaceumbau f. AWorks DM 20. Mahr, Waldakker 71, 73 Esslingen

Einkaufsführer

a) Sie suchen alle Bytes $CC im Bereich $1000-$5000. Dann geben Sie ein:
*CC<1000.5000S
b) Sie suchen alle JSRs nach $FDED im Bereich $D000-$FFFF. Dann geben Sie ein:
*FDED<D000.FFFFS
Bei mehr als zwei Hex-Bytes dreht die Suchroutine durch. Für ASCII-Folgen ist der Suchbefehl nicht gedacht, zumal ein maximal 2stelliger String *rückwärts* eingegeben werden müßte. Das „Durchdrehen" ist daran ersichtlich, daß Zero-Page-Adressen angezeigt werden.
Außerdem kann man jetzt im Speicheränderungsmodus statt Hex-Bytes auch ASCII-Zeichen eingeben, die allerdings immer mit Bit 7 on interpretiert werden:
c) Sie wollen $C1 $C2 $C3 = „ABC" ab $1000 eingeben. Dafür gibt es jetzt zwei Möglichkeiten:
*1000: C1 C2 C3
oder
*1000:'A 'B 'C
Nach dem letzten ASCII-Zeichen darf keine Leertaste stehen, sonst dreht die Routine durch.

3. Für **Pascal** wurden einige Verbesserungen vorgenommen. Beispielsweise soll jetzt von der Profile gebootet werden können (von mir mangels Harddisk noch nicht ausprobiert).

4. Der **Interrupt-Handler** wurde jetzt ähnlich wie beim Apple IIc implementiert, wobei der IRQ-Vektor allerdings beim neuen IIe auf $C3FA zeigt (beim IIc auf $C803, beim alten IIe auf $FA40). Dieser Handler wird Gegenstand eines gesonderten Peeker-Artikels sein.

Teil 2: Standardein- und -ausgabe

Tiefgreifende Änderungen haben sich beim „Enhanced Apple IIe" bei der Standard(tastatur)eingabe und Standard(bildschirm)ausgabe ergeben. Es hat mich viel Zeit gekostet, die Feinheiten herauszuarbeiten, denn natürlich habe ich zu Hause nicht 4 Apples nebeneinander stehen. Im einzelnen muß man unterscheiden:

1. Ausgabe von Ctrl-Zeichen

Die Ausgabe von Ctrl-Zeichen über COUT ($FDED) in Assembler und PRINT in Applesoft hat zahlreiche Modifikationen erfahren. Die **Tabelle 1:** Standardausgabe von Ctrl-Zeichen zeigt systematisch alle Befehle, während das **Demo 3:** DEMO.CTRL für Applesoft die Anwendung exemplifiziert. Man beachte, daß PRINT CHR$(21) jetzt zusätzlich den Bildschirm löscht. Dies war beim alten Apple IIe nicht der Fall.
Für Pascal 1.2 (nicht für 1.1) sind die Cursorbefehle Ctrl-F (bedeutet unterdrücke Cursor) und Ctrl-E (bedeutet zeige Cursor) neu hinzugekommen. Das **Demo 4:** DEMOCTRL für Pascal veranschaulicht die Anwendung dieser zwei Befehle. Je länger eine Ausgabezeile ist, desto schneller scrollt Pascal, wenn der Cursor unterdrückt wird. Man erweitere/kürze also den String S in dem Beispiel, um unterschiedliche Zeitdifferenzen zu messen. Im übrigen ist in meinen Augen eine der wesentlichsten Vorzüge der neuen IIe-ROMs die deutlich sichtbare Erhöhung der Scroll-Geschwindigkeit im 80-Z/Z-Modus unter allen Betriebssystemen.
Ferner wurde im Hinblick auf Modems die Unterdrückung der Standardausgabe von Ctrl-Zeichen durch Änderung von Bit 5 des sog. Mode-Bytes vorgesehen. Danach werden nur noch der Piepston, der Linkspfeil, der Zeilenvorschub und der Wagenrücklauf als Ctrl-Zeichen „durchgelassen", nicht mehr jedoch z.B. CHR$(21) usw. Das **Demo 5:** CTRL.DISABLE zeigt, daß man ein kleines Maschinenprogramm schreiben muß, wenn man aus einem Applesoftprogramm heraus den Ctrl-Zeichen-Unterdrückungsmodus einschalten/ausschalten will. Andernfalls ist nur die Eingabe über eine Tastatur-ESC-Sequenz möglich (s.u.).

2. Eingabe von Ctrl-Zeichen

Es gibt nach wie vor nur wenige Ctrl-Zeichen, die durch Tastatureingabe per INPUT und GET in Applesoft oder per GETLN ($FD6A) und RDKEY ($FD0C) in Assembler eine Wirkung auslösen. Die **Tabelle 2:** Standardeingabe von Ctrl-Zeichen zeigt die Eingabe-Ctrl-Zeichen, wobei das Wort „entfallen" hier nicht bedeutet, daß es die Ctrl-Befehle früher gab, sondern daß es sie noch nie gab. Eine Änderung betrifft Ctrl-L, das beim alten Apple IIe (aus Versehen?) den HOME-Befehl (mit „Syntax-Error") bewirkte.

3. Eingabe von ESC-Sequenzen

Die ESC-Sequenzen werden durch Tippen von ESC, gefolgt von einem anderen zulässigen Zeichen, über die Tastatur eingegeben (s. **Tabelle 3:** Standardeingabe von ESC-Sequenzen). Im Gegensatz zum alten Apple IIe kann man als Folgezeichen jetzt auch Kleinbuchstaben tippen. Entfallen sind ESC-R (bewirkte beim alten IIe die Umwandlung von getippten Kleinbuchstaben in Großbuchstaben) sowie ESC-T (= Aufhebung des Großbuchstabenmodus). Statt dessen sind ESC-Ctrl-E und ESC-Ctrl-D neu hinzugekommen (wie beim IIc). ESC-Ctrl-E (E = enable) ermöglicht die Ausgabe aller Ctrl-Zeichen, während ESC-Ctrl-D (D = disable) den Ctrl-Zeichen-Unterdrückungsmodus einschaltet (s.o.).

4. Bugs

Einige alte Bugs wie z.B. der berüchtigte ESC-Ctrl-L-Bug oder der HTAB-Bug sind ausgemerzt worden, während uns einige alte Bugs erhalten blieben bzw. neu „implementiert" wurden. Das **Demo 6:** HTAB1.BUG veranschaulicht einen neuen Bug, während das **Demo 7:** INVERSE.BUG einen altvertrauten Bug demonstriert.
Insgesamt muß man festhalten, daß die Verbesserungen überwiegen, so daß die Anschaffung der neuen ROMs nachdrücklich empfohlen werden kann.

A) Demos zu den neuen IIe-ROMs

1. CHARSET (Maus-Zeichensatz)
2. MINI-ASSEMBLER (Übungsbeispiel)
3. DEMO.CTRL (Ctrl-Zeichen in Applesoft)
4. DEMOCTRL (Cursor-Ctrl-Zeichen in Pascal)
5. CTRL.DISABLE (Unterdrücken der Ctrl-Ausgabe) mit Maschinenprogramm DISABLE.ENABLE
6. HTAB1.BUG (HTAB 1 nach PRINT ...;)
7. INVERSE.BUG (nach PR#3 und in Zeile 24)

1. CHARSET

BSAVE CHARSET, A$Ø3ØØ, L1Ø9
Starten mit BRUN CHARSET
Mit A- und M-Taste Zeichensätze wechseln.
Funktioniert auch uneingeschränkt auf dem Apple IIc.

```
                      1             ORG  $3ØØ
                      2    *
                      3    * CHARSET
                      4    * ========
                      5    * Demo zu dem neuen Zeichensatz
                      6    * des "Enhanced IIe"/us/17.Ø8.85
                      7    *
                      8    BASL     EQU  $ØØ28
                      9    ZEILE    EQU  $ØØFF
                      1Ø   HOME     EQU  $FC58
                      11   COUT     EQU  $FDED
                      12   BASCALC  EQU  $FBC1
                      13   TABV     EQU  $FB5B
                      14   ALTOFF   EQU  $CØØE
                      15   ALTON    EQU  $CØØF
                      16   *
                      17   * Zeichensatz in 8 Zeilen
                      18   * zu je 32 Zeichen anzeigen
                      19   * Die Mauszeichen liegen im
                      2Ø   * Bereich $4Ø-$5F im alternativen
                      21   * Zeichensatz anstelle der
                      22   * inversen Großbuchstaben beim
                      23   * alten IIe-ROM, die früher
                      24   * sowohl im $ØØ-$1F- als auch
                      25   * im $4Ø-$4F-Bereich doppelt
                      26   * vorkamen.
                      27   *
Ø3ØØ: 2Ø 58 FC        28            JSR  HOME
Ø3Ø3: A2 ØØ           29            LDX  #Ø
Ø3Ø5: 86 FF           3Ø            STX  ZEILE
Ø3Ø7: A5 FF           31   LOOP     LDA  ZEILE
Ø3Ø9: 2Ø C1 FB        32            JSR  BASCALC
Ø3ØC: AØ ØØ           33            LDY  #Ø
Ø3ØE: 8A              34   LOOP1    TXA
Ø3ØF: 91 28           35            STA  (BASL),Y
Ø311: E8              36            INX
Ø312: FØ ØB           37            BEQ  LOOP2
Ø314: C8              38            INY
Ø315: CØ 2Ø           39            CPY  #32
Ø317: DØ F5           4Ø            BNE  LOOP1
Ø319: E6 FF           41            INC  ZEILE
Ø31B: E6 FF           42            INC  ZEILE
Ø31D: DØ E8           43            BNE  LOOP
                      44   *
                      45   * Haupt- und alternativen
                      46   * Zeichensatz wählen
                      47   *
Ø31F: A9 14           48   LOOP2    LDA  #2Ø
Ø321: 2Ø 5B FB        49            JSR  TABV
Ø324: A2 ØØ           5Ø            LDX  #Ø
Ø326: BD 5Ø Ø3        51   LOOP3    LDA  STRING,X
Ø329: FØ Ø6           52            BEQ  LOOP4
Ø32B: 2Ø ED FD        53            JSR  COUT
Ø32E: E8              54            INX
Ø32F: DØ F5           55            BNE  LOOP3
Ø331: AD ØØ CØ        56   LOOP4    LDA  $CØØØ       ;Taste
Ø334: 1Ø FB           57            BPL  LOOP4
Ø336: 2C 1Ø CØ        58            BIT  $CØ1Ø       ;Strobe
Ø339: C9 C1           59            CMP  #"A"
Ø33B: DØ Ø5           6Ø            BNE  LOOP5
Ø33D: 8D ØF CØ        61            STA  ALTON
Ø34Ø: FØ EF           62            BEQ  LOOP4
Ø342: C9 CD           63   LOOP5    CMP  #"M"
Ø344: DØ Ø5           64            BNE  LOOP6
Ø346: 8D ØE CØ        65            STA  ALTOFF
Ø349: FØ E6           66            BEQ  LOOP4
Ø34B: C9 C5           67   LOOP6    CMP  #"E"
Ø34D: DØ E2           68            BNE  LOOP4
Ø34F: 6Ø              69            RTS
                      7Ø   *
Ø35Ø: C1 BD C1        71   STRING   ASC  "A=Altchar"
Ø353: EC F4 E3 E8 E1 F2
Ø359: 8D              72            HEX  8D
Ø35A: CD BD CD        73            ASC  "M=Mainchar"
Ø35D: E1 E9 EE E3 E8 E1 F2
Ø364: 8D              74            HEX  8D
Ø365: C5 BD C5        75            ASC  "E=Exit"
Ø368: F8 E9 F4
Ø36B: 8D ØØ           76            HEX  8DØØ

1Ø9 Bytes
```

2. MINI-ASSEMBLER

Man beachte die Leertaste nach dem Ausrufezeichen, das man selbst nur beim ersten Mal eingeben muß.
Funktioniert nicht auf Apple IIc und altem IIe.

```
CALL -151                  -> *
*!                         -> !
!3ØØ: LDX #Ø               -> Ø3ØØ- A2 ØØ     LDX #$ØØ
! LDA #'A                  -> Ø3Ø2- A5 C1     LDA #$C1
! JSR FDED                 -> Ø3Ø4- 2Ø ED FD  JSR $FDED
! INX                      -> Ø3Ø7- E8        INX
! BNE 3Ø4                  -> Ø3Ø8- DØ FA     BNE $Ø3Ø4
! RTS                      -> Ø3ØA- 6Ø        RTS
!r = Return ohne Leertaste -> *                .
*3ØØG                      -> AAAAAAAAA usw.
```

3. DEMO.CTRL für Applesoft

Funktioniert auch uneingeschränkt auf dem Apple IIc.
Beim alten Apple IIe funktionieren die Maus-Ctrl-Codes nicht. Ferner führt CHR$(21) kein zusätzliches HOME aus.

```
1ØØ PRINT CHR$ (4);"PR#3"
1Ø5 PRINT "DEMO.CTRL ";
11Ø GET X$
115 PRINT CHR$ (12);: REM Ctrl-L
12Ø PRINT "HOME ";
125 GET X$
13Ø PRINT "oder HOME ";
135 GET X$
14Ø HOME
145 PRINT "HOME"
15Ø PRINT
155 PRINT "BELL ";
16Ø GET X$
165 PRINT CHR$ (7): REM Ctrl-G
17Ø GET X$
175 PRINT
18Ø Y = 5ØØ
185 FOR X = 1 TO Y
19Ø PRINT "A";
195 NEXT
2ØØ PRINT "Linkspfeil ";
2Ø5 GET X$
21Ø FOR X = 1 TO Y
215 PRINT CHR$ (8);: REM Ctrl-H
22Ø NEXT
225 GET X$
23Ø PRINT "Rechtspfeil ";
235 GET X$
24Ø FOR X = 1 TO 1ØØ
245 PRINT CHR$ (28);: REM Ctrl-Ö
25Ø NEXT
255 GET X$
26Ø PRINT "Hochpfeil ";
265 GET X$
27Ø FOR X = 1 TO 5
275 PRINT CHR$ (31);: REM Ctrl-_
28Ø NEXT
285 GET X$
29Ø PRINT "Tiefpfeil ";
295 GET X$
3ØØ FOR X = 1 TO 5
3Ø5 PRINT CHR$ (1Ø);: REM Ctrl-J
31Ø NEXT
315 GET X$
32Ø PRINT "CLREOL ";
325 GET X$
33Ø PRINT CHR$ (29): REM Ctrl-Ü
335 GET X$
34Ø PRINT "CLREOS ";
345 GET X$
35Ø PRINT CHR$ (11): REM Ctrl-K
355 GET X$
36Ø PRINT "Return "
365 FOR X = 1 TO 5
37Ø PRINT CHR$ (13);
375 NEXT
38Ø GET X$
385 PRINT CHR$ (15);: REM Ctrl-O
```

```
390 PRINT "INVERSE";
395 PRINT CHR$ (14);: REM Ctrl-N
400 PRINT "NORMAL";
405 GET X$
410 PRINT : PRINT "Oder ";
415 INVERSE : PRINT "INVERSE";
420 NORMAL : PRINT "NORMAL";
425 GET X$
430 VTAB 24: PRINT
435 PRINT "Scrolldown ";
440 GET X$
445 VTAB 15: PRINT
450 FOR X = 1 TO 8
455 PRINT CHR$ (22);: REM Ctrl-V
460 NEXT
465 VTAB 24: PRINT
470 PRINT "Scrollup ";
475 GET X$
480 VTAB 15: PRINT
485 FOR X = 1 TO 8
490 PRINT CHR$ (23);: REM Ctrl-W
495 NEXT
500 VTAB 24: PRINT
505 GET X$
510 PRINT "40 Z/Z ";
515 GET X$
520 PRINT CHR$ (17): REM Ctrl-Q
525 GET X$
530 PRINT "80 Z/Z ";
535 GET X$
540 PRINT CHR$ (18): REM Ctrl-R
545 GET X$
550 PRINT "VTAB 1, HTAB 1 ";
555 GET X$
560 PRINT CHR$ (25);: REM Ctrl-Y
565 GET X$
570 PRINT "CLRLINE ";
575 GET X$
580 PRINT CHR$ (26);: REM Ctrl-Z
585 GET X$
590 PRINT : PRINT "Mauszeichen ein ";
595 GET X$
600 PRINT CHR$ (27); CHR$ (15);: REM Ctrl-Â,Ctrl-O
605 PRINT "ABCDEFGHI";
610 GET X$
615 PRINT CHR$ (24); CHR$ (14);: REM Ctrl-X,Ctrl-N
620 PRINT " Mauszeichen aus ";
625 GET X$
630 PRINT "ABCDEFGHI ";
635 GET X$
640 HOME
645 PRINT "80 Z/Z abstellen"
650 PRINT CHR$ (21): REM Ctrl-U
```

4. DEMOCTRL für Pascal

Unterdrückung des sichtbaren Cursors nach Ctrl-F.
Nur für Pascal 1.2 implementiert.
Funktioniert auch auf dem Apple IIc, jedoch NICHT auf dem alten Apple IIe.

```
PROGRAM DEMOCTRL;

CONST
  S = ' 123456789012345678901234567890';
  Y = 1200;
VAR
  X: INTEGER;
PROCEDURE SCHREIBEN;
  BEGIN
  FOR X:= 1000 TO Y DO
  WRITELN (X, '  ', S);
  END;
BEGIN
  WRITELN ('Writeln ohne Cursor');
  WRITELN (CHR(6)); {Ctrl-F}
  SCHREIBEN;
  WRITELN ('Writeln mit Cursor');
  WRITELN (CHR(5)); {Ctrl-E mit Cursor}
  SCHREIBEN;
END.
```

5. 1. CTRL.DISABLE

Funktioniert auch uneingeschränkt auf dem Apple IIc. NICHT beim alten Apple IIe ausprobieren, da sonst das veränderte Mode-Byte ein Bildschirm-"Chaos" anrichtet.

```
10 HOME : PRINT "CTRL.DISABLE ";: GET X$
15 PRINT : PRINT CHR$ (4)"PR#3"
20 PRINT PEEK (1275)
25 GOSUB 75: CALL 512:
   PRINT "Ctrl-Enabled -> ESC Ctrl-E"
30 PRINT CHR$ (21): PRINT "CHR$(21) wird ausgeführt"
35 PRINT PEEK (1275)
40 GET X$: PRINT
45 PRINT CHR$ (4)"PR#3"
50 PRINT PEEK (1275)
55 GOSUB 75: CALL 512 + 9:
   PRINT "Ctrl-Disabled -> ESC Ctrl-D"
60 PRINT CHR$ (21): PRINT "CHR$(21) wirkungslos"
65 PRINT PEEK (1275)
70 END: REM s.u. ENABLE.DISABLE
75 DATA 173,251,4,41,223,141,251,4,96,
        173,251,4,9, 32, 141,251,4,96
80 RESTORE : FOR X = 1 TO 18: READ Y:
   POKE 512 - 1 + X,Y: NEXT
85 RETURN
90 REM $04FB=1275=Mode
```

5. 2. ENABLE.DISABLE

Als DATA-Statements in CTRL.DISABLE enthalten

```
                    1             ORG  $2000
                    2     *
                    3     * Ctrl-Zeichen Enable/Disable
                    4     *
0200: AD FB 04      5     ENABLE  LDA  $4FB       ;ESC-Ctrl-E
0203: 29 DF         6             AND  #%11011111
0205: 8D FB 04      7             STA  $4FB
0208: 60            8             RTS
0209: AD FB 04      9     DISABLE LDA  $4FB       ;ESC-Ctrl-D
020C: 09 20         10            ORA  #%00100000
020E: 8D FB 04      11            STA  $4FB
0211: 60            12            RTS

18 Bytes
```

6. HTABL.BUG

Gilt nur für HTAB 1, nicht für HTAB 2 usw.
Der Bug tritt in derselben Form beim Apple IIc auf.
FAST dasselbe gilt für den alten Apple IIe.

```
10 HOME : PRINT "HTAB-1-Bug der neuen ROMs ";: GET X$
15 PRINT : PRINT CHR$ (4);"PR#3": PRINT CHR$ (21):
   PRINT "40 Z/Z F8-ROM": GOSUB 35: GET X$
20 PRINT : PRINT CHR$ (4);"PR# 3": PRINT CHR$ (17):
   PRINT "40 Z/Z CX-ROM": GOSUB 35: GET X$
25 PRINT : PRINT CHR$ (4);"PR#3": PRINT CHR$ (18):
   PRINT "80 Z/Z CX-ROM": PRINT "Hier Bug!":
   GOSUB 35: GET X$
30 PRINT "Traurig, traurig!": END
35 FOR V = 1 TO 23
40 VTAB V
45 HTAB 24 - V
50 PRINT "/";
55 VTAB 11
60 HTAB 1: REM Hier!!!
65 PRINT "-";
70 NEXT
75 RETURN
```

7. INVERSE.BUG

Tritt nur auf, wenn nach INVERSE gescrollt wird.
Gilt ebenso für den Apple IIc und den alten Apple IIe.

```
10 HOME : PRINT "INVERSE-Scroll-Bug der neuen ROMs ";:
   GET X$
15 PRINT : PRINT CHR$ (4);"PR#3": PRINT CHR$ (21):
   GOSUB 35: PRINT "40 Z/Z F8-ROM": GET X$
20 PRINT : PRINT CHR$ (4);"PR# 3": PRINT CHR$ (17):
   GOSUB 35: PRINT "40 Z/Z CX-ROM":
   PRINT "Hier Bug!": GET X$
25 PRINT : PRINT CHR$ (4);"PR#3": PRINT CHR$ (18):
   GOSUB 35: PRINT "80 Z/Z CX-ROM":
   PRINT "Hier Bug!": GET X$
30 PRINT "Traurig, traurig!": END
35 FOR X = 1 TO 13
40 INVERSE
45 PRINT "INVERSE"
50 NORMAL
55 PRINT "NORMAL?"
60 NEXT
65 RETURN
```

Technische Beschreibung der Geräte im redaktionellen Teil dieser Zeitschrift.

Händleranfragen erwünscht unter Tel. 02191/50041

Electronic Shop Chr. Kaup
4400 Münster, Bremer Platz 42–46
☎ 0251 / 665887

MicroComp GmbH
6200 Wiesbaden, Klarentaler Str. 6
☎ 06121 / 45377

Gelsen Electronic
4650 Gelsenkirchen
Kurt-Schumacher-Straße 124
☎ 0209 / 83033

Menkens Elektronik
Ulrich Menkens
2870 Delmenhorst, Grünestr. 70
☎ 04221 / 14591

Thönnes Elektronik
8000 München 70
Fürstenrieder Straße 206
☎ 089 / 148285

ALPHATRON
marco hildebrandt
8520 Erlangen · Luitpoldstraße 22
☎ 09131 / 22600

electronic + computer
GRIGENTIN + FALK
7580 Bühl, Hauptstraße 17,
☎ 07223 / 21170

SW design
Stolte – Wilken
51 Aachen/Kor.
Korneliusmarkt 20
☎ 02408 / 4249

Dreieich-Rechner-Versand
6072 Dreieichenhain
Am Kellersbusch 4
☎ 06103 / 84647

MiCom-Computer Olaf Mertens
Industriehof Lüttringhausen
5630 Remscheid 11
Grünenplatzstraße 16–18
☎ 02191 / 590313

VID DATA SYSTEME
WEISING KG
5483 Bad Neuenahr · Postf. 933
☎ 02641 / 1478

Wollen Sie mit Ihrem MACINTOSH grafisch arbeiten?

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

Wir stellen aus:
SYSTEMS 85, München
28. 10. – 1. 11. 85
Halle 15 OG
Stand BB

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

Summagraphics® LIMITED
Niederlassung Deutschland

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

Eine andere Form des INVERSE-Bugs ist folgende:

```
10 PRINT CHR$(4); "PR#3"
20 INVERSE: PRINT "A";: NORMAL
```

Danach ist der GESAMTE Bildschirm invers. Das erste Zeichen nach PR#3 muß NORMAL ausgegeben werden. Gilt auch für den Apple IIc sowie den alten Apple IIe.

B) Tabellen zu den neuen IIe-ROM-Steuerzeichen

1. Standardausgabe von Ctrl-Zeichen

Standardausgabe in Applesoft über PRINT CHR$(X)
Standardausgabe in Assembler über JSR COUT ($FDED)
* = funktioniert nur nach PR#3 ($CX00-Routinen)
Tabelle gilt auch uneingeschränkt für den Apple IIc.
n-IIe! = nicht IIe, d.h. nicht beim alten Apple IIe.

Hex.	Dez.	ASCII	Kür.	CX	Wirkung
05	5	Ctrl-E	ENQ	*	Cursor sichtbar (Pascal 1.2); n-IIe!
06	6	Ctrl-F	ACK	*	Cursor unsichtbar (Pascal 1.2); n-IIe!
07	7	Ctrl-G	BEL		Piepston (ca. 0,1s)
08	8	Ctrl-H	BS		Cursor nach links; vgl. 1C
09	9	Ctrl-I	HT		entfällt
0A	10	Ctrl-J	LF		Cursor nach unten; vgl. 1F
0B	11	Ctrl-K	VT	*	CLREOS: Clear to end of screen
0C	12	Ctrl-L	FF	*	HOME = CLRSCR + VTAB 1, HTAB 1, vgl. 19
0D	13	Ctrl-M	CR		Return
0E	14	Ctrl-N	SO	*	NORMAL
0F	15	Ctrl-O	SI	*	INVERSE
10	16	Ctrl-P	DLE		entfällt
11	17	Ctrl-Q	DC1	*	40 Z/Z mit CX-Routinen
12	18	Ctrl-R	DC2	*	80 Z/Z mit CX-Routinen
13	19	Ctrl-S	DC3		entfällt
14	20	Ctrl-T	DC4		entfällt
15	21	Ctrl-U	NAK	*	40 Z/Z mit F8-Routinen und HOME; HOME n-IIe!
16	22	Ctrl-V	SYN	*	Scrolldown (letzte Zeile verschwindet)
17	23	Ctrl-W	ETB	*	Scrollup (erste Zeile verschwindet)
18	24	Ctrl-X	CAN	*	Mauszeichensatz aus; n-IIe!
19	25	Ctrl-Y	EM	*	Nur VTAB 1, HTAB 1; vgl. 0C
1A	26	Ctrl-Z	SUB	*	CLRL: Clear line; vgl. 1D
1B	27	Ctrl-Ä	ESC	*	Mauszeichensatz ein; n-IIe!
1C	28	Ctrl-Ö	FS	*	Cursor nach rechts, vgl. 08
1D	29	Ctrl-Ü	GS	*	CLREOL: Clear to end of line, vgl. 1A
1E	30	Ctrl-↑	RS	*	GOTOXY (nur Pascal, jede Version)
1F	31	Ctrl-_	US	*	Cursor nach oben, vgl. 0A

2. Standardeingabe von Ctrl-Zeichen

Tastatureingabe in Applesoft über INPUT und GET
Tastatureingabe in Assembler über GETLN ($FD6A) und RDKEY ($FD0C)
* = funktioniert nur nach PR#3 ($CX00-Routinen)
Tabelle gilt auch uneingeschränkt für den Apple IIc und altem IIe.
Hinweis: Tastatureingabe über GET und RDKEY akzeptiert bei den neuen IIe-ROMs sowie beim Apple IIc ALLE Ctrl-Zeichen. Beim alten Apple IIe werden hingegen ESC und Ctrl-U NICHT angenommen.

Taste	Alternativtaste	CX	Wirkung
Ctrl-A	bis Ctrl-F		entfallen
Ctrl-G			Piepston
Ctrl-H	(Linkspfeil)		Cursor nach links
Ctrl-I	(Tabulatortaste)		entfällt
Ctrl-J	(Tiefpfeil)		Cursor nach unten
Ctrl-K	(Hochpfeil)		entfällt
Ctrl-L		*	entfällt (Bug-HOME beim alten IIe)
Ctrl-M	(Returntaste)		Return
Ctrl-N	bis Ctrl-R		entfallen
Ctrl-S			Stoppt Standardausgabe
Ctrl-T			entfällt
Ctrl-U	(Rechtspfeil)		Cursor nach rechts
Ctrl-V	bis Ctrl-W		entfallen
Ctrl-X			Eingabewiederholung bei INPUT, GETLN
Ctrl-Y	bis Ctrl-Z		entfallen
Ctrl-Ä	(ESC-Taste)		s.u.
Ctrl-Ö	bis Ctrl-_		entfallen

Teil 3: Applesoft-Interpreter

von Harald Grumser

Eine Änderung des Applesoft-Interpreters dürfte von den meisten bestehenden Programmen, die einzelne Routinen aufrufen, mit noch größerer Sensibilität quittiert werden wie beim Monitor. Da hier keine definierten Einsprungstellen veröffentlicht wurden, würde eine umfangreiche Änderung das Ende vieler Applikationen „von der Stange" bedeuten. Da auch die meisten Compiler bei verantwortungslosen Patches ihre Arbeit versagt hätten, haben sich die Änderungen auf das Nötigste beschränkt (diese Aussage bezieht sich nur auf den Interpreter).
Tatsächlich wurden nur solche Stellen geändert, die ohnehin nicht zum externen Einsprung geeignet waren. Um für die neuen Fähigkeiten Platz zu schaffen, wurde jedoch leider auf den CX-Bereich ausgewichen. Da hiervon nur der STORE-, RECALL- und SHLOAD-Befehl betroffen sind, kann jedoch nicht auf einen erheblichen Kompatibilitätsverlust geschlossen werden.
Welche Stellen wurden nun verändert? Für Interpreter-Kenner folgt diesem Artikel ein disassembliertes Listing aller geänderten Stellen. Wichtiger sind jedoch die Änderungen, die sich für den Applesoft-Programmierer ergeben:

1. Komma-Tabulierung

Beim alten Interpreter kam der Cursor im 80-Z/Z-Modus nicht über die zweite Tabulatorstelle hinaus. Mit der Anweisung „PRINT 1,222,33,4" erhielt man z.B. folgende Ausgabe:

```
1 432
```

Der neue Interpreter verarbeitet diesen Befehl jetzt korrekt, allerdings nur, wenn die von der Firmware unterstützte 80-Zeichenkarte benutzt wird (dies gilt im übrigen auch für den HTAB-Befehl, s.u.).
Ein alter Fehler im 40-Z/Z-Modus blieb erhalten:

„PRINT 1,12345678,2" ergibt

```
1                 12345678
2
```

„PRINT 1,2,123456789" ergibt

```
1                 2           12345678
9
```

Um diesen Fehler zu beseitigen, hätte jedoch der komplette Algorithmus umgeschrieben werden müssen.

2. Eingabe von Kleinbuchstaben

Nunmehr ist es wie beim Apple IIc möglich, alle Applesoft-Befehle in Kleinbuchstaben oder auch gemischt einzugeben:

ccp datentechnik

640 KByte-Drives für den Apple //c!!

- 5¼- od. 3½-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 35/40 Track
- Anschluß an die externe Laufwerkbuchse
- Durch Einbauplatine (kein Löten) 640 KByte im Direktzugriff
- Einfache Anpassung für DOS 3.3, UCSD-Pascal und PRODOS durch menügeführten Patch
- Anpassung von CP/M in Verbindung mit einer Z 80-Zusatzplatine in Vorbereitung
- anschlußfertig im Gehäuse . . . . . **DM 1090,–**

Festplatten für Apple II (//e)

- 5¼ Zoll-Format (Slimline)
- Booten direkt von der Festplatte in DOS 3.3, UCSD-Pascal, PRODOS und CP/M 2.2 / 3.0
- Gemischtbetr. mit 35/40/80/160 Track-Drives
- Copy- und Install-Programme im Lieferumfang
- Umfangreiches Manual
- z. B. 10 MB incl. Netzteil u. Contr., anschlußfertig an Ihren Apple . . . . . **DM 3450,–**

640 KByte-Drives für Apple II (//e)

- 5¼- od. 3½-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 40 Track (Apple kompatible)
- Installationssoftware für DOS 3.3, UCSD-Pascal, CP/M 2.2, CP/M 2.23 (60K), PRODOS, AP22, ALS CP/M+
- Umfangreiches Handbuch
- Anschlußfertige Auslieferung incl. Contr. und 2 Drives
- Diskstation 55II (2 Teac FD55-F, 1.2 MB) . . . **DM 1540,–**
- Diskstation 35II (2 Teac FD35-F, 1.2 MB) . . . **DM 1598,–**

80 Zeichen + 64 K für Apple //e

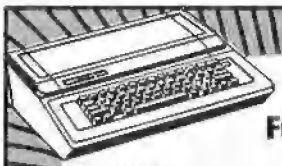
- und jetzt hinsetzen . . . . . **DM 138,–**

Alles für Ihren Apple

Info bei:

ccp-datentechnik

Herderstraße 12 – 2000 Hamburg 76
Telefon 040/225676



Für Apple II, IIe

Z-80-Karte	69,–	**80-Zeichen-Karte** mit Softswitch, neue Vers. m. gest. scharf. Bild	139,–
Disk-Interface	69,–		
Centronics-Interf. m. Kabel	69,–	**Speech-Karte**	55,–
16-K-RAM-Karte	99,–	**Clock-Karte**	129,–
RS-232-Karte	169,–	**Super-Serial-Karte**	199,–
Eprommer (4, 8, 16 K)	139,–	**Komp 2E** Apple 2E kompatible, Rechner 64 K im 2E-Design, ohne Firmware	797,–
128-K-RAM-Karte	279,–		
256-KB-RAM-Karte	448,–	**80Z + 64K-Karte** für 2E kompatible	99,–
Wild-Karte (knackt geschützte Programme)	69,–		

Händleranfragen erwünscht! **Apple-Info 1,- DM (Porto)**

Apple II + Kompatible

Komp 48 538,–
48 K, 6502 ohne Firmware

Komp 64 640,–
64 K, 6502, Z-80, 15er-Block ohne Firmware

Komp 64 S 940,–
wie Komp 64, jedoch mit abgesetzter Tastatur mit 188 Funktionen.

Motherboard 48 K 399,–
8 Slots, alle IC's gesockelt, ohne Firmware, fertig geprüft

Motherboard 64 K 399,–
wie oben, mit 6502 und Z 80, 64 K

SUPER PREISE

Klaus Jeschke
Hard-, Software
Viertstr. 3-13
6233 Kelkheim
☎ (0 61 98) 75 23

Alle Preise inklusive Mehrwertsteuer. 6 Monate Garantie Versand erfolgt per NN oder Vorkasse

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,–
Centronics-Schnittstelle für 2 Drucker gleichzeitig			129,–
Eprommer incl. Software			198,–

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

Floppy-Controller

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

Erphi-Controller **298,–**

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

Joy Stick De Luxe	59,–	**Netzteil 5A**	149,–
Gehäuse für 1 5¼" Slimline Laufwerk			39,–
Gehäuse für 2 5¼" Slimline Laufwerke mit Platz für ein Netzteil			159,–
Gehäuse für 2 3½" Slimline Laufwerke mit Platz für ein Netzteil			79,–
IBM®-Gehäuse			229,–
Floppy-Kabel 34pol. für 2 Laufwerke mit Shugart-Bus			40,–

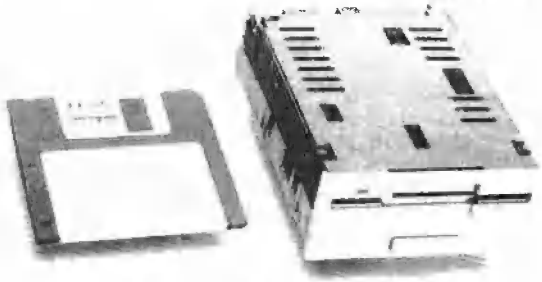
Preh Commander Keyboards

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

Preh Commander Keyboard, frei programmierbar bis zu 10 Ebenen, pro Taste bis zu 250 Zeichen . . . . . nur **599,–**

TEAC 3½" Laufwerk FD 35 F 535,–

Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur **535,–**



TEAC FD 55 AV 1 x 40 Track	425,–	**TEAC FD 55 BV** 2 x 40 Track	460,–
TEAC FD 55 EV 1 x 80 Track	445,–	**TEAC FD 55 FV** 2 x 80 Track	490,–

SONY 3½" Laufwerk . . . . . nur **799,–**
Apple®-kompatibles Laufwerk incl. Gehäuse + Kabel **599,–**

320 KB Laufwerk für IIc **948,–**
640 KB Laufwerk für IIc **1090,–**

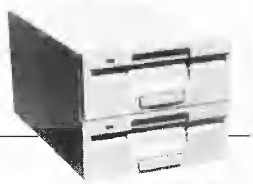
EPSON DRUCKER

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

Die Microfloppy mit Zukunft:

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

Low Power Version 1740,–



10 MB Winchester
mit Software für DOS 3.3, CP/M 2.20, Pascal, Pro-DOS, incl. Controler und Gehäuse **3990,–**

Sonderangebot
Distar Laufwerk für II + IIe,
incl. Kabel u. Gehäuse . . . . . jetzt nur **439,–**

Gesamt-Preisliste anfordern! **Preise inclusive gesetzlicher Mehrwertsteuer.**

UEDING electronics

Holtewiese 2
5750 Menden 1

DFÜ 02373/66877
Tel. 02373/63159

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 abschrecken, 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.

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

Apple ProDOS

Band 2

von Ulrich Stiehl
1985, 208 S., kart., DM 30,–

Der zweite Band von „ProDOS für Aufsteiger" ist wieder ein typisches Tips- und Tricks-Buch, das viele nützliche Utilities und Hilfsroutinen enthält, die dem Applesoft- und Assemblerprogrammierer den Umgang mit ProDOS erleichtern sollen.

Zunächst wird eine bewußt leichtverständliche Einführung in die Applesoft-Programmierung unter ProDOS gegeben, die sich insbesondere an diejenigen Leser wendet, die das alte DOS 3.3 nur noch „am Rande" kennengelernt haben.

Im Anschluß daran wird gezeigt, wie man einige Unzulänglichkeiten des BASIC-SYSTEMs durch Tricks beheben kann, z. B. Einlesen von Strings mit Komma und Doppelpunkt, Laden und Speichern von Zahlen als Binärdateien, Simulierung des fehlenden MON-Befehls u.a..

Den Hauptteil des Buches bilden sofort einsatzfähige Utilities, die alle klassischen „Pflichtübungen" abdecken, z. B. Dateileseprogramme mit ASCII- und Hex-Dump, Dateikopierprogramme, Diskettenformatier- und -kopierprogramme, Diskettenvergleichsprogramme, Konvertierungsprogramme (DOS 3.3 nach ProDOS), Bad-Block-Programme u.a.. Zur Anwendung dieser Utilities sind keinerlei Assemblerkenntnisse erforderlich.

Beachten Sie die Buch-Shop-Karte

BUCH-SHOP

Betriebssystem CP/M

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

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

Das Buch zum Apple II

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

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

Apple II ROM Listing

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

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

Auch für Apple-IIe und c und Kompatible!

Apple II Pascal

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

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

Start mit Apple-Logo für II, IIe und IIc

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

Viele Mikrocomputer-Hersteller bieten für ihre Geräte neben BASIC und anderen Programmiersprachen zunehmend auch Logo an. Durch ihre Benutzerfreundlichkeit hat diese Sprache bereits viele Freunde im Ausbildungs- und Freizeitbereich gefunden. Dabei ist Logo eine mächtige Sprache, die auch dem anspruchsvollen Anwender kaum Wünsche offenläßt.

Mittels Schildkrötengrafik wird das kleine Logo-Einmaleins in 12 Lektionen entwikkelt. Große Bildschirmfotos begleiten den Leser durch die Lektionen. Das Buch verlangt aktive Mitarbeit. Es hat seinen Platz neben dem Computer und gibt Hilfen und Anregungen für eigenes Forschen. Dank des bausteinorientierten Konzepts kann jeder seine eigenen Teilbausteine erzeugen und sie zu neuen Blöcken zusammenfügen. Neben dem Einmaleins werden neue Einsatzbereiche für den Einsteiger erschlossen. Musik und Sound fehlen nicht.
In diesem Buch werden die beiden offiziellen Logo-Produkte der Firma Apple für die Rechnerfamilie Apple II, IIe und IIc behandelt und deren Unterschiede verdeutlicht. Weiterhin sind sämtliche Apple-Logo-Vokabeln übersichtlich zusammengestellt. Dieses Buch ist ideal zum problemlosen und vergnüglichen Start in die Apple-Logo-Welt.

Apple Pascal Grafik

von Tom Swan
1984, 298 S., kart., DM 49,–

22 PASCAL-Programme, mit denen Sie die Grafik-Möglichkeiten Ihres APPLE voll ausschöpfen: DESIGNER läßt Sie eigene Zeichensätze entwerfen; GREDIT unterstützt Sie beim Entwerfen und Abspeichern kompletter Bildschirmgrafiken; PRINT-FOTO bringt Ihre Entwürfe aufs Papier; Darüberhinaus bietet das Buch eine Fülle fertiger Prozeduren, die Sie zeitsparend in Ihre eigenen Programme einbauen können.

Apple II Raster Grafik

von Jeffrey Stanton
1984, 306 S., kart., DM 49,–

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 Assemblerroutinen, mit denen Sie viele Effekte bekannter Spiele selbst programmieren können. Gute BASIC-Kenntnisse werden vorausgesetzt, eine Einführung in Assembler-Programmierung wird gegeben.

Apple II Schaltpläne

von Winston D. Gayler
1984, 215 S., kart., DM 64,–

Eine detaillierte Beschreibung der Apple II-Schaltungen.

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

„GOTO, Goto, und GoTo“ werden jetzt erkannt, beim Listen jedoch nach wie vor in Großbuchstaben ausgegeben (wegen der internen Token-Darstellung). Dadurch kann man auch auf einfache Weise DOS 3.3 vorübergehend abhängen: „pr#3“ wird vom DOS 3.3 nicht erkannt und somit direkt vom Monitor ausgeführt, was die DOS-Ausgabevektoren überschreibt. Mit Reset kann DOS wieder aktiviert werden.

3. Geändertes Listing

Beim Listen eines Programms wird nun wie beim Apple IIc vor der Zeilennummer stets ein Leerzeichen eingefügt. Dieses Leerzeichen wird aktuell ausgegeben und nicht etwa durch Manipulation des Cursor-Wertes gewonnen. Der Grund für diese Änderung (die Vermutung, das „hätte sich beim Patchen so ergeben“ hat sich nicht bestätigt) kann nur im „erhöhten Editierkomfort“ liegen. Wegen des Prompt-Zeichens mußte der Cursor bisher immer um ein Zeichen nach links verschoben werden, um an den Anfang der BASIC-Zeile zu gelangen. Dies ist nun nicht mehr der Fall.

4. HTAB-Befehl

Bisher wurde im 80-Z/Z-Modus der HTAB-Befehl nicht korrekt unterstützt. Der neue Interpreter tabuliert auch über die 40. Spalte hinaus. Das Textfenster wird immer noch ungebührend berücksichtigt, so daß nach wie vor eine PRINT-Anweisung das Programm „überdrucken“ kann, wenn das Textfenster nicht in der Spalte 1 ($0020 = $00) beginnt. So ist auch immer noch eine Tabulierung über die rechte Bildschirmbegrenzung hinweg möglich. Beim IIc wurde diese Änderung in ähnlicher Weise vollzogen.

Mit diesen Modifikationen dürften kaum Probleme mit vorhandener Software auftreten. Was jedoch sehr verwundert, ist die Tatsache, daß bekannte Bugs, die z.T. auf einen Tippfehler beim Programmmieren des Interpreters zurückzuführen sind (also nur ein Byte umfassen), nicht behoben wurden.
So wird das 5. Byte des RND-Startwertes beim Kaltstart immer noch nicht in die Zero-Page übertragen und das von Cornelis Bongers bereits in CALL-A.P.P.L.E beschriebene Problem mit FOR-NEXT-Schleifen in Unterprogrammen besteht weiterhin.
Noch ein letzter Bug, der nicht behoben wurde und allerdings auch nicht auf einen Tippfehler zurückzuführen ist: Geben Sie folgende Zeile ein und harren Sie der Dinge, die kommen werden:
LIST 437760

3. Standardeingabe von ESC-Sequenzen (Tastatureingabe)

```
Tastatureingabe in Applesoft über INPUT und GET
Tastatureingabe in Assembler über GETLN ($FD6A) und RDKEY ($FDØC)

Eine ESC-Sequenz wird durch Drücken der ESC-Taste
eingeleitet, gefolgt von einer weiteren Taste, die dann
zusammen mit der ESC-Taste die gewünschte Funktion auslöst.
Neben Großbuchstaben können jetzt auch Kleinbuchstaben
als Folgetasten verwendet werden.

ja = Man bleibt nach EINMALIGEM ESC-Drücken
     so lange im ESC-Modus, bis ein ungültiges Folgezeichen,
     z.B. Leertaste, eingegeben wird.

* = funktioniert nur nach PR#3 ($CXØØ-Routinen)
/ = Nach dem Schrägstrich sind Alternativtasten angegeben.
Tabelle gilt auch uneingeschränkt für den Apple IIc.
Beim alten Apple IIe entfallen ESC Ctrl-E, ESC Ctrl-F.
Dafür zusätzlich ESC Ctrl-R, ESC Ctrl-T.

ESC-Sequenz        Bleibt?  CX Wirkung
-------------------------------------------------------------------
ESC §/Shift 3      nein       HOME: CLRSCR und VTAB 1, HTAB 1
ESC A/a            nein       Cursor nach rechts
ESC B/b            nein       Cursor nach links
ESC C/c            nein       Cursor nach unten
ESC D/d            nein       Cursor nach oben
ESC F/f            nein       CLREOS: Clear to end of screen
ESC I/Hochpfeil    ja         Cursor nach oben
ESC J/Linkspfeil   ja         Cursor nach links
ESC K/Rechtspfeil  ja         Cursor nach rechts
ESC M/Tiefpfeil    ja         Cursor nach unten
ESC R                         entfällt (Nur Großb. beim alten IIe)
ESC T                         entfällt (Groß/Kleinb. beim alten IIe)
ESC 4              nein    *  4Ø Z/Z mit CX-Routinen
ESC 8              nein    *  8Ø Z/Z mit CX-Routinen
ESC Ctrl-D         nein    *  Keine Standardausgabe von Ctrl-Zeichen
                              (außer BEL, BS, LF, CR); n-IIe!
ESC Ctrl-E         nein    *  Standardausgabe von Ctrl-Zeichen; n-IIe!
ESC Ctrl-Q         nein    *  4Ø Z/Z mit F8-Routinen

(Andere ESC-Sequenzen sind nicht implementiert)
```

INT.IIE.NEU

```
                    1    * Disassembliertes Listing der gepatchten
                    2    * Stellen im Applesoft-Interpreter
                    3
                    4    ENDCHR   EQU  $ØE
                    5    DATAFLG  EQU  $13
                    6    WNDWDTH  EQU  $21
                    7    CH       EQU  $24
                    8    IN       EQU  $2ØØ
                    9    CH8Ø     EQU  $57B
                    1Ø
                    11   MEMERR   EQU  $D41Ø
                    12   CRDO     EQU  $DAFB
                    13   TABIT    EQU  $DB2B
                    14   NEXTCHR  EQU  $DB2F
                    15   OUTDO    EQU  $DB5C
                    16   SYNERR1  EQU  $DB83
                    17   GETBYTC  EQU  $E6F5
                    18   GETBYT   EQU  $E6F8
                    19   LINPRT   EQU  $ED24
                    2Ø
                    21            ORG  $D56D
D56D: 2Ø 8C F7      22            JSR  PPATCH2    ;-> LDA IN,X
                    23
                    24            ORG  $D5A8
D5A8: 2Ø 8C F7      25            JSR  PPATCH2    ;-> LDA IN,X
                    26
                    27            ORG  $D5BE
D5BE: 2Ø 87 F7      28            JSR  PPATCH1    ;-> LDA IN+1,X
                    29
                    3Ø            ORG  $D5E9
D5E9: 2Ø 8C F7      31            JSR  PPATCH2    ;-> LDA IN,X
                    32
                    33            ORG  $D6ØB
D6ØB: 2Ø 9A F7      34            JSR  PPATCH3    ;-> LDA IN,X
                    35
                    36            ORG  $D6FA
D6FA: 2Ø AA F7      37            JSR  LSTPTCH    ;-> JSR LINPRT
                    38
                    39            ORG  $D7Ø5
D7Ø5: 2Ø B4 F7      4Ø            JSR  $F7B4      ;-> LDA CH
D7Ø8: EA            41            NOP             ;   CMP #33
                    42
                    43            ORG  $DADB
DADB: FØ 3C         44            BEQ  TABWHERE   ;geänderte Adresse
```

```
                        45
                        46              ORG   $DAEØ
DAEØ: FØ 37             47              BEQ   TABWHERE     ;geänderte Adresse
                        48
                        49              ORG   $DBØ3
DBØ3: 2Ø B4 F7          5Ø    TAB       JSR   TABPTCH      ;Komma-Tabulierung
DBØ6: 3Ø Ø9             51              BMI   NXTCLM
DBØ8: C9 18             52              CMP   #4Ø-16
DBØA: 9Ø Ø5             53              BCC   NXTCLM
DBØC: 2Ø FB DA          54              JSR   CRDO
DBØF: DØ 1E             55              BNE   NEXTCHR     ;unbedingt
DB11: 69 1Ø             56    NXTCLM    ADC   #16         ;neue TAB,-Position
DB13: 29 FØ             57              AND   #%1111ØØØØ  ; bestimmen
DB15: AA                58              TAX
DB16: 38                59              SEC
DB17: BØ ØC             6Ø              BCS   TAB1
DB19: Ø8                61    TABWHERE  PHP
DB1A: 2Ø F5 E6          62              JSR   GETBYTC
DB1D: C9 29             63              CMP   #')'
DB1F: DØ 62             64              BNE   SYNERR1
DB21: 28                65              PLP
DB22: 9Ø Ø7             66              BCC   TABIT
DB24: CA                67              DEX
DB25: 2Ø C3 F7          68    TAB1      JSR   TABPTCH1
                        69
                        7Ø    * Das folgende Byte dürfte zukünftig als
                        71    * ID-Byte für den Interpreter dienen:
                        72    *
                        73    * $ØØ = II Plus/IIe (alt)
                        74    * $C4 = IIe (neu)
                        75    * $89 = IIc
                        76
                        77              ORG   $EØØ6
E0Ø6: C4                78              DFB   $C4         ;-> DFB $ØØ
                        79
                        8Ø              ORG   $F3B6
F3B6: 2Ø 77 F7          81              JSR   TAPEPNT     ;-> JSR $F7BC
                        82
                        83              ORG   $F3D2
F3D2: 2Ø 77 F7          84              JSR   TAPEPNT     ;-> JSR $F7BC
                        85
                        86              ORG   $F775
F775: 38                87    SHLOAD    SEC
F776: 9Ø                88              DFB   $9Ø         ;täuscht vor:
                        89    *         BCC   $F79Ø
F777: 18                9Ø    TAPEPNT   CLC               ;<- STORE/RECALL
F778: 8D Ø7 CØ          91              STA   $CØØ7       ;CX-Firmware
F77B: 2Ø ØØ C5          92              JSR   $C5ØØ       ; aufrufen
F77E: 8D Ø6 CØ          93              STA   $CØØ6       ;Slot-ROM aktiv.
F781: BØ Ø1             94              BCS   MEMERR1
F783: 6Ø                95              RTS
F784: 4C 1Ø D4          96    MEMERR1   JMP   MEMERR
                        97
F787: BD Ø1 Ø2          98    PPATCH1   LDA   IN+1,X      ;<- PARSE
F78A: 1Ø 11             99              BPL   PARSE1
F78C: A5 ØE             1ØØ   PPATCH2   LDA   ENDCHR      ;<- PARSE
F78E: FØ 16             1Ø1             BEQ   PARSE3
F79Ø: C9 22             1Ø2             CMP   #'"'
F792: FØ 12             1Ø3             BEQ   PARSE3
```

```
F794: A5 13             1Ø4             LDA   DATAFLG
F796: C9 49             1Ø5             CMP   #$49
F798: FØ ØC             1Ø6             BEQ   PARSE3
F79A: BD ØØ Ø2          1Ø7   PPATCH3   LDA   IN,X        ;<- PARSE
F79D: Ø8                1Ø8   PARSE1    PHP
F79E: C9 61             1Ø9             CMP   #'a'
F7AØ: 9Ø Ø2             11Ø             BCC   PARSE2
F7A2: 29 5F             111             AND   #%Ø1Ø11111
F7A4: 28                112   PARSE2    PLP
F7A5: 6Ø                113             RTS
                        114
F7A6: BD ØØ Ø2          115   PARSE3    LDA   IN,X
F7A9: 6Ø                116             RTS
                        117
F7AA: 48                118   LSTPTCH   PHA
F7AB: A9 2Ø             119             LDA   #' '        ;JSR OUTSPC hätte
F7AD: 2Ø 5C DB          12Ø             JSR   OUTDO       ; es auch getan
F7BØ: 68                121             PLA
F7B1: 4C 24 ED          122             JMP   LINPRT
                        123
F7B4: A5 24             124   TABPTCH   LDA   CH          ;<- LIST/TAB
F7B6: C9 21             125             CMP   #39-6       ;für LIST
F7B8: 2C 1F CØ          126             BIT   $CØ1F       ;8Ø Zeichen?
F7BB: 1Ø Ø5             127             BPL   IS4ØCLM     ;nein, dann weiter
F7BD: AD 7B Ø5          128             LDA   CH8Ø
F7CØ: C9 49             129             CMP   #79-6       ;für LIST
F7C2: 6Ø                13Ø   IS4ØCLM   RTS
                        131
F7C3: 8A                132   TABPTCH1  TXA
F7C4: 2C 1F CØ          133             BIT   $CØ1F       ;8Ø Zeichen?
F7C7: 3Ø Ø8             134             BMI   IS8ØCLM     ;ja, dann weiter
F7C9: 2C                135             DFB   $2C         ;täuscht vor:
                        136   *         BIT   $2485
F7CA: 85 24             137   HTABPTCH  STA   CH
F7CC: 38                138             SEC
F7CD: 8A                139             TXA
F7CE: E5 24             14Ø             SBC   CH
F7DØ: 6Ø                141   RETURN    RTS
F7D1: ED 7B Ø5          142   IS8ØCLM   SBC   CH8Ø
F7D4: 6Ø                143             RTS
                        144
F7D5: ØØ                145             BRK
F7D6: ØØ                146             BRK
F7D7: ØØ                147             BRK
F7D8: ØØ                148             BRK
                        149
                        15Ø             ORG   $F7E7
F7E7: 2Ø F8 E6          151   HTAB      JSR   GETBYT      ;HTAB-Routine
F7EA: CA                152             DEX
F7EB: A9 28             153   HTABLOOP  LDA   #4Ø
F7ED: C5 21             154             CMP   WNDWDTH
F7EF: BØ Ø2             155             BCS   HTABWND
F7F1: A5 21             156             LDA   WNDWDTH
F7F3: 2Ø CA F7          157   HTABWND   JSR   HTABPTCH
F7F6: 86 24             158             STX   CH
F7F8: 9Ø D6             159             BCC   RETURN
F7FA: AA                16Ø             TAX
F7FB: 2Ø FB DA          161             JSR   CRDO
F7FE: DØ EB             162             BNE   HTABLOOP    ;unbedingt

195 Bytes
```

EPROM-Programmiergerät für den Apple II

von Dr. Roland Schulé

Foto 1: EPROM-Gerät im Gehäuse

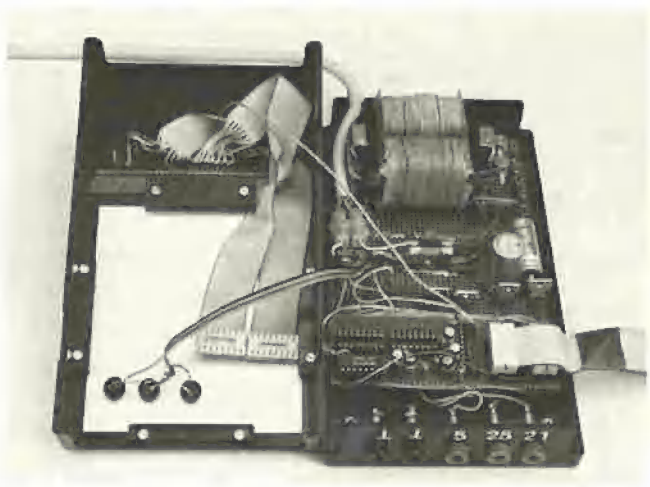


Foto 2: EPROM-Gerät mit geöffnetem Gehäuse

Es hat in der Literatur sicherlich noch nicht an Bauvorschlägen für ein EPROM-Programmiergerät gemangelt (EPROM = Erasable Programmable Read Only Memory). Dennoch gibt es Gründe, ein Instrument wie dieses vorzustellen. Zum einen eignet es sich hervorragend als Projekt für jemanden, der in die Thematik der Erweiterungskarten des Apple II einsteigen will. Zum anderen liegt die Besonderheit dieses EPROM-Programmiergeräts darin, daß der gesamte Software-Kern auf der Erweiterungskarte Platz findet. Damit braucht er nicht eigens geladen zu werden und nimmt kein einziges Byte an Speicherplatz weg. Jeglicher Teil des Apple-Speichers kann somit unmittelbar in ein EPROM gebrannt werden.

1. Adreßbereich der Apple-Erweiterungskarten

Alle Ein- und Ausgaben des Apple II sind in dem Adreßbereich $C000 bis $CFFF angesiedelt. Dies ist der Bereich, den wir für unser Erweiterungskartenprojekt genauer betrachten müssen. Die Adressen $C000 bis $C07F werden von den eingebauten Ein/Ausgabekanälen belegt, z.B. dem Kassettenrekorder-Interface, dem eingebauten Lautsprecher und der Tastatur.

Die nachfolgenden Adressen sind von größerem Interesse für uns. Jedem Steckplatz ist ein definierter Speicherbereich von 16 Bytes zugewiesen (s. **Tabelle 1**). Diese sind für den eigentlichen Ein/Ausgabebaustein gedacht. Bei dem in unserem Projekt verwendeten Baustein VIA 6522 (VIA = Versatile Interface Adapter) geht die Rechnung exakt auf; er hat ebenfalls 16 Register. Daher genügt es, die vier Adreßleitungen des Steckers mit den entsprechenden „Register select"-Eingängen RS0 bis RS3 des VIA 6522 zu verbinden. Zusätzlich wird noch die Busleitung $\overline{\text{DEVICE SELECT}}$ mit dem Eingang $\overline{\text{CS2}}$ verbunden (s. **Abb. 1**).

Die Datenrichtung wird über die Leitung R/$\overline{\text{W}}$ (Read/Write) gesteuert. Zu guter Letzt muß der VIA 6522 noch seinen Takt erhalten, mit dem er arbeiten soll. Da das geforderte Signal Φ2 auf dem Apple-Bus nicht vorhanden ist, erzeugen wir es durch Verzögerung des Signals Φ0. Hierzu tut erfahrungsgemäß die Gatterlaufzeit eines CMOS-Treibers 4050 (CMOS = Complementary Metal Oxide Semiconductor) gerade die richtigen Dienste.

Mit einem Ein/Ausgabebaustein allein ist jedoch noch kein Interface fertig. In aller Regel gehört zu dem Baustein noch ein Treiberprogramm, um die entsprechende Steuerung der Ein- und Ausgabesignale zu veranlassen. Der Apple reserviert hierzu pro Steckplatz einen Adreßbereich von 256 Bytes, um ein entsprechendes Treiberprogramm *auf* der Erweiterungskarte unterzubringen (s. Tabelle 1).

Nun sind 256 Bytes zugegebenermaßen nicht viel, und deshalb sind im Adreßraum

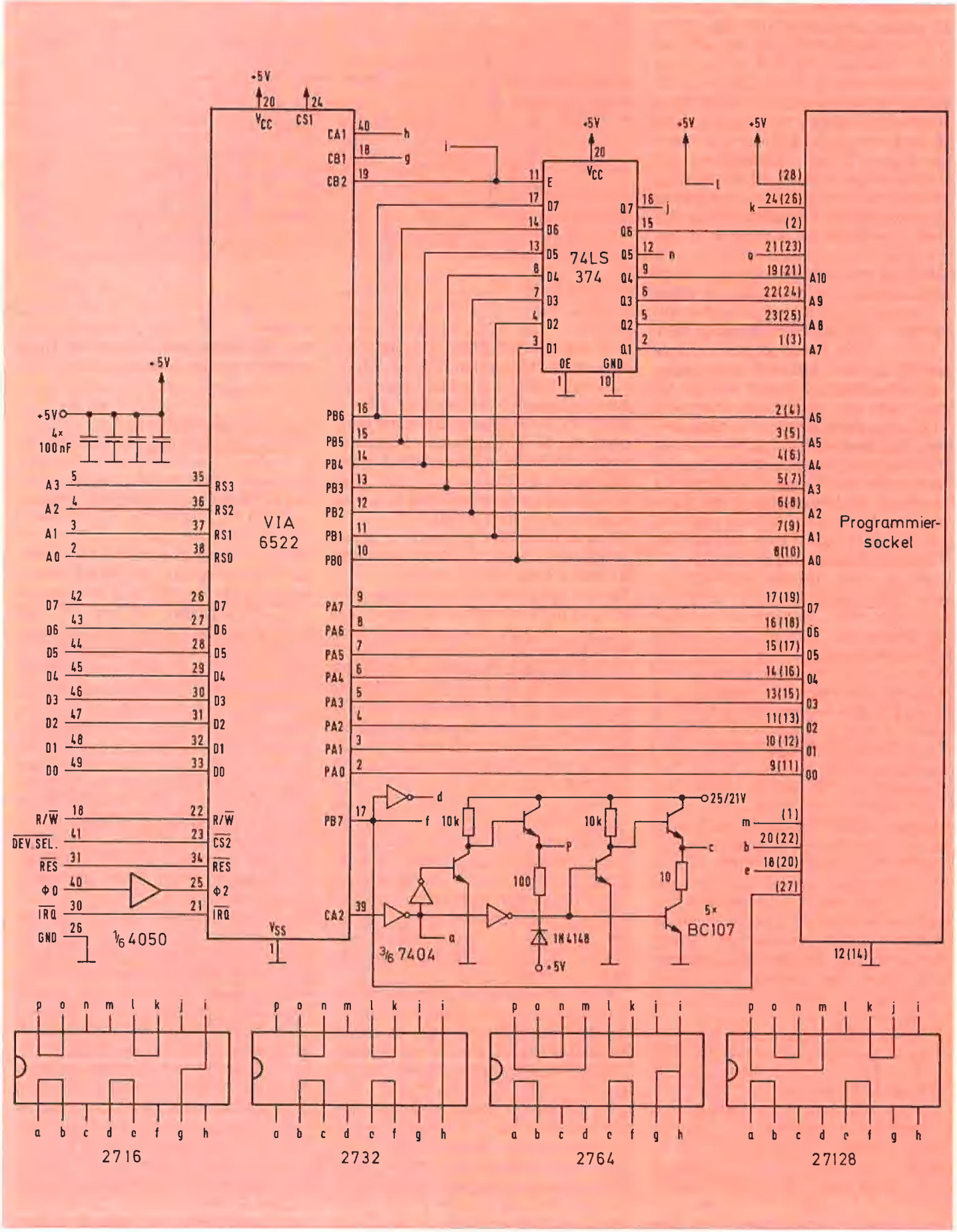


Abb. 1: Schaltung des EPROM-Interface

des Apple II nochmals volle 2K reserviert. Es handelt sich dabei um die obere Hälfte des Ein/Ausgabebereichs, $C800 bis $CFFF. Die Auswahl erfolgt nach einem Banking-Prinzip.
Für das hier beschriebene Projekt wurde allerdings die Treibersoftware in die 256 Bytes des der Erweiterungskarte direkt zugeordneten ROMs hineingepackt. Die Software findet in einem 2K-EPROM Platz. Die Verschwendung von 1792 Bytes sollte uns nicht grämen; EPROMs vom Typ 2716 sind nun einmal am leichtesten zu beschaffen.
Das Programm braucht auch noch einige Speicherzellen, in die es Zeiger, Daten etc. ablegen kann. Auch hierfür ist in dem Aufbau des Apple II bestens gesorgt. Ganze 8 Bytes RAM stehen pro Steckplatz zur Verfügung. Und dieser Ausbau genügt auch tatsächlich, wenn man sich auf ein paar Werte beschränkt, die über mehrere Programmaufrufe erhalten bleiben müssen. Platz für Zwischenspeicher finden wir dann noch zur Genüge in dem Betriebssystem-RAM. Insbesondere ist die Benutzung von Speicherplätzen in der Zero-Page von $0000 bis $00FF von Interesse, da der 6502 dann eine kürzere und schnellere Adressierung ermöglicht.
Zum Abschluß unserer Überlegungen über Speicherplatz wollen wir uns noch vor Augen halten, was passiert, wenn wir die Erweiterungskarte in einen anderen Steckplatz stecken. Zunächst einmal verändern sich alle Adressen. Diese wurden jedoch in dem Programm verwendet, und deshalb funktioniert jetzt gar nichts mehr. Gewiefte Programmierer kennen Tricks, mit deren Hilfe das Programm einer Erweiterungskarte erfahren kann, in welchem Steckplatz es sich befindet. Wenn dann das Programm ohne jeglichen Bezug auf absolute Speicheradressen geschrieben ist, d.h. nur Branch-Befehle verwendet und mit Zeigern arbeitet, kann es „positionsunabhängig" bzw. relokativ sein. Allerdings muß man sich vor Augen halten, daß eine derartige Programmierung mehr Platz in Anspruch nimmt. Wir sind diesen Schwierigkeiten aus dem Weg gegangen und haben das Treiberprogramm auf Steckplatz 4 ausgelegt.

2. Programmieren von EPROMs

EPROMs sind die idealen Festwertspeicher für denjenigen, der nur eine geringe Auflage an ROMs benötigt, vielleicht sogar nur ein einziges für sein Programm, das als Steckmodul zur Verfügung stehen soll. EPROMs werden programmiert, indem eine deutlich höhere Spannung, als sonst bei Gattern, Mikroprozessoren und Speichern üblich ist, an den Baustein angelegt

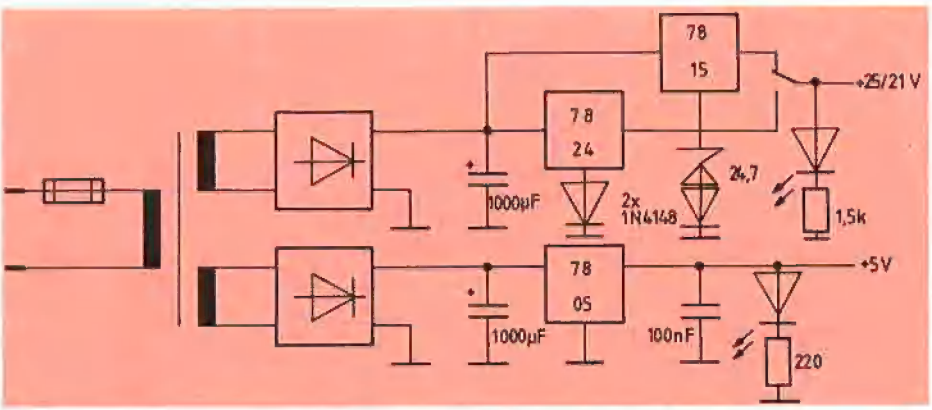


Abb. 3: Netzteil des EPROM-Gerätes

wird. Bei den EPROMs 2716 und 2732 sind es 25 Volt, bei den EPROMs 2732A sind es 21 Volt. Beide hier betrachteten Speicherbausteine haben eine nahezu identische Pin-Belegung; sie folgen beide dem sog. Byte-Wide-Konzept. Dies trifft auch noch für die größeren EPROMs dieser Familie (2764 und 27128) zu. Unser EPROM-Programmiergerät läßt sich auch noch auf jene Typen erweitern.
Wenn Unterschiede auftreten, so liegen sie gerade in der Art und Weise, wie diese Bausteine programmiert werden.
Bei dem 2716 muß vor dem Programmiervorgang der Anschluß $\overline{OE}$ auf high gelegt werden, um die Ausgangstreiber des EPROMs abzuschalten. An den Anschluß Vpp, an dem normalerweise ein 5V-Pegel liegt, wird nun die Programmierspannung von 25 Volt angelegt. Nun wird die Adresse des zu programmierenden Bytes an den Anschlüssen A0 bis A10 eingestellt und die gewünschten Daten auf O0 bis O7 gelegt. Zu guter Letzt wird der Eingang $\overline{CE}$/PGM für eine Dauer von 50 ms auf ein Pegel von 5 V geschaltet (s. **Abb. 3**). Diese Programmierzeit muß exakt eingehalten werden. Ist sie zu lang, kann das EPROM zerstört werden. Ist sie zu kurz, werden die Daten nicht richtig programmiert und können nach einiger Zeit wieder verlorengehen. Die Programmierzeit werden wir mit Hilfe des Zeitgeberteils des VIA 6522 bestimmen.
Bei den 4K-Bausteinen 2732 und 2732A wird ein zusätzliches Adreßbit A11 benötigt. Aus diesem Grund wurde bei diesen Bausteinen der $\overline{OE}$- und der Vpp-Eingang zusammengelegt. Außerdem wird der Programmierimpuls als 0V-Pegel an den Eingang $\overline{CE}$ gelegt, gerade umgekehrt also wie bei dem 2716. Das Programmier-Impulsdiagramm ist in **Abb. 4** gezeigt.
Ausgelesen werden die beiden Speicherbausteine auf die gleiche Art und Weise. Dazu muß die Adresse an die Adreßleitungen gelegt und sowohl der $\overline{OE}$- als auch der $\overline{CE}$-Eingang auf low gelegt werden. Die Daten des angewählten Bytes erscheinen an den Leitungen O0 bis O7.

3 Aufbau des EPROM-Programmiergeräts

Wie zuvor schon erwähnt, werden wir für das EPROM-Programmiergerät den Baustein VIA 6522 verwenden. Dieser besitzt zwei Ein/Ausgaberegister zu je 8 Bits, die jeweils noch von zwei Handshake-Leitungen flankiert werden. Wir werden alle 20 Bits benutzen. An den Port A schließen wir die Datenleitungen O0 bis O7 des EPROMs an. Der Ausgang des Zeitgebers im VIA 6522 liegt auf Bit 7 des Port B. Diesen Ausgang werden wir zur Erzeugung des Programmierimpulses benutzen. Das Ein- und Ausschalten der Programmierspannung haben wir dem Handshake-Ausgang CA2 zugedacht. Die verbleibenden Leitungen des VIA 6522 reichen nicht aus, um die Adreßleitungen des EPROMs direkt anzusteuern. Hier benutzen wir einen 8-Bit-TTL-Speicher (TTL = Transistor-Transistor-Logik), um einen Teil der Adressen zwischenzuspeichern. Als Speicherimpuls wird das Signal am Handshake-Ausgang CB2 verwendet.
Diejenigen Leitungen, die sich bei den verschiedenen EPROMs unterscheiden, werden wie üblich an einen IC-Sockel gelegt. Durch einen aufgesteckten IC-Stekker werden die für das jeweilige EPROM notwendigen Verbindungen hergestellt. Über solche Schalter wird auch der Speicherimpuls von CB2 an die beiden Handshake-Eingänge CA1 und CA2 zurückgeführt. Durch unterschiedliche Belegung bei den einzelnen Steckern kann von dem Programm der Steckertyp und damit der Typ des EPROMs erkannt werden.
Wie schon oben erwähnt, befindet sich die Software des Programmiergeräts ebenfalls auf der Erweiterungskarte. Die Beschaltung des EPROMs weist keine Besonderheiten auf. Da der Adreßraum nur 256

Bytes beträgt, werden nur die Adreßleitungen A0 bis A7 mit den entsprechenden Leitungen des Apple-Bus verbunden. Zur Auswahl des Bausteins wird das Signal I/O-SELECT zu dem Eingang CS des EPROMs geführt. A8 bis A10 sind auf Schiebeschalter (DIP-Switches) gelegt, so daß bis zu acht verschiedene Programme in dem EPROM angewählt werden können. Hier ließe sich z.B. siebenmal das gleiche Programm unterbringen, jedoch jeweils für Steckplatz 1 bis 7 assembliert. Damit wäre im nachhinein doch noch die Positionsunabhängigkeit erreicht. Bei dem abgebildeten Gerät des Autors ist die Schaltung jedoch in eine universelle Parallelport-Platine und das eigentliche Programmiergerät aufgeteilt. Dieses befindet sich mit dem Adreßspeicherbaustein 74LS374 und dem Netzteil in einem externen Gehäuse. Dies hat einerseits den Vorteil, daß der Programmiersockel gut zugänglich ist. Andererseits kann die VIA-6522-Karte auch alleine oder mit anderer Peripherie benutzt werden. In diesem Fall wird über die Schiebeschalter einfach das dazugehörige Programm angewählt.

Zusätzlich kann das EPROM noch ganz abgeschaltet werden, indem der Schalter an dem OE-Eingang geöffnet wird.
Der Aufbau des Geräts wurde auf einer im Handel erhältlichen Apple-Experimentierkarte und einer Lochrasterplatine vorgenommen.

4. Die Treibersoftware

Eingangs hatten wir versprochen, daß der Kern der Treibersoftware kein einziges Byte des Apple-Speichers wegnehmen wird. Dies ist möglich, da die eigentlichen EPROM-Routinen auf der Erweiterungskarte liegen. Es handelt sich dabei um das Programm **EPROM** mit den Routinen EPROM LESEN, EPROM LEERTEST, EPROM PROGRAMMIEREN und EPROM VERGLEICHEN. Der Aufruf der Routinen erfolgt am besten vom Monitor des Apple II. Dazu wird die RAM-Anfangsadresse in den Speicherzellen $047C und $04FC (Low-Byte und High-Byte) abgelegt. Die Anfangsadresse des EPROMs kommt nach $057C/$05FC und die Endadresse + 1 nach $067C/$06FC. Bei $077C wird schließlich der EPROM-Typ eingetragen: $00 für den 2716, $80 für die übrigen.
Danach kann die gewünschte Routine aufgerufen werden. Die in dem Scratchpad-RAM (= Screenholes = freien Stellen des Bildschirmspeichers) abgelegten Daten bleiben unverändert, so daß die Bedienung des EPROM-Programmiergeräts nicht zu kompliziert ist.
Dennoch wurde zu dem Maschinenprogramm noch das Applesoft-Rahmenprogramm **EPROMMER** mit einer komfortablen Menüsteuerung geschrieben. Dieses Programm leistet neben dem Aufruf der oben beschriebenen Routinen noch folgende Dienste:

- Anfangs- und Endadressen setzen.
- Erkennen des EPROM-Typs über den Auswahlstecker.
- Vorbesetzen des Speichers mit $FF, um Leerbereiche eines EPROMs nicht unnötig zu programmieren.
- Binär-Files von der Diskette laden.
- Binär-Files auf die Diskette abspeichern.
- Disketteninhaltsverzeichnis darstellen.

Gleichgültig, ob Sie die Maschinenroutinen direkt aufrufen oder das Applesoft-

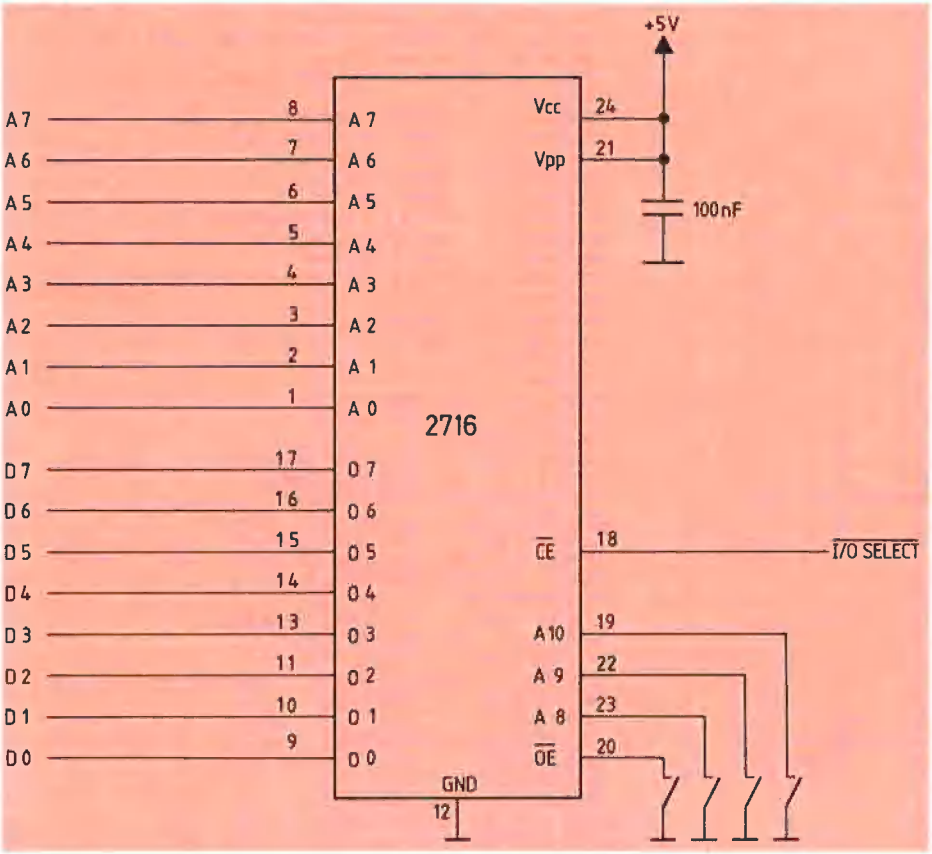


Abb. 2: Schaltung des EPROMs

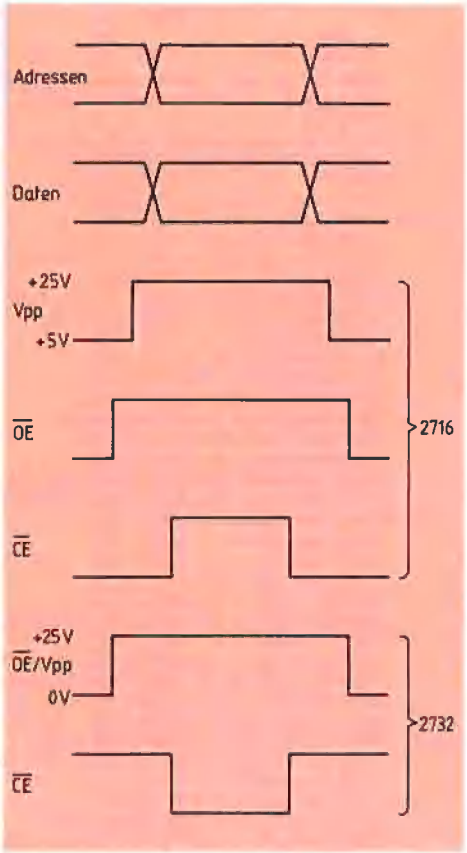


Abb. 4: Impulsdiagramm des Programmiervorgangs

Programm benutzen: In beiden Fällen findet eine Fehlerüberprüfung durch die Routinen statt. Eine Fehlermeldung kann z.B. beim EPROM LEERTEST ausgelöst werden, wenn ein Byte des zu überprüfenden EPROM-Bereichs nicht auf $FF gesetzt ist. Die Routine EPROM VERGLEICHEN reagiert mit einer Fehlermeldung, falls eine Diskrepanz zwischen EPROM und RAM vorliegt. Da die Routine EPROM PROGRAMMIEREN direkt in die Routine EPROM VERGLEICHEN übergeht, wird auch gleich nach dem Programmieren die Korrektheit des Ergebnisses überprüft.
Alle Adressen, wie Sie hier angegeben sind, beruhen auf der Steckplatznummer 4. Durch Ändern des Slot-Parameters im Quelltext läßt sich auch jede andere Steckplatznummer einstellen.

Wie kommt man jedoch zu dem EPROM auf der Steckkarte, wenn dies das einzige EPROM-Programmiergerät ist, auf das man Zugriff hat? In diesem Fall muß das Programm für seinen ersten Einsatz in den RAM-Bereich verlegt werden, indem z.B. ORG $8000 gesetzt wird. Die Adresse des VIA 6522 darf jedoch nicht verschoben werden. Auf diese Weise kann das EPROM-Programm sich selbst in Ihr erstes EPROM kopieren. Nehmen wir an, daß es nicht das letzte sein wird.

Kurzhinweise

1. Zweck:
Aufbau eines Eprommers mit Hard- und Firmware zum Brennen eigener EPROMs.

2. Konfiguration:
Apple II+ und IIe (nicht IIc wegen Zusatzkarte). DOS 3.3 (kein ProDOS wegen HIMEM-Änderung)

3. Test:
Nach Installierung der Hardware und Einsetzen eines EPROMs vom Typ 2716, 2732, 2732A
RUN EPROMMER

4. Sammeldisk:
EPROMMER
(Applesoft-Rahmenprogramm)
EPROM
(Maschinenprogramm für Firmware)
T.EPROM
(Big-Mac-Quelltext).

Tabelle 1: Ein/Ausgabespeicherbereiche des Apple II

Steckplatz	Peripheriebaustein	Treiberprogramm	Scratchpad-RAM
0	$C080-$C08F		
1	$C090-$C09F	$C100-$C1FF	$479, $4F9, ..., $7F9
2	$C0A0-$C0AF	$C200-$C2FF	$47A, $4FA, ..., $7FA
3	$C0B0-$C0BF	$C300-$C3FF	$47B, $4FB, ..., $7FB
4	$C0C0-$C0CF	$C400-$C4FF	$47C, $4FC, ..., $7FC
5	$C0D0-$C0DF	$C500-$C5FF	$47D, $4FD, ..., $7FD
6	$C0E0-$C0EF	$C600-$C6FF	$47E, $4FE, ..., $7FE
7	$C0F0-$C0FF	$C700-$C7FF	$47F, $4FF, ..., $7FF

EPROM

```
1000 REM Eprommer-BASIC-Rahmenprogramm
1010 REM Dr. Roland Schule, 1984
1020 REM
1030 REM Menüsteuerung der Routinen
1040 REM in der Eprommer-Karte Slot 4.
1050 REM
1060 REM Freispeicher schaffen
1070 HIMEM: 8192
1080 REM Konstanten
1090 DATA 4096,256,16,1
1100 READ H(0),H(1),H(2),H(3)
1110 REM Hex Characters
1120 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
1130 DIM C$(15)
1140 FOR C = 0 TO 15
1150 READ C$(C): NEXT
1160 REM Aufrufadressen
1170 DATA 50176,50198,50225,50309
1180 READ LIES,TESTEN,PROG,VERGL
1190 REM Scratchpad-RAM Slot 4
1200 DATA 1148,1276,1404,1532,1660,1788,1916
1210 READ LRAM,HRAM,LSROM,HSROM,LEROM,HEROM,RTYP
1220 REM VIA 6522
1230 DATA 49344
1240 READ DEV
1250 PB = DEV: REM B-Port
1260 PA = DEV + 1: REM A-Port
1270 DB = DEV + 2: REM Datenrichtung B
1280 DA = DEV + 3: REM Datenrichtung A
1290 PC = DEV + 12: REM Kontrollregister
1300 FL = DEV + 13: REM Flaggenregister
1310 REM Ersatzwerte ablegen (2716)
1320 POKE LRAM,0: POKE HRAM,32: REM Startadr. $2000
1330 POKE LSROM,0: POKE HSROM,0: REM Startadr. $0000
1340 POKE LEROM,0: POKE HEROM,8: REM Endadr. + 1 $0800
1350 POKE RTYP,0
1360 REM Test auf EPROM-Typ
1370 POKE DB,255: POKE DA,0
1380 POKE PC,174: POKE PB,0
1390 TYP = PEEK (FL)
1400 POKE PC,238
1410 POKE PB,0:DUMMY = PEEK (PA)
1420 REM EPROM-Typ 2732
1430 IF TYP = 2 THEN POKE HEROM,16: POKE RTYP,128
1440 REM -----Menü-----
1450 HOME
1460 PRINT "EPROM"
1470 VTAB 5
1480 PRINT "EPROMTYP IST ";
1490 IF TYP = 2 THEN PRINT "2732": GOTO 1520
1500 IF TYP = 16 THEN PRINT "2716": GOTO 1520
1510 PRINT "NICHT ERKANNT"
1520 PRINT "RAM ANFANGSADRESSE    $";
1530 LN = PEEK (LRAM):HN = PEEK (HRAM): GOSUB 2330
1540 PRINT "EPROM ANFANGSADRESSE $";
1550 LN = PEEK (LSROM):HN = PEEK (HSROM): GOSUB 2330
1560 PRINT "EPROM ENDADRESSE+1    $";
1570 LN = PEEK (LEROM):HN = PEEK (HEROM): GOSUB 2330
1580 VTAB 11
1590 PRINT "A - ADRESSEN FESTLEGEN"
1600 PRINT "E - EPROM TYP FESTSTELLEN"
1610 PRINT "P - EPROM PROGRAMMIEREN"
1620 PRINT "R - EPROM LESEN"
1630 PRINT "T - EPROM LEERTEST"
1640 PRINT "V - EPROM MIT RAM VERGLEICHEN"
1650 PRINT "F - RAM MIT $FF VORBESETZEN"
1660 PRINT "L - DISK FILE LADEN"
1670 PRINT "S - DISK FILE ABSPEICHERN"
1680 PRINT "C - CATALOG"
1690 PRINT "X - MONITOR"
1700 PRINT "Z - ENDE"
1710 GET K$
1720 REM -----Adressen-----
1730 IF K$ < > "A" THEN 1880
1740 PRINT "RAM ANFANGSADRESSE ";
1750 GOSUB 2400
1760 HN = INT (Z / 256): POKE HRAM,HN
1770 LN = Z - HN * 256: POKE LRAM,LN
1780 PRINT "EPROM ANFANGSADRESSE";
1790 GOSUB 2400
1800 HN = INT (Z / 256): POKE HSROM,HN
1810 LN = Z - HN * 256: POKE LSROM,LN
1820 PRINT "EPROM ENDADRESSE +1 ";
1830 GOSUB 2400
1840 HN = INT (Z / 256): POKE HEROM,HN
1850 LN = Z - HN * 256: POKE LEROM,LN
1860 GOTO 1450
1870 REM -----Catalog-----
1880 IF K$ < > "C" THEN 1940
1890 PRINT
1900 PRINT CHR$ (4);"CATALOG"
1910 GET K$
1920 GOTO 1450
1930 REM -----BLOAD File-----
1940 IF K$ < > "L" THEN 1990
1950 GOSUB 2510
1960 PRINT CHR$ (4);"BLOAD";F$
1970 GOTO 1450
```

```
198Ø REM -----BSAVE File-----
199Ø IF K$ < > "S" GOTO 2Ø6Ø
2ØØØ GOSUB 251Ø
2Ø1Ø MA = PEEK (LRAM) + 256 * PEEK (HRAM)
2Ø2Ø ML = PEEK (LEROM) - PEEK (LSROM) + 256 * ( PEEK (HEROM)
     - PEEK (HSROM))
2Ø3Ø PRINT CHR$ (4);"BSAVE";F$;",A";MA;",L";ML
2Ø4Ø GOTO 145Ø
2Ø5Ø REM -----EPROM-Typ-----
2Ø6Ø IF K$ < > "E" THEN 2Ø9Ø
2Ø7Ø GOTO 132Ø
2Ø8Ø REM -----EPROM Lesen-----
2Ø9Ø IF K$ < > "R" THEN 212Ø
21ØØ CALL LIES: GOTO 145Ø
211Ø REM -----EPROM Testen-----
212Ø IF K$ < > "T" THEN 215Ø
213Ø CALL TESTEN: GOTO 145Ø
214Ø REM -----EPROM Programmieren-----
215Ø IF K$ < > "P" THEN 218Ø
216Ø CALL PROG: GOTO 145Ø
217Ø REM -----EPROM Vergleichen-----
218Ø IF K$ < > "V" THEN 221Ø
219Ø CALL VERGL: GOTO 145Ø
22ØØ REM -----RAM Vorbesetzen-----
221Ø IF K$ < > "F" THEN 228Ø
222Ø MA = PEEK (LRAM) + 256 * PEEK (HRAM)
223Ø ME = MA + PEEK (LEROM) - PEEK (LSROM) + 256 * ( PEEK
     (HEROM) - PEEK (HSROM))
224Ø FOR M = MA TO ME
225Ø POKE M,255: NEXT
226Ø GOTO 145Ø
227Ø REM -----Monitor-----
228Ø IF K$ < > "X" THEN 231Ø
229Ø CALL - 151
23ØØ REM -----Programmende-----
231Ø IF K$ < > "Z" THEN 145Ø
232Ø END
233Ø REM Drucke 2-Byte-Hexzahl
234Ø AR = HN * 256 + LN
235Ø FOR J = Ø TO 3
236Ø H1 = INT (AR / H(J)): PRINT C$(H1);
237Ø AR = AR - H1 * H(J)
238Ø NEXT
239Ø PRINT : RETURN
24ØØ REM Hexadezimaleingabe
241Ø INPUT "? $";X$
242Ø Z = Ø:L = LEN (X$)
243Ø IF L > 4 THEN 241Ø
244Ø FOR J = 1 TO L
245Ø N = ASC ( MID$ (X$,J,J)) - 48
246Ø IF N > 9 THEN N = N - 7
247Ø IF N > 15 THEN 241Ø
248Ø Z = Z + N * 16 ↑ (L - J): NEXT
249Ø RETURN
25ØØ REM Frage nach Filenamen
251Ø INPUT "FILENAME ?";F$
252Ø IF F$ < > "" THEN RETURN
253Ø POP : GOTO 145Ø: REM zurück zum Menü
```

EPROMMER

```
1    ********************************
2    *                              *
3    *  E P R O M - R O U T I N E N *
4    *                              *
5    *  Dr. Roland Schule, 1984     *
6    *                              *
7    ********************************
8    *
9    * Monitorroutinen:
1Ø   *
11   ERROR    EQU  $D42D      ;Fehlermeldung
12   *
13   * Erweiterungskarte in Slot 4
14   *
15   SLOT     =    4
16   *
17   * VIA-6522-Adressen:
18   *
19   DEV      EQU  $1Ø*SLOT+$CØ8Ø ;VIA 6522
2Ø   PORTB    EQU  DEV        ;Port B
21   PORTA    EQU  DEV+1      ;Port A
22   DDRB     EQU  DEV+2      ;Datenrichtung B
23   DDRA     EQU  DEV+3      ;Datenrichtung A
24   TIL      EQU  DEV+4      ;Timer, Low
25   TIH      EQU  DEV+5      ;Timer, High
26   ACR      EQU  DEV+11     ;Aux. Ctrl-Register
27   PCR      EQU  DEV+12     ;Periph. Ctrl-Reg.
28   IFR      EQU  DEV+13     ;Interrupt Flag-Reg.
29   *
3Ø   * Scratchpad-RAM des Slot
31   *
32   RAML     EQU  $Ø478+SLOT ;Anfangsadr. RAM
33   RAMH     EQU  $Ø4F8+SLOT
34   ROML     EQU  $Ø578+SLOT ;Anfangsadr. EPROM
35   ROMH     EQU  $Ø5F8+SLOT
36   ENDL     EQU  $Ø678+SLOT ;Endadresse EPROM
37   ENDH     EQU  $Ø6F8+SLOT
38   ROMTYPE  EQU  $Ø778+SLOT ;Typ: $8Ø = 2732
39                            ;      $ØØ = 2716
4Ø   *
41   * Zero-Page-Variablen
42   *
43   RAMPL    EQU  $3C        ;RAM-Zeiger
44   RAMPH    EQU  $3D
45   ROMPL    EQU  $3E        ;EPROM-Zeiger
46   ROMPH    EQU  $3F
47   EL       EQU  $1E        ;EPROM-Adr.-Byte
48   EH       EQU  $1F
49   *
5Ø   * Startadresse entspr. Slot-Nummer
51   *
52            ORG  $1ØØ*SLOT+$CØØØ
53   *
54   * EPROM lesen
55   *
C4ØØ: 2Ø 9D C4 56   LESEN    JSR  RINIT      ;Anfangswerte setzen
C4Ø3: 2Ø CD C4 57   LNEXT    JSR  LIES       ;lies EPROM-Byte
C4Ø6: 91 3C    58            STA  (RAMPL),Y  ;speichere ab
C4Ø8: 2Ø E8 C4 59            JSR  RINC       ;Adressen weiter
C4ØB: CD FC Ø6 6Ø            CMP  ENDH       ;Endadr. erreicht?
C4ØE: DØ F3    61            BNE  LNEXT
C41Ø: EC 7C Ø6 62            CPX  ENDL
C413: DØ EE    63            BNE  LNEXT
C415: 6Ø       64            RTS
               65   *
               66   * EPROM-Leertest
               67   *
C416: 2Ø 9D C4 68   TESTEN   JSR  RINIT      ;Anfangswerte setzen
C419: 2Ø CD C4 69   TNEXT    JSR  LIES       ;lies EPROM-Byte
C41C: C9 FF    7Ø            CMP  #$FF       ;Byte gelöscht?
C41E: DØ ØE    71            BNE  FEHLER
C42Ø: 2Ø F5 C4 72            JSR  EINC       ;Adressen weiter
C423: CD FC Ø6 73            CMP  ENDH       ;Endadr. erreicht?
C426: DØ F1    74            BNE  TNEXT
C428: EC 7C Ø6 75            CPX  ENDL
C42B: DØ EC    76            BNE  TNEXT
C42D: 6Ø       77            RTS
C42E: 4C 2D D4 78   FEHLER   JMP  ERROR      ;Fehlermeldung
               79   *
               8Ø   * EPROM-Programmieren
               81   *
C431: 2Ø 9D C4 82   PROG     JSR  RINIT      ;Anfangswerte setzen
C434: 8C C4 CØ 83            STY  TIL        ;Timer-Low-Reg = Ø
C437: A9 8Ø    84            LDA  #$8Ø       ;Bit 7 des B-Ports
C439: 8D CB CØ 85            STA  ACR        ;ist Timer-Output
C43C: A9 FF    86            LDA  #$FF       ;Bit Ø-7 des A-Ports
C43E: 8D C3 CØ 87            STA  DDRA       ;ist Output
C441: AD FC Ø5 88            LDA  ROMH
C444: AE 7C Ø5 89            LDX  ROML       ;Zeiger setzen
C447: 2Ø B7 C4 9Ø   PNEXT    JSR  EADR       ;EPROM-Adr. bilden
C44A: A9 AC    91            LDA  #$AC       ;CB2-Latch,
C44C: 8D CC CØ 92            STA  PCR        ;Prog.-Spannung an
C44F: A5 1F    93            LDA  EH         ;EPROM-Adr., High
C451: 8D CØ CØ 94            STA  PORTB      ;abspeichern
C454: A9 EC    95            LDA  #$EC       ;CB2-Latch aus
C456: 8D CC CØ 96            STA  PCR        ;Prog.-Sp. weiter an
C459: A5 1E    97            LDA  EL         ;EPROM-Adr., Low
C45B: 8D CØ CØ 98            STA  PORTB      ;abspeichern
C45E: B1 3C    99            LDA  (RAMPL),Y  ;Daten-Byte
C46Ø: 8D C1 CØ 1ØØ           STA  PORTA      ;an Port A
C463: A9 C3    1Ø1           LDA  #195       ;5Ø ms Zeitintervall
C465: 8D C5 CØ 1Ø2           STA  TIH        ;starten
C468: AD CD CØ 1Ø3   WARTE   LDA  IFR        ;Flag-Register
C46B: ØA       1Ø4           ASLA            ;Uhr-Interrupt?
C46C: 1Ø FA    1Ø5           BPL  WARTE
C46E: 2Ø E8 C4 1Ø6           JSR  RINC       ;Adressen weiter
C471: CD FC Ø6 1Ø7           CMP  ENDH       ;Endadr. erreicht?
C474: DØ D1    1Ø8           BNE  PNEXT
C476: EC 7C Ø6 1Ø9           CPX  ENDL
C479: DØ CC    11Ø           BNE  PNEXT
C47B: A9 EE    111           LDA  #$EE       ;Prog.-Spannung
C47D: 8D CC CØ 112           STA  PCR        ;abschalten
C48Ø: A9 ØØ    113           LDA  #Ø         ;Timer ausschalten
C482: 8D CB CØ 114           STA  ACR
               115  *
               116  * RAM und EPROM vergleichen
               117  *
C485: 2Ø 9D C4 118  VERGLEI  JSR  RINIT      ;Anfangswerte setzen
C488: 2Ø CD C4 119  VNEXT    JSR  LIES       ;lies EPROM-Byte
```

```
C48B: D1 3C     120           CMP  (RAMPL),Y  ;vergleiche mit RAM
C48D: D0 9F     121           BNE  FEHLER
C48F: 20 E8 C4  122           JSR  RINC       ;Adressen weiter
C492: CD FC 06  123           CMP  ENDH       ;Endadr. erreicht?
C495: D0 F1     124           BNE  VNEXT
C497: EC 7C 06  125           CPX  ENDL
C49A: D0 EC     126           BNE  VNEXT
C49C: 60        127           RTS
                128 *
                129 * Subroutinen
                130 *
                131 * Anfangswerte setzen
                132 *
C49D: AD 7C 04  133 RINIT     LDA  RAML       ;Zeiger setzen
C4A0: 85 3C     134           STA  RAMPL
C4A2: AD FC 04  135           LDA  RAMH
C4A5: 85 3D     136           STA  RAMPH
C4A7: A0 FF     137           LDY  #$FF       ;Bit 0-7 des B-Ports
C4A9: 8C C2 C0  138           STY  DDRB       ;ist Output
C4AC: C8        139           INY             ;Bit 0-7 des A-Ports
C4AD: 8C C3 C0  140           STY  DDRA       ;ist Input
C4B0: AD FC 05  141           LDA  ROMH
C4B3: AE 7C 05  142           LDX  ROML       ;Zeiger setzen
C4B6: 60        143           RTS
                144 *
                145 * EPROM-Adresse einstellen
                146 *
C4B7: 86 3E     147 EADR      STX  ROMPL      ;Zeiger setzen
C4B9: 86 1E     148           STX  EL         ;EPROM-Adr., Low
C4BB: 06 1E     149           ASL  EL         ;anpassen
C4BD: 85 3F     150           STA  ROMPH
C4BF: 85 1F     151           STA  EH
C4C1: 26 1F     152           ROL  EH
C4C3: 18        153           CLC             ;PB7 = 0 für 2732
C4C4: 2C 7C 07  154           BIT  ROMTYPE    ;EPROM-Typ?
C4C7: 30 01     155           BMI  E2732
C4C9: 38        156           SEC
C4CA: 66 1E     157 E2732     ROR  EL
C4CC: 60        158           RTS
                159 *
                160 * EPROM-Byte lesen
                161 *
C4CD: 20 B7 C4  162 LIES      JSR  EADR       ;EPROM-Adr. einst.
C4D0: A9 AE     163           LDA  #$AE       ;CB2-Latch ein
C4D2: 8D CC C0  164           STA  PCR        ;Prog.-Spannung aus
C4D5: A5 1F     165           LDA  EH         ;EPROM-Adr., High
C4D7: 8D C0 C0  166           STA  PORTB      ;abspeichern
C4DA: A9 EE     167           LDA  #$EE       ;CB2-Latch aus
C4DC: 8D CC C0  168           STA  PCR        ;Prog.-Spannung aus
C4DF: A5 1E     169           LDA  EL         ;EPROM-Adr., Low
C4E1: 8D C0 C0  170           STA  PORTB
C4E4: AD C1 C0  171           LDA  PORTA      ;lies Daten-Byte
C4E7: 60        172           RTS
                173 *
                174 * Adressen inkrementieren
                175 *
C4E8: 18        176 RINC      CLC             ;Carry löschen
C4E9: A5 3C     177           LDA  RAMPL      ;RAM-Zeiger weiter
C4EB: 69 01     178           ADC  #1
C4ED: 85 3C     179           STA  RAMPL
C4EF: A5 3D     180           LDA  RAMPH
C4F1: 69 00     181           ADC  #0
C4F3: 85 3D     182           STA  RAMPH
C4F5: 18        183 EINC      CLC
C4F6: A5 3E     184           LDA  ROMPL      ;EPROM-Zeiger weiter
C4F8: 69 01     185           ADC  #1
C4FA: AA        186           TAX
C4FB: A5 3F     187           LDA  ROMPH
C4FD: 69 00     188           ADC  #0
C4FF: 60        189           RTS

256 Bytes
```

Telefonische Bestellungen?

Da unsere Peeker-Disketten in offener Rechnung und nicht in dem für Sie teuren Nachnahme-Verfahren ausgeliefert werden, haben Sie bitte Verständnis dafür, daß wir **nur noch schriftliche Bestellungen annehmen.**

Sie können dazu beispielsweise die in jedem Peeker eingehefteten Bestellkarten verwenden.

Hüthig Software Service

Pascal-Kompaktkurs für Applesoft-Programmierer

UCSD-Apple-Pascal (1.1/1.2) und Turbo-Pascal (3.0)

Die Peeker-Sammeldisk #10 (DOS-3.3-Format) enthält zwei sachlich gleiche, aber auf die jeweilige Programmiersprache angepaßte Pascal-Kompaktkurse für Apple-Pascal (1.1/1.2) und Turbo-Pascal (3.0 oder älter). Es werden mehr als die Hälfte aller Befehle behandelt und in Form von 9 Bildschirmübersichten vorgestellt (Bildschirm-Ausgabe, Formatierung der Ausgabe, Wertzuweisung, Tastatur-Eingabe, Integer-Mathematik, Fließkomma-Mathematik, Verzweigungen und Schleifen, String-Verarbeitung, Hauptmenü). Im Apple-Pascal-Quelltext wird jedem einzelnen Pascal-Befehl der entsprechende Applesoft-Befehl zum Vergleich als Kommentarzeile vorangestellt (bei der Turbo-Pascal-Version aus Platzgründen entfallen).

UCSD.TEXT

- UCSD.TEXT von DOS-3.3-Sammeldisk #10 mit GETDOS auf Pascal-Diskette konvertieren, d.h.
- Pascal von Drive 1 booten und Sammeldisk #10 in Drive 2 einlegen
- Peeker-Sammeldisk #9 in Drive 1 einlegen und mit E<xecute #4:GETDOS Konvertierungsprogramm starten. Danach eigene Pascal-Systemdiskette wieder in Drive 1 einlegen.
- 5 für Quelldrive 2
- 4 für Zieldrive 1 und
- UCSD.TEXT als Name eingeben
- Nach Konvertierung E(ditor aufrufen und Name UCSD eingeben, danach Editor mit U(pdate verlassen, dann mit C(ompile compilieren und schließlich mit R(un starten.

TURB.PAS

- TURB.PAS von DOS-3.3-Sammeldisk #10 mit APDOS auf CP/M-Turbo-Pascal-Diskette konvertieren, d.h.
- CP/M-Diskette von Drive 1 booten und Sammeldisk #10 in Drive 2 einlegen.
- APDOS eingeben
- TURB.PAS=TURB.PAS eingeben
- Nach erfolgter Konvertierung mit Ctrl-C zurück zu CP/M
- Jetzt Turbo-Pascal mit TURBO starten. Auf "Error-Messages" N(ein antworten, dann E(dit und dann TURB eingeben. Nun mit C(ompile compilieren und schließlich mit R(un starten.

Der Quelltext des Kompaktkurses wird im nächsten Peeker abgedruckt. Das Durchspielen des Kurses setzt jedoch keinerlei Kenntnisse voraus. Sie brauchen lediglich die gewünschte Übung aus dem Menü auszuwählen. Sehen Sie sich danach im Editor die entsprechende Passage des Quelltextes an, und ändern und ergänzen Sie nach Belieben. Wenn Sie bislang noch niemals in Pascal programmiert haben, so lassen Sie sich beim Konvertieren von jemandem helfen.

U. Stiehl

Rechtsprechung zum Software-Urheberrecht

von RA Dieter Naber

Nach anfänglichen Zweifeln herrscht in der deutschen Rechtsprechung heute Übereinstimmung, daß Computer-Software Urheberrechtsschutz genießen kann, sofern sie nach den Kriterien des Urhebergesetzes als „Werk" zu qualifizieren ist. Im Einzelfall bereitet die Feststellung des Werkcharakters jedoch häufig Probleme, auch Art und Umfang des gewährten Schutzes sind noch nicht vollständig geklärt.

Zunächst eine Definition: **„Software"** wird als Oberbegriff für das Programm selbst, das Begleitmaterial (Handbücher usw.) und die Programmdokumentation (Datenflußplan usw.) verwendet. Einzelne Bestandteile der Software können urheberrechtlich durchaus unterschiedlich beurteilt werden. Ein **Anwenderhandbuch** kann z. B. – wie jedes andere Buch – als Schriftwerk urheberrechtlich geschützt sein, ohne daß es dabei auf das darin beschriebene Programm ankäme. Im folgenden soll jedoch ausschließlich vom spezifischen Schutz der Computer-Programme die Rede sein.

Auch das **Programm** selbst wird meist der Kategorie der Schriftwerke zugeordnet, da es vom Programmierer unter Verwendung einer Sprache erdacht und geschaffen wurde. Auf die Allgemeinverständlichkeit der Sprache kommt es hier wie auch sonst im Urheberrecht nicht an; die Verwendung mathematischer Zeichen und Symbole schadet nicht. Wo es an jeglichen Sprachsymbolen fehlen sollte, steht noch die Kategorie der Darstellung wissenschaftlicher oder technischer Art zur Verfügung.

Mit der Zuordnung zu den vom Urhebergesetz vorgegebenen Kategorien ist allein noch nichts gewonnen. Um schutzfähig zu sein, muß sich das Programm als „Werk" darstellen, also als persönliche geistige Schöpfung. Wie und woran läßt sich der Werkcharakter erkennen?

Der **Objektcode** ist zwar für Maschinen, doch nicht für Menschen lesbar, so daß er als unmittelbare Grundlage der Prüfung ausscheidet. Lesbar ist dagegen der **Quellcode** und die begleitende Dokumentation, die zumindest bei Programmen mittleren und größeren Umfangs regelmäßig wesentlicher Bestandteil der Programmentwicklung ist. An Datenflußplan, Programmablaufplan und Quellcode zeigt sich für den Sachkundigen, wie das Programm „gewebt" ist. Auf dieses Webmuster kommt es an.

Nachdem kürzlich der Bundesgerichtshof, das höchste deutsche Zivilgericht, über die Anforderungen an den Werkcharakter eines Computer-Programms zu entscheiden hatte, ist durch weitgehende Bestätigung der bisher von unteren Gerichten vertretenen Auffassung eine gewisse Klärung eingetreten. Nach nunmehr herrschender Rechtsprechung kommt es entscheidend auf die Gestaltung des Programms an, wie sie sich in Auswahl, Sammlung, Anordnung und Einteilung des gesamten Stoffes, der Informationen und Anweisungen zeigt. Nicht ausreichend ist eine rein handwerksmäßige Zusammenstellung und sich auf eine mechanisch-technische Aneinanderreihung beschränkende Zusammenfügung des Materials (1). In gleicher Weise hatten bereits verschiedene Land- und Oberlandesgerichte in den vergangenen Jahren geurteilt, so daß man insoweit heute von einer gefestigten Rechtsprechung ausgehen kann.

Bereits 1981 war die Schutzfähigkeit von Computer-Programmen von mehreren Gerichten anerkannt worden (2). Lediglich das LG Mannheim (3) vermochte keine „greifbare, einer geistig-ästhetischen Wertung zugängliche Formgestaltung" zu erkennen und hielt deshalb Computer-Programme für eine „abstrakte Kombination von Gedankenschritten, die als solche der sinnlichen Wahrnehmung entzogen" seien. Diese zu sehr auf ästhetische Gesichtspunkte abstellende Wertung wurde aber bereits vom Berufungsgericht verworfen (4) und schließlich auch vom BGH in dem o.g. Urteil nicht bestätigt.

Die Gerichte mußten sich mit nahezu jeder Art von Programmen befassen. Eine wichtige Rolle spielten Branchenlösungen. Programmen über Baustatik (5), Verschnittoptimierung (6) und Inkasso (7) wurde aufgrund der oben aufgezeigten Kriterien bereits der Werkcharakter und damit die Schutzfähigkeit zuerkannt.

Als Standardsoftware machte **Visicalc**, der Vater der Tabellenkalkulationsprogramme, von sich reden, als das LG München (8) ihm im Dezember 1982 urheberrechtlichen Schutz gewährte, ohne sich mit der Begründung allzu schwer zu tun: Jedes nicht ganz triviale Programm weise jedenfalls eine individuelle Prägung auf, die mit der Länge des Programms wachse. Die neben der bloßen Individualität erforderliche geistige Leistung leitete das Gericht kurzerhand daraus ab, „daß die hohe Qualität und Originalität des Programms in zahlreichen Presseartikeln bestätigt wird".

Auch wenn das Urteil im Ergebnis wahrscheinlich zutreffend war, könnte die Begründung den heute vom BGH und der überwiegenden Zahl der Oberlandesgerichte gestellten Anforderungen nicht mehr genügen. Die Länge eines Programms oder die Anzahl der Anweisungen ist allein allenfalls ein schwaches Indiz, ebensowenig begründet der geistige Aufwand und die Mühe der Programmierung bereits die urheberrechtliche Schutzfähigkeit. Wesentlich ist, daß die Aufgabenstellung bei der Programmierung einen weiten Spielraum gewährt, also eine willkürliche Lösung zuläßt. Nur wenn eine solche **individuelle Lösung** zudem die eingangs beschriebenen Merkmale einer besonderen

Gestaltungsfähigkeit aufweist, kann eine schöpferische Leistung angenommen werden. Dies ist sicher nicht mit einem Blick in die einschlägige Presse festzustellen, dazu bedarf es einer eingehenden Untersuchung des Programms selbst.

Die Notwendigkeit dieser Untersuchung hat zur Folge, daß der auf sein Urheberrecht pochende Kläger in einem Rechtsstreit gezwungen ist, sein Programm nebst der zugehörigen **Dokumentation offenzulegen**. Dies kann für ihn unerwünschte Folgen haben, denn möglicherweise erhält der verklagte Konkurrent erst auf diesem Wege ihm bisher unbekannte Detailinformationen. Das OLG Nürnberg erkannte das mögliche Dilemma der klagenden Partei und gewährte ihr die Möglichkeit, Programm und Dokumentation ausschließlich einem gerichtlich bestellten und vereidigten Sachverständigen, *nicht aber der gegnerischen Partei* auszuhändigen (9). Andererseits ist nicht zu verkennen, daß dadurch die Möglichkeit des Beklagten beschnitten wird, Feststellungen des Sachverständigen anhand eigener Untersuchungen zu überprüfen und ggf. anzugreifen. Unter dem Gesichtspunkt der Chancengleichheit sicher ein nicht leicht zu lösendes Problem.

Eine eigene Gruppe, auch im Spiegel der Rechtsprechung, bilden die **Spielprogramme**. Es besteht Einigkeit, daß die Spielidee als solche urheberrechtlich nicht schutzfähig ist, sei sie noch so genial oder neu. Das OLG Frankfurt stützte seine Beurteilung des „Donkey Kong Junior" auf Sachverständigengutachten, in denen es die Darlegung einer schöpferischen Leistung allerdings vermißte. Es konnte nicht feststellen, „daß etwas Besonderes, aus der Masse der alltäglichen Gebilde der auf dem Markt erscheinenden Video-Spiele Herausragendes geschaffen worden wäre" (10).

Auch gegenüber „Pengo" mußte das OLG Frankfurt hart bleiben (11), denn die Antragstellerin hatte keinerlei Material vorgelegt, anhand dessen das Gericht die urheberrechtliche Schutzfähigkeit hätte feststellen können. Es mußte sich daher auf die Betrachtung des äußeren Spielablaufs beschränken und fand, „da es sich um eines von vielen auf dem Markt existierenden Reaktions- und Geschicklichkeitsspielen handelt, ist eine urheberrechtsbegründende Auswahl des Stoffes nicht erkennbar".

Nicht anders erging es der Firma **Atari** mit ihren Spielkassetten vor dem OLG Karlsruhe (12), denn auch sie hatte nichts zur Begründung des behaupteten Urheberrechts vorgetragen. Der Grund mag darin gelegen haben, daß sich in diesem konkreten Fall (13) die Prozeßparteien über den Urheberschutz einig waren. Die Gerichte sind an solche Übereinstimmung jedoch nicht gebunden, denn es handelt sich bei der urheberrechtlichen Schutzfähigkeit um eine rechtliche Schlußfolgerung, die der Disposition der Parteien entzogen und gerichtlich eigenständig zu prüfen ist (14).

Bei den stark **grafikorientierten Spielen** stellt sich die Frage, ob sie als „Filmwerke" oder „Werke, die ähnlich wie Filmwerke geschaffen werden" (§ 2 Abs. 1 Nr. 6 UrhG), geschützt sein können. Das OLG Hamburg bejahte dies grundsätzlich, obgleich in dem ihm vorliegenden Fall „PUCKMAN gegen Supermampfer" (15) eine konkrete Verletzung des Urheberrechts nicht nachzuweisen war. Das OLG Frankfurt zeigt sich zurückhaltender: Im Gegensatz zu einem Filmwerk liege das sichtbare Ergebnis des Spiels nicht fest, sondern weise in Reaktion auf die Eingriffe des Spielers unterschiedliche Varianten auf (16). Außerdem komme als „Filmwerk" nur die Umsetzung des Progamms in eine bewegte Bildfolge in Betracht (ein Drehbuch = Quellcode allein macht noch keinen Film), und dieser Umsetzungsvorgang lasse keinen schöpferischen Gestaltungsspielraum mehr zu, sondern laufe weitgehend automatisch ab. Der Bildschirm gebe keinen Film wieder, sondern stelle lediglich das rechnerische Ergebnis des jeweiligen Programmabschnittes dar, mit der einzigen Besonderheit, daß die Darstellung nicht numerisch oder alphanumerisch, sondern auf grafische Art erfolgt (17).

Vom Video zurück zum reinen Programm: In dem kürzlichen Urteil des BGH taucht ein hier noch nicht erwähntes Kriterium auf, das zu künftigen Diskussionen Anlaß geben wird. Der BGH verlangt, daß die besondere, den Urheberschutz begründende **Gestaltung des Programms** über das Können eines durchschnittlichen Programmierers hinausgehen müsse! Diese Entwicklung wird man kritisch verfolgen müssen, denn es wäre bedenklich, die vom UrhG verlangte schöpferische Leistung an einem unbestimmten Qualitätsmaßstab messen zu wollen. Weder läßt sich in der Praxis ein abstraktes „Durchschnittskönnen" ermitteln, noch ist es Aufgabe des Urheberrechts, ein bestimmtes Qualitätsniveau zu erhalten. Das LG Mosbach hat dies zutreffend ausgedrückt: „Zur Versagung des Schutzes reicht nicht aus, daß auch ein durchschnittlicher Programmierer das zu beurteilende Programm erarbeiten kann, denn niemand denkt daran, einem Schriftsteller oder Musiker den Urheberschutz zu versagen, wenn er nur durchschnittlich ist" (18).

Ebenfalls noch nicht abschließend geklärt ist das Problem des Schutzumfangs, das sich an folgendem Beispiel aufzeigen läßt: Ein Softwarehaus schafft im Auftrage eines Kunden ein spezifisches Branchenprogramm. Die urheberrechtlichen Nutzungs- und Verwertungsrechte werden vereinbarungsgemäß ausschließlich dem Kunden eingeräumt. Einige Zeit später erhält das Softwarehaus einen ganz ähnlichen Auftrag von einem anderen Kunden. Es wäre lebensfremd, wollte man verlangen, daß der Hersteller sein aus dem ersten Auftrag gewonnenes Know-how vergißt und das Rad zum zweiten Mal erfindet. Auch wenn er nicht die Absicht hat, einfach abzuschreiben, wird er bei der Problemlösung auf frühere Erfahrungen zurückgreifen.
Wenn nun der urheberrechtliche Schutz nicht nur die äußere Form, sondern – abgesehen vom freien Algorithmus – auch die inhaltliche Funktions- und Arbeitsweise des Programms erfaßt, läuft der Softwareproduzent Gefahr, sich selbst zu plagiieren (19).

Aus den eingangs genannten, von der Rechtsprechung an den Werkcharakter eines Programms gestellten Anforderungen wird zwar erkennbar, daß das Gewicht auf der Formgestaltung liegt. Doch gibt es grundsätzlich keine zweckfreie, vom Inhalt völlig gelöste Form; vielmehr wird die Form der Gedankenführung zwangsläufig vom Inhalt der Mitteilung beeinflußt (20). So finden sich denn in einzelnen Urteilen durchaus Hinweise, daß neben der Formgestaltung auch inhaltliche Elemente des Progamms den Werkcharakter bestimmen und damit ebenfalls dem Urheberschutz zugänglich sein können (21).

Der Unterschied zwischen Form- und Inhaltsschutz hat vielfältige Aspekte. Die aus einem Inhaltsschutz drohende Gefahr der Selbstkollision wurde bereits genannt. Auch wird es ungleich schwieriger sein, eine inhaltliche geistige Leistung festzustellen: Wenn schon die Grenzen zwischen **Form und Inhalt** fließend sind, so sind sie zwischen Inhalt und Algorithmus erst recht unsicher (19). Andererseits ist der Umfang des Formschutzes geringer, da sich die äußere Gestaltung eines Programms meist mit geringerem Aufwand ändern bzw. neu schaffen läßt als die inhaltliche Funktionsweise. Der Formschutz wirkt daher am ehesten gegen echte 1:1 Kopien oder Teilübernahmen.

Wie man sieht, hat der Urheberschutz für Computer-Programme heute in der Recht-

sprechung zwar Anerkennung gefunden, die juristische Diskussion im einzelnen ist aber noch lange nicht beendet. Noch ein Hinweis zum Schluß: Das Urheberrecht ist nur ein Teilbereich des gewerblichen Rechtsschutzes. Auch wenn einem Programm im Einzelfall kein Urheberschutz gewährt werden kann, ist es noch lange nicht vogelfrei; insbesondere gegen Raubkopien schützen auch das Gesetz gegen den unlauteren Wettbewerb und ggf. auch das Warenzeichengesetz!

Literatur

Die verwendeten Abkürzungen bedeuten:
LG – Landgericht;
OLG – Oberlandesgericht;
BGH – Bundesgerichtshof;
LAG – Landesarbeitsgericht;
BAG – Bundesarbeitsgericht.
Die nachstehenden Fundstellen beziehen sich auf:
BB – „Betriebsberater" und
GRUR – „Gewerblicher Rechtsschutz und Urheberrecht", jeweils mit Jahr und Seite.

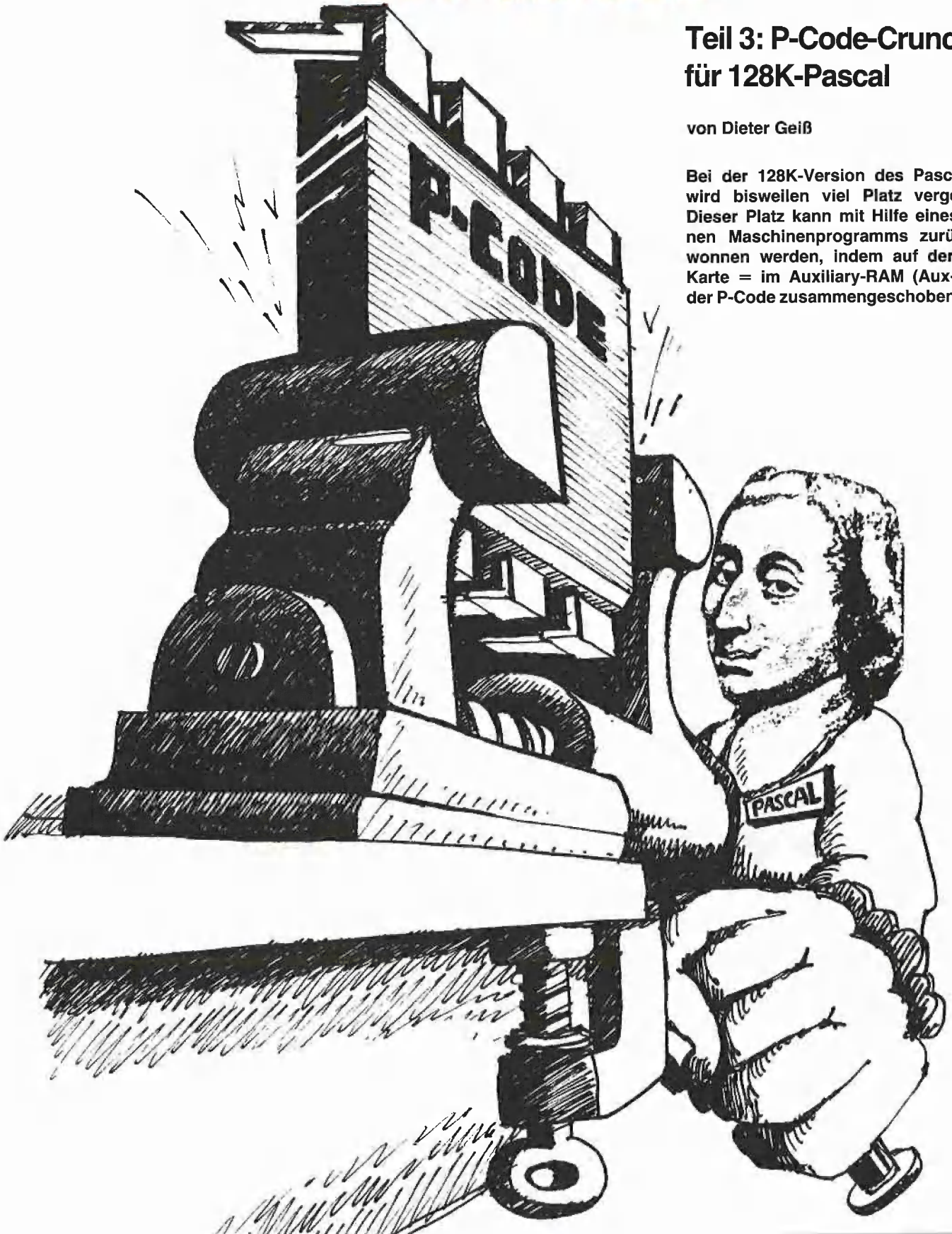
(1) BGH Urteil v. 9.5.1985, im vollen Wortlaut bei Drucklegung noch nicht veröffentlicht.
(2) LG Kassel BB 83, 992; OLG Koblenz BB 83, 992; LAG Schleswig BB 83, 994.
(3) LG Mannheim BB 81, 1543 (Inkassoprogramm).
(4) OLG Karlsruhe GRUR 83, 300 (Inkassoprogramm).
(5) LG Kassel BB 83, 992; LG Mosbach GRUR 83, 70; BAG GRUR 84, 429.
(6) OLG Nürnberg GRUR 84, 736.
(7) OLG Karlsruhe GRUR 83, 300.
(8) LG München GRUR 83, 175.
(9) OLG Nürnberg GRUR 84, 736 (Verschnittoptimierung).
(10) OLG Frankfurt GRUR 83, 757 und GRUR 84, 509.
(11) OLG Frankfurt GRUR 83, 753.
(12) OLG Karlsruhe GRUR 84, 521.
(13) Ähnlich auch im „Inkassoprogramm-Fall" des LG Mannheim/OLG Karlsruhe /BGH.
(14) OLG Karlsruhe GRUR 84, 521 und GRUR 83, 300.
(15) OLG Hamburg GRUR 83, 436. PUCKMAN ist die Spielhallenversion von PacMan.
(16) OLG Frankfurt GRUR 83, 757 (Donkey Kong). Gerade diesem Argument maß das OLG Hamburg keine Bedeutung bei.
(17) OLG Frankfurt GRUR 83, 753 (Pengo).
(18) LG Mosbach GRUR 83, 70 (Baustatik).
(19) Brandi-Dohrn, Zur Reichweite und Durchsetzung des urheberrechtlichen Softwareschutzes, GRUR 85, 179.
(20) OLG Frankfurt GRUR 83, 753 (Pengo); OLG Karlsruhe GRUR 83, 300 (Inkaso).
(21) LG München GRUR 83, 175 (Visicalc); OLG Koblenz BB 83, 992 (Statik); OLG Hamburg GRUR 83, 426 (PUCKMAN); OLG Nürnberg GRUR 84, 736 (Verschnitt).

Tips und Tricks in Pascal

Teil 3: P-Code-Cruncher für 128K-Pascal

von Dieter Geiß

Bei der 128K-Version des Pascal 1.2 wird bisweilen viel Platz vergeudet. Dieser Platz kann mit Hilfe eines kleinen Maschinenprogramms zurückgewonnen werden, indem auf der 64K-Karte = im Auxiliary-RAM (Aux-RAM) der P-Code zusammengeschoben wird.



Doch zuerst zur Theorie des „Cruncher“. Wer sich einmal nähere Gedanken darüber gemacht hat, wie das 128K-Pascal für den Apple IIc bzw. den Apple IIe mit erweiterter 64K-Karte arbeitet, dem mögen einige Dinge aufgefallen sein.

1. Die Theorie des „Crunchens“

Wie schon im Peeker 3/85, S. 68 erwähnt, werden die oberen 64K (Aux-RAM) nur vom P-Code belegt. Ein Segment wie z.B. das Hauptprogramm, eine Segment-Prozedur oder eine Unit kann aber auch Maschinenprozeduren enthalten. Nun wird beim Laden aber das gesamte Segment auf einmal von der Diskette geholt und, nachdem es zuerst im Main-RAM kurzzeitig Platz gefunden hat, ins Aux-RAM geschoben.
Maschinenprozeduren müssen aber im Main-RAM ablaufen, da sie auf globale (PUBLIC) und lokale (PRIVATE) Variablen zugreifen könnten, die sich ja im Main-RAM befinden. Deshalb verfährt das Pascal-System wie folgt:
Lade das Segment und überprüfe, ob es Assemblerprozeduren enthält. Wenn ja, dann suche die tiefste und höchste Maschinenprozedur im Code und verschiebe alles, was dazwischen liegt, also sämtliche Maschinenprozeduren, ins Main-RAM. Da der P-Code verschiebbar ist, muß noch der Zeiger auf den ersten auszuführenden Befehl (Enter-IC, s.a. „Apple Pascal Operating System Reference Manual“: Operation of the P-Machine) korrigiert werden. Dieser zeigt dann auf die Stelle im Main-RAM, wo der Code nach dem Verschieben liegt.
Bild 1 zeigt ein Beispiel für die Speicherbelegung, nachdem ein Programm mit zwei Maschinenprozeduren (Prozeduren #2 und #4) geladen wurde. Der Assemblercode auf dem Aux-RAM ist nicht mehr zugänglich, da der zugehörige Enter-IC ins Main-RAM zeigt. Die beiden halbfetten Felder im Aux-RAM sind identisch mit den halbfetten Feldern im Main-RAM, d.h. der Assemblercode im Aux-RAM ist überflüssig und könnte entfernt werden. Speicherplatz könnte dann gespart werden, wenn der untere Teil des Segments, also der P-Code, nach oben geschoben werden würde (Crunching). Genau dies ist die Aufgabe des Crunchers. Außer dem eigentlichen Zusammenschieben müssen aber noch sämtliche Einträge im Procedure-Dictionary und die Enter-IC-Einträge der Assemblerprozeduren korrigiert werden.

Im einzelnen arbeitet der Cruncher nach folgendem Algorithmus:
1 Markiere alle Maschinenprozeduren im gerade geladenen Segment.

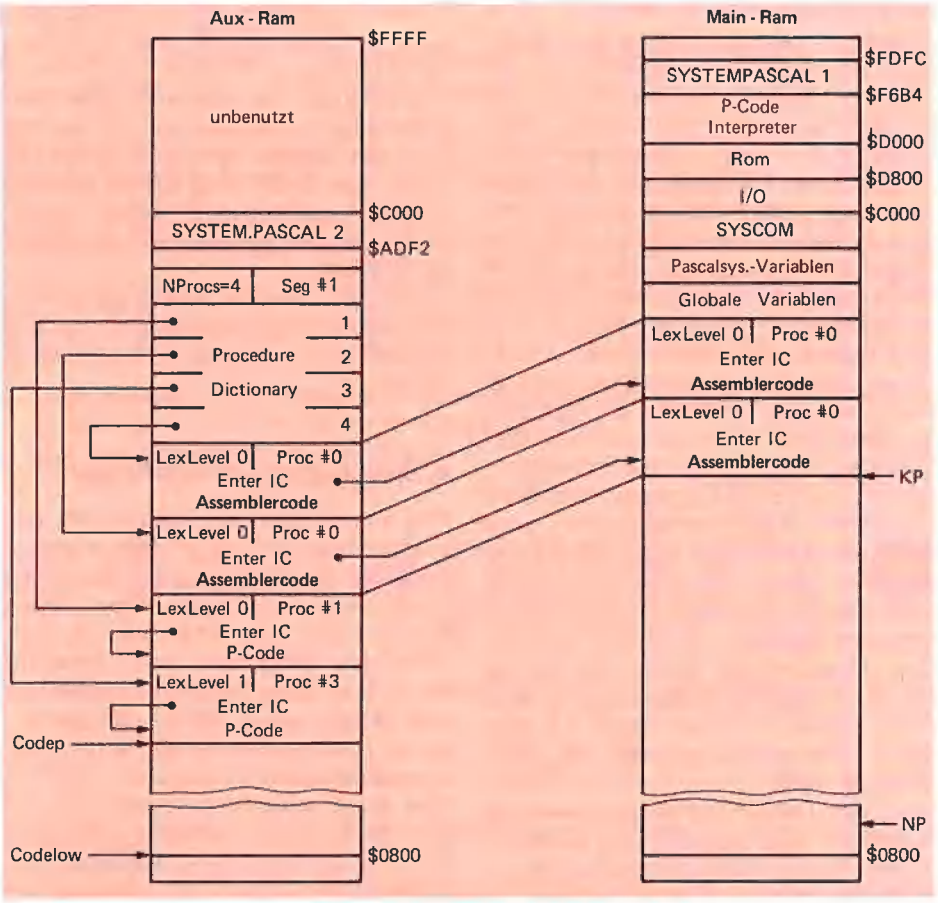


Bild 1: Cruncher von zwei Maschinenprozeduren (IC = Interpreter Counter)

2 Solange noch Maschinenprozeduren markiert sind, berechne jeweils die Maschinenprozedur mit der höchsten Adresse, sonst mache weiter bei 7.
3 Verschiebe den rudimentären Teil, der noch gebraucht wird, also lexikalische Verschachtelungstiefe (bei Maschinenprozeduren „Relocatable Segment Number“), Prozedurnummer und Enter-IC.
4 Korrigiere den Eintrag im Procedure-Dictionary.
5 Korrigiere den Wert des Enter-IC.
6 Mache weiter bei 2.
7 Verschiebe P-Code.
8 Korrigiere alle Einträge für P-Code-Prozeduren im Procedure-Dictionary.
9 Kehre zurück zum System.

2. Der Patch

Damit der Cruncher an das System angeschlossen werden kann, muß eine Stelle gepatcht werden, und zwar die Stelle, an der ein Segment gerade geladen wurde und eventuelle Maschinenprozeduren ins Main-RAM geschoben wurden. Statt des Umschaltens auf Lesen des Main-RAMs wird ein JSR $0800 erzwungen. Dort liegt die Anfangsadresse des Crunchers im Aux-RAM. Ist schon ein anderer Treiber oder etwas Ähnliches in das Aux-RAM geschoben worden, ändert sich auch der JSR-Befehl. Auf jeden Fall zeigt der Befehl auf die Anfangsadresse des Crunchers.
Da mir das Patchen von Systemfiles wie SYSTEM.APPLE auf Diskette nicht so sehr liegt, weil man vielleicht dieses System auf einem anderen Computer ohne den fehlenden File (CRUNCHER.CODE) laufen läßt und dann nichts mehr funktioniert, habe ich dieses Problem mit einer Art Auto-Patch gelöst.

3. Die Implementierung

3.1. Theoretische Hinweise

Das gesamte Cruncher-Paket besteht aus drei Files. Der „StartCruncher“, auf der Peeker-Sammeldisk **CRUNCH.CODE** genannt – sucht den eigentlichen „Cruncher“ – auf Sammeldisk **CRUNCHER .CODE** genannt – und lädt ihn. Dann ruft es die Maschinenprozedur „MoveCruncher“ auf – auf Sammeldisk **MOVER.CODE** genannt. Diese verschiebt den Cruncher von der Ladeadresse zur eigentlichen Arbeitsadresse, z.B. Adresse $0800 im Aux-RAM. Da der Cruncher wie jeder andere TLA-Assemblerfile, der ohne „.ABSOLUTE“ übersetzt wurde, verschiebbar ist, muß der MoveCruncher noch eine

Adressentranslation durchführen. Diese Translation geschieht ähnlich der Translation beim Laden eines Segments. Es werden nur Enter-IC-Translationen berücksichtigt, also nicht BASE-relativ-, Segment-relativ- oder Interpreter-relativ-Translationen. Dies genügt für den Treiber durchaus, da dort keine PUBLICs, PRIVATEs, REFs und DEFs auftreten. Außer der Translation wird dabei auch das System im RAM gepatcht. Dabei tritt noch ein kleines Problem auf:
Es gibt zwei verschiedene Pascal-1.2-Versionen des P-Code-Interpreters, zum einen die reguläre Version, die man zum Arbeiten benutzt, zum anderen eine spezielle Run-Time-Version, die Softwareherstellern zur Verfügung gestellt wird, damit Programme erstellt werden können, die ja ein System benötigen, damit sie ablaufen können. Diese Run-Time-Version darf dann von den Softwareherstellern weitergegeben werden.
Der MoveCruncher patcht die richtige Stelle im regulären System und im Run-Time-System. Wird kein Pascal 1.2 oder keine 128K-Version gefunden, so unterbleibt der Patch.
Nach dem Patchen und dem Kopieren des Cruncher wird schließlich der Codelow-Pointer hochgesetzt, um den Cruncher vor dem Überschreiben durch P-Code zu schützen.

3.2. Kurzhinweise

Wenn Sie nicht die Peeker-Sammeldisk #9 mit dem fertig gelinkten Programm besitzen, so sollten Sie folgendermaßen vorgehen:
1. Das Pascal-Programm „StartCruncher" eingeben und (C)ompilieren. Dieses dann CRUNCH.TEXT und CRUNCH.CODE nennen.
2. Das Assemblerprogramm „MoveCruncher" eingeben und (A)ssemblieren. Dieses dann MOVER.TEXT und MOVER.CODE nennen.
3. Das Assemblerprogramm „Cruncher" eingeben und (A)ssemblieren. Dieses dann CRUNCHER.TEXT und CRUNCHER.CODE nennen.
4. Dann den (L)inker aufrufen und die auftauchenden Fragen folgendermaßen beantworten:
a) Host file? CRUNCH <Return>
b) Lib file? MOVER <Return>
c) Lib file? <Return>
d) Map file? <Return>
e) Output file? <Return>
Der gelinkte File heißt im Augenblick SYSTEM.WRK.CODE. Je nach Bedarf kann man diesen umbenennen. Mögliche Varianten sind unter anderem CRUNCH.CODE (der ungelinkte Codefile CRUNCH.CODE wäre dann gelöscht, was nicht sonderlich schlimm ist), C.CODE oder auch SYSTEM.ATTACH oder SYSTEM.STARTUP.
Das fertig gelinkte Programm kann dann entsprechend aufgerufen werden, also mit R(un oder X(ecute. Wird es SYSTEM.ATTACH oder SYSTEM.STARTUP genannt, wird es beim Booten automatisch ausgeführt, was sehr praktisch ist (Auto-Patch nur im RAM).
Wurde das Programm einmal ausgeführt, kann seine Wirkung nicht mehr rückgängig gemacht werden – außer durch erneuertes Booten.

4. Speicherplatz-Einsparungen

Was im einzelnen an Speicherplatz gespart werden kann, hängt natürlich im wesentlichen von der Länge der geladenen Assemblerprozeduren ab. Wenn diese sehr kurz sind, z.B. weniger als 1 K, dann lohnt sich der Einsatz des Crunchers kaum, da er selbst 600 Bytes Speicher des Aux-RAMs benötigt. Sobald man aber etwa die Turtlegraphics benutzt, ist die Speicherplatzeinsparung beachtlich.
Man denkt normalerweise, daß man bei der 128K-Version ohnehin genügend Speicher für den Code zur Verfügung hat. Ich mußte jedoch schon die Erfahrung machen, daß dies nicht immer der Fall ist. Nachdem ich eine neue, wesentlich erweiterte Turtlegraphics geschrieben hatte (die in Kürze im Hüthig Software Service erscheint, Anm.d.Red.), stellte sich heraus, daß diese über 8K an Maschinenprozeduren benötigte. Da ich auch die Double-Turtlegraphics eingebaut hatte und damit die Hires-Grafikseite im Aux-RAM vor Überschreiben schützen mußte, gingen allein dadurch 14K an Code verloren, weitere 8K durch die doppelt vorhandenen Maschinenprozeduren, also insgesamt 22K. Durch den Cruncher werden davon 8K zurückgeholt. Auf die 8K Grafik muß man ohnehin verzichten, und die anderen 6K kann man durch einen kleinen Trick ebenfalls ausnutzen, so daß die Situation schon wieder freundlicher aussieht. Der gerade erwähnte Trick soll ein andermal erklärt werden.

Die Sache mit dem Verschieben der Maschinenprozeduren vom Aux-RAM ins Main-RAM ist übrigens nicht das einzige Problem bei der 128K-Pascal-Version. Da P-Code interpretiert wird, kann ein Befehl leicht aus dem Aux-RAM geholt werden. Das gilt nicht für Maschinenprozeduren, wie wir oben gesehen haben. Das gleiche gilt aber auch nicht für Daten, die im P-Code enthalten sind. Diese Daten sind konstante Strings und „Packed Arrays of Char", die in Befehlen wie
S := 'Hallo'
auftreten. Nachdem die Adresse des Strings auf dem Evaluation-Stack abgelegt worden ist, kann das System nicht mehr entscheiden, ob diese Adresse auf das Aux-RAM oder Main-RAM zeigt. Bei konstanten Strings würde die Adresse auf das Aux-RAM zeigen, bei Stringvariablen auf das Main-RAM. Eine Adresse hat ja nur zwei Bytes und kein „Main-Aux-Merkmal".
Um die Adresse des konstanten Strings eindeutig festzulegen, wird jeder konstante String beim Auftreten auf den Program-Stack kopiert und dessen Adresse auf den Evaluation-Stack geschoben. Damit der Program-Stack nicht zu schnell überläuft – wie dies bei der Befehls-Sequenz „For I := 1 to 10000 Do Write ('Hallo')" auftreten würde –, wird vom System eine verzeigerte Liste von allen konstanten Strings angelegt, die in einer Prozedur angesprochen werden. Diese Liste wird zuerst durchgegangen, bevor neuer Speicherplatz auf dem Program-Stack reserviert wird. Dadurch werden Zugriffe auf konstante Strings in der Pascal-1.2-128K-Version etwas langsamer als in der 64K-Version.
Außerdem ist dies die vielleicht schon von einigen Programmierern gesuchte Antwort auf die Frage: Warum wird bei der 128K-Version am Anfang einer Prozedur ein anderes „Memavail" ausgegeben als am Ende der Prozedur, wenn in dieser konstante Strings aufgerufen werden und in ihr keine dynamischen Variablen angelegt werden?

In der nächsten Folge befassen wir uns ausführlich mit allen Compiler-Optionen.

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.

---

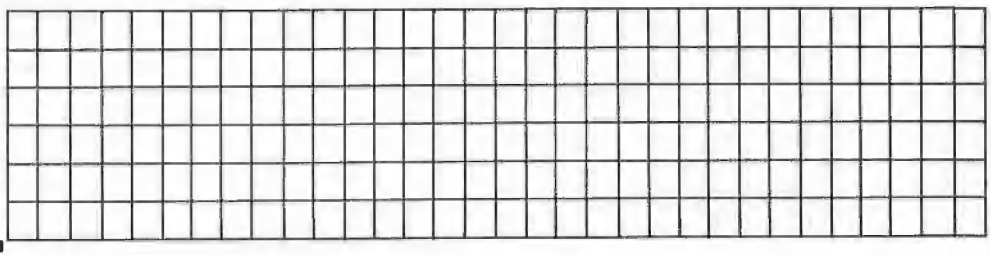
peeker MAGAZIN FÜR APPLE-COMPUTER

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

Der Cruncher umfaßt drei Dateien:

1. CRUNCH.TEXT: Quelltext für Compiler
2. MOVER.TEXT: Quelltext für Assembler
3. CRUNCHER.TEXT: Quelltext für Assembler

1. CRUNCH.TEXT

```
StartCrunch (CRUNCH)

{$L PRINTER:}
{$I-}
{$R-}
{$C von Dieter Geiss, 13. April 85}

program StartCruncher (input, output);

type Buffer = packed array [0..0] of integer;

var  Leng   : integer;
     Blocks : integer;
     A      : ↑Buffer;
     F      : file;

procedure MoveCruncher (var Adr; CodeLeng: integer);
external;

begin
  reset (F, '%CRUNCHER.CODE');
  mark (A);
  if IOresult <> 0
  then writeln (chr (7), 'CRUNCHER.CODE nicht gefunden')
  else
    if blockread (F, A↑, 1) <> 1
    then writeln (chr (7), 'CRUNCHER.CODE defekt')
    else
      begin
        Leng   := A↑ [3];
        Blocks := (Leng + 511) div 512;
        if blockread (F, A↑, Blocks) <> Blocks
        then writeln (chr (7), 'CRUNCHER.CODE defekt')
        else MoveCruncher (A↑, Leng);
      end; {else}
  close (F)
end.
```

2. MOVER.TEXT

```
MoveCruncher (MOVER)

;*****************************************************************
;*                                                               *
;*                    M o v e C r u n c h e r                    *
;*                                                               *
;*                 von Dieter Geiss, 13. April 85                *
;*                                                               *
;*  Diese Prozedur verschiebt den eigentlichen Cruncher in das   *
;*  Auxiliary-RAM, wenn es sich beim vorliegenden Pascalsystem   *
;*  um Version 1.2 handelt und 128K Ram vorliegen.               *
;*  Des weiteren wird zwischen regulärem Laufzeitsystem und dem  *
;*  sogenannten Run-Time-System unterschieden. Vor dem Verschie- *
;*  ben wird eine Adressentranslation um EnterIC durchgeführt.   *
;*                                                               *
;*****************************************************************

               .PROC MoveCruncher,2

;procedure MoveCruncher (var Adr; CodeLeng : integer)

Return         .EQU 00           ;Rückkehradresse von Pascal
CodeLeng       .EQU 02           ;Länge des "Crunchers"
Source         .EQU 04           ;Quelladresse der Verschiebung
Dest           .EQU 06           ;Zieladresse der Verschiebung
Index          .EQU 08           ;Index für Adressentranslation
Number         .EQU 0A           ;Anzahl der zu mod. Stellen
ModPos         .EQU 0C           ;Stelle der Modifikation


Codelow        .EQU 62           ;zeigt auf Untergrenze P-Code
Version        .EQU 0BF21        ;Versionsnummer für Pascal
Flavor         .EQU 0BF22        ;"Duftnote" des Systems
RamWrtMain     .EQU 0C004        ;Schalter für Schreiben Main
RamWrtAux      .EQU 0C005        ;Schalter für Schreiben Aux
Rg128k         .EQU 0EA28        ;Patchadresse reguläres System
RT128k         .EQU 0EA57        ;Patchadresse Run-Time-System

               PLA               ;Hole Pascal-Returnadresse
               STA Return        ;und speichere sie ab
               PLA
               STA Return+1
               PLA               ;Hole Länge des "Crunchers"
               STA CodeLeng
               PLA
               STA CodeLeng+1

               PLA               ;Hole Adresse des Arrays, in
               STA Source        ;den der "Cruncher"
               PLA               ;vom Pascal-Lade-Programm
               STA Source+1      ;geladen wurde

               LDA Return+1      ;Schiebe Pascal-Returnadresse
               PHA               ;wieder auf den Stack
               LDA Return
               PHA

               LDA Codelow       ;Codelow ist die Anfangsadresse,
               STA Patch+1       ;zu der der gepatchte
               LDA Codelow+1     ;Interpreter springen muß
               STA Patch+2

               LDA Version       ;Überprüfe die Versionsnummer:
               CMP #03           ;3 = Pascal 1.2 (oder Modula)
               BNE Exit          ;kein Pascal 1.2

               LDA Flavor        ;Überprüfe die Duftnote:
               CMP #41           ;41 entspricht Run-Time-System
               BEQ PatchRT128k

               CMP #40           ;40 entspricht regulärem System
               BNE Exit          ;keine 128K-Version

PatchReg128k   LDX #02           ;Ein Patch ist erforderlich,
$0             LDA Patch,X       ;und zwar an die Stelle, an
               STA Rg128k,X      ;der ein Segment geladen
               DEX               ;wurde. Vorhandene
               BPL $0            ;Maschinenprozeduren wurden
               BMI Copy          ;schon verschoben.

PatchRT128k    LDX #02           ;Dasselbe gilt für das
$0             LDA Patch,X       ;Run-Time-System
               STA RT128k,X
               DEX
               BPL $0

Copy           JSR Translate     ;Zuerst die Translation!
               LDA Codelow       ;Von der vom Stack geholten
               STA Dest          ;Quelladresse im Main-RAM
               LDA Codelow+1     ;wird der "Cruncher" ins
               STA Dest+1        ;Aux-RAM geschoben
               STA RamWrtAux
               JSR MoveLeft
               STA RamWrtMain

               CLC               ;Der Codelow-Pointer wird
               LDA Codelow       ;hochgesetzt, um den
               ADC CodeLeng      ;"Cruncher" vor dem Über-
               STA Codelow       ;schreiben durch andere
               LDA Codelow+1     ;Codefiles zu sichern
               ADC CodeLeng+1
               STA Codelow+1

Exit           RTS               ;Zurück ins Pascal

Patch          JSR 0800          ;Anfangsadresse des "Crunchers"

Translate      CLC               ;Index wird auf die Anzahl der
               LDA Source        ;zu modifizierenden Adressen
               ADC CodeLeng      ;gestellt:
               STA Index         ;Index = Source + CodeLeng - 14
               LDA Source+1
               ADC CodeLeng+1
               STA Index+1
               SEC
               LDA Index
               SBC #0E
               STA Index
               BCS $0
               DEC Index
$0             LDY #00           ;Anzahl der Adressen
               LDA (Index),Y
               STA Number
               INC Number
               INY
               LDA (Index),Y
               STA Number+1
               JMP $2
$1             LDY #00           ;Die eigentliche Stelle der
               SEC               ;Modifikation ist ModPos
               LDA Index
               SBC (Index),Y
               STA ModPos
               INY
```

```
                LDA Index+1
                SBC (Index),Y
                STA ModPos+1
                DEY
                CLC                 ;Die eigentliche Translation
                LDA (ModPos),Y      ;um Codelow = Enter IC
                ADC Codelow
                STA (ModPos),Y
                INY
                LDA (ModPos),Y
                ADC Codelow+1
                STA (ModPos),Y
$2              SEC                 ;Index weiterschalten
                LDA Index
                SBC #Ø2
                STA Index
                BCS $3
                DEC Index+1
$3              DEC Number          ;Weitere Adressen?
                BNE $1
                DEC Number+1
                BPL $1
                RTS

MoveLeft        LDX CodeLeng+1      ;Verschiebt "CodeLeng" Bytes
                LDY #ØØ             ;von der Quelladresse zur
$Ø              CPX #ØØ             ;Zieladresse
                BEQ $2
                DEX
$1              LDA (Source),Y
                STA (Dest),Y
                INY
                BNE $1
                INC Source+1
                INC Dest+1
                JMP $Ø
$2              LDX CodeLeng
                BEQ $4
$3              LDA (Source),Y
                STA (Dest),Y
                INY
                DEX
                BNE $3
$4              RTS

                .END
```

3. CRUNCHER.TEXT

Cruncher (CRUNCHER)

```
;*****************************************************************
;*                                                               *
;*                        C r u n c h e r                        *
;*                                                               *
;*                 von Dieter Geiß, 13. April 85                 *
;*                                                               *
;* Der "Cruncher" schiebt den P-Code im Aux-RAM so zusammen, daß *
;* die vorher geladenen und vom System auf das Main-RAM gescho-  *
;* benen Assemblerprozeduren bis auf einen rudimentären Rest     *
;* eliminiert werden. Der dabei zurückgewonnene Speicherplatz    *
;* hängt von der Länge der Assemblerprozeduren ab, kann also     *
;* im Einzelfall mehrere K betragen.                             *
;*                                                               *
;*****************************************************************

                .PROC Cruncher

NumMachs        .EQU ØØ         ;Anzahl d. Maschinenprozeduren/Segment
NumProcs        .EQU Ø1         ;Anzahl der Prozeduren im Segment
Index           .EQU Ø2         ;Index in das "Procedure-Dictionary"
Procp           .EQU Ø4         ;Zeiger auf "Procedure-Code-Section"
Movep           .EQU Ø6         ;Ziel des zu verschiebenden rud. Teils
Highp           .EQU Ø8         ;Zeiger auf höchste Maschinenprozedur
High            .EQU ØA         ;Zeiger auf zu verschiebende Prozedur
Entry           .EQU ØC         ;Zeiger auf "Procedure-Entry"
Which           .EQU ØE         ;Nummer der zu verschiebenden Prozedur
Help1           .EQU ØF         ;Hilfvariable 1
Help2           .EQU 1Ø         ;Hilfvariable 2
Dist            .EQU 12         ;Verschiebedistanz
Source          .EQU 14         ;Quelle der P-Code-Verschiebung
Dest            .EQU 16         ;Ziel der P-Code-Verschiebung
Leng            .EQU 18         ;Länge des gewonnenen Speicherplatzes

Codep           .EQU 6Ø         ;Zeiger auf zu ladenden Code
ProcDicp        .EQU 7E         ;Zeiger auf "Procedure-Dictionary"
MachLeng        .EQU 8C         ;Länge aller Maschinenprozeduren
Seg             .EQU 9Ø         ;Segmentnummer

MFlags          .EQU 2ØØ        ;Maschinenprozedur-Markierungen

SegEnd          .EQU ØBC9E      ;Segment-Endadressen
RamRdMain       .EQU ØCØØ2      ;Schalter Main-RAM lesen
RamRdAux        .EQU ØCØØ3      ;Schalter Aux-RAM lesen
RamWrtMain      .EQU ØCØØ4      ;Schalter Main-RAM schreiben
RamWrtAux       .EQU ØCØØ5      ;Schalter Aux-RAM schreiben

                LDX #16         ;Um die Segment-Endadresse
$Ø              LDA ToMove,X    ;festzustellen, wird ein kleines
                STA ØØ,X        ;Programm in die Zero-Page
                DEX             ;geschoben und aufgerufen
                BPL $Ø
                JMP ØØØØ

ToMove          STA RamRdMain   ;Die Adresse des Procedure-
                LDA Seg         ;Dictionary liegt im Main-
                ASL A           ;RAM
                TAX
                LDA SegEnd,X
                STA ProcDicp
                LDA SegEnd+1,X
                STA ProcDicp+1
                STA RamRdAux
                JMP Start

Start           STA RamWrtAux
                LDA #ØØ         ;Initialisierungen
                STA NumMachs
                STA NumProcs
                LDA ProcDicp
                STA Index
                LDA ProcDicp+1
                STA Index+1
                LDY #Ø1
                LDA (ProcDicp),Y ;Anzahl der Prozeduren
                STA Help1        ;im Segment
FindMachs       INC NumProcs
                SEC
                LDA Index       ;Nächster Index in der
                SBC #Ø2         ;Procedure-Dictionary
                STA Index
                BCS $Ø
                DEC Index+1
$Ø              JSR GetProcAddr ;Adresse Procedure-Code-Section
                LDX NumProcs
                LDA #Ø1
                STA MFlags,X    ;Markiere Prozedur als P-Code
                LDY #ØØ
                LDA (Procp),Y   ;Prozedurnummer
                BNE $1
                LDA #ØFF
                STA MFlags,X    ;Markiere Prozedur als Maschinen-
                INC NumMachs    ;prozedur
$1              DEC Help1
                BNE FindMachs   ;Suche weitere Maschinenprozeduren

                LDA NumMachs
                BNE $2          ;Keine Maschinenprozeduren
                JMP Exit        ;im Segment => exit
$2              LDA NumProcs    ;Berechnung der Adresse
                STA Help2       ;der Prozedur, deren Code
                INC Help2       ;als höchste im Segment-Code
                LDA #ØØ         ;vorkommt (Highp)
                STA Help2+1
                ASL Help2
                ROL Help2+1
                SEC
                LDA ProcDicp
                SBC Help2
                STA Movep
                STA Highp
                LDA ProcDicp+1
                SBC Help2+1
                STA Movep+1
                STA Highp+1     ;Highp = Movep
                LDA MachLeng    ;Die Länge des gesamten Maschinen-
                STA Leng        ;codes wird noch gebraucht
                LDA MachLeng+1
                STA Leng+1
                LDA NumMachs    ;Untersuche alle Maschinen-
                STA Help1       ;prozeduren
Search          LDA #ØØ         ;Suche die Adresse der Maschinen-
                STA High        ;prozedur, deren Code als höchste
                STA High+1      ;im Segment-Code vorkommt
                LDA ProcDicp    ;Der Index läuft über alle
                STA Index       ;Prozeduren im Code-Segment
                LDA ProcDicp+1
                STA Index+1
                LDX #Ø1
Search1         SEC
                LDA Index       ;Index weiterlaufen lassen
                SBC #Ø2
                STA Index
```

```
                BCS $Ø
                DEC Index+1
$Ø              LDA MFlags,X
                BPL $1          ;Keine Maschinenprozedur
                JSR GetProcAddr
                LDA Procp       ;Vergleiche Adresse der
                CMP High        ;Procedure-Code-Section
                LDA Procp+1     ;mit der letzten gefundenen
                SBC High+1      ;Adresse
                BCC $1
                LDA Procp       ;Momentane Adresse ist
                STA High        ;größer
                LDA Procp+1     ;Merke also momentane Adresse
                STA High+1
                STX Which       ;Merke Prozedurnummer
                LDA Index       ;Merke Index im Procedure-
                STA Entry       ;Dictionary
                LDA Index+1
                STA Entry+1
$1              INX             ;Nächste Prozedur
                CPX NumProcs
                BCC Searchl     ;Suche weiter
                BEQ Searchl

Crunch          LDX Which       ;Lösche Merker für bearbeitete
                LDA #ØØ         ;Maschinenprozedur
                STA MFlags,X
                SEC             ;Berechne die Verschiebungs-
                LDA Movep       ;distanz in Abhängigkeit des
                SBC High        ;augenblicklichen Movep und
                STA Dist        ;der höchsten gefundenen
                LDA Movep+1     ;Adresse (High)
                SBC High+1
                STA Dist+1
                SEC
                LDA Movep       ;Movep := Movep - 2
                SBC #Ø2
                STA Movep
                BCS $Ø
                DEC Movep+1
$Ø              SEC
                LDA High        ;Highp := Highp - 2
                SBC #Ø2
                STA High
                BCS $1
                DEC High+1
$1              LDY #Ø3         ;Verschiebe den rudimentären
$2              LDA (High),Y    ;Teil (4 Bytes), der noch ge-
                STA (Movep),Y   ;braucht wird.
                DEY
                BPL $2
                SEC             ;Movep muß weitergeschaltet
                LDA Movep       ;werden
                SBC #Ø2
                STA Movep
                BCS $3
                DEC Movep+1
$3              SEC             ;Der Eintrag in das Procedure-
                LDY #ØØ         ;Dictionary muß korrigiert werden,
                LDA (Entry),Y   ;da der Entry-Point der Maschinen-
                SBC Dist        ;prozedur verschoben wurde
                STA (Entry),Y
                INY
                LDA (Entry),Y
                SBC Dist+1
                STA (Entry),Y
                LDA Entry       ;Suche den EnterIC der ver-
                STA Index       ;schobenen Maschinenprozedur
                LDA Entry+1
                STA Index+1
                JSR GetProcAddr
                SEC
                LDA Procp
                SBC #Ø2
                STA Procp
                BCS $4
                DEC Procp+1
$4              CLC             ;Da der EnterIC auf das Main-RAM
                LDY #ØØ         ;zeigt, muß der Wert des EnterIC
                LDA (Procp),Y   ;ebenfalls korrigiert werden
                ADC Dist
                STA (Procp),Y
                INY
                LDA (Procp),Y
                ADC Dist+1
                STA (Procp),Y
                SEC             ;Es werden noch vier Bytes pro
                LDA Leng        ;Maschinenprozedur benötigt
                SBC #Ø4
                STA Leng
                BCS $5
                DEC Leng+1
```

```
$5              DEC Helpl
                BEQ MovePcode   ;Fertig mit Maschinenprozeduren
                JMP Search      ;Suche weitere Maschinenprozeduren

MovePcode       CLC             ;Der P-Code muß jetzt noch
                LDA Codep       ;um MachLeng verschoben werden
                STA Source      ;Die Quelladresse ist der
                ADC Leng        ;momentane Codep, die
                STA Dest        ;Zieladresse Codep + Leng
                LDA Codep+1     ;MachLeng muß noch korrigiert
                STA Source+1    ;werden
                ADC Leng+1
                STA Dest+1
                SEC             ;MachLeng = Highp - MachLeng
                LDA Highp
                SBC MachLeng
                STA MachLeng
                LDA Highp+1
                SBC MachLeng+1
                STA MachLeng+1
                LDA MachLeng    ;MachLeng = MachLeng - Codep
                SBC Codep
                STA MachLeng
                LDA MachLeng+1
                SBC Codep+1
                STA MachLeng+1
                CLC             ;MachLeng = MachLeng + 2
                LDA MachLeng
                ADC #Ø2
                STA MachLeng
                BCC $Ø
                INC MachLeng+1
$Ø              LDA Dest        ;Neuen Codep setzen
                STA Codep
                LDA Dest+1
                STA Codep+1
                JSR MoveRight   ;Verschiebe P-Code
                LDA ProcDicp
                STA Index
                LDA ProcDicp+1
                STA Index+1
                LDX #Ø1
Correct         SEC             ;Korrigiere jetzt noch
                LDA Index       ;alle Einträge von P-Code-
                SBC #Ø2         ;Prozeduren im Procedure-
                STA Index       ;Dictionary des Segments
                BCS $Ø
                DEC Index+1
$Ø              LDA MFlags,X
                BEQ $1          ;Kein P-Code
                SEC             ;Korrigiere Eintrag
                LDY #ØØ
                LDA (Index),Y
                SBC Leng
                STA (Index),Y
                INY
                LDA (Index),Y
                SBC Leng+1
                STA (Index),Y
$1              INX             ;Erhöhe Prozedurnummer
                CPX NumProcs
                BCC Correct     ;Noch nicht alle Prozeduren
                BEQ Correct     ;durchgegangen

Exit            STA RamWrtMain  ;Verlasse den "Cruncher"
                LDX #Ø4         ;über die Zero-Page
$Ø              LDA ExitCode,X  ;Dabei wird nach dem
                STA ØØ,X        ;Umschalten auf Schreiben
                DEX             ;des Main-RAM auch Lesen
                BPL $Ø          ;des Main-RAMs bewirkt
                JMP ØØØØ

ExitCode        STA RamRdMain
                RTS

GetProcAddr     SEC             ;Berechne den Wert:
                LDY #ØØ         ;Procp = Index - Index↑
                LDA Index       ;Dies ist die übliche
                SBC (Index),Y   ;Methode im verschiebbaren
                STA Procp       ;P-Code
                INY
                LDA Index+1
                SBC (Index),Y
                STA Procp+1
                RTS

MoveRight       CLC             ;Diese Prozedur entspricht
                LDA Source      ;der UCSD-Standardprozedur
                ADC MachLeng    ;"MOVERIGHT":
                STA Source      ;Der zu verschiebende Code
                LDA Source+1    ;wird dabei von hinten ver-
                ADC MachLeng+1  ;schoben, damit er sich nicht
```

```
                STA Source+1    ;selbst überschreibt
                CLC             ;Quelladresse ist Source,
                LDA Dest        ;Zieladresse ist Dest,
                ADC MachLeng    ;Länge ist MachLeng
                STA Dest
                LDA Dest+1
                ADC MachLeng+1
                STA Dest+1
                LDY #ØFF
                LDX MachLeng+1
$Ø              DEC Source+1
                DEC Dest+1
                CPX #ØØ
                BEQ $2
                DEX
```

```
$1              LDA (Source),Y
                STA (Dest),Y
                DEY
                CPY #ØFF
                BNE $1
                BEQ $Ø
$2              LDX MachLeng
                BEQ $4
$3              LDA (Source),Y
                STA (Dest),Y
                DEY
                DEX
                BNE $3
$4              RTS

                .END
```

Peeker-Sammeldisk #10

(DOS 3.3; Hefte 9 + 10/1985; Auslieferung etwa 14 Tage nach Erscheinen von Heft 10/85; Einzelpreis DM 28,–; Fortsetzungspreis DM 20,–)

(1) = Zweck; (2) = Heft/Seitenzahl; (3) = Gerätekonfiguration; (4) = Betriebssystem; (5) = Programmstart; (6) = Sonstiges

T.AS.FILER
AS.FILER
STARTUP
ENDUP
RAMDISKLC
(1) Dateikopierprogramm als Applesoft-Erweiterung; (2) Heft 9/85, S. 12; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3 (auch in LC); (5) RUN STARTUP; (6) STARTUP benutzt RAMDISKLC (von Disk #2).

MKBOOT.BAS
T.MKBOOT.OBJ
MKBOOT.OBJ
T.DOBOOT.OBJ
DOBOOT.OBJ
(1) Installierung eines schnellen ProDOS-Boot-Programms; (2) Heft 9/85, S. 20; (3) II+ (mit LC), IIe oder IIc; (4) ProDOS jede Version; (5) s. Heft.

SUPER.HGR
SUPER.HGR.ASM
(1) Hardcopy-Programm für NEC- und ITOH-Drukker; (2) Heft 9/85, S. 30; (3) II+ oder IIe; NEC- oder ITOH-Drucker mit entsprechendem Interface; (4) DOS 3.3; (5) RUN SUPER.HGR (für NEC 8023)

T.FLAG.MONITOR
FLAG.MONITOR
T.FLAG.MONITOR.TEST
FLAG.MONITOR.TEST
(1) Taktzähler für Assemblerprogramme; (2) Heft 9/85, S. 32; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) BLOAD FLAG.MONITOR; BLOAD FLAG.MONITOR.TEST; CALL 768

VERSAL
T.VERSAL
(1) Umwandlung von Klein- in Großbuchstaben in einem Applesoft-Progamm; (2) Heft 9/85, S. 31; (3) II+; (4) DOS 3.3 oder ProDOS; (5) RUN VERSAL; LOAD Programmname; CALL 768

GRAFIK.EDITOR
GRAF.QUATTRO.1
(1) Editor für Hires-Grafik; (2) Heft 10/85, S. 6; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN GRAFIK.EDITOR; (6) GRAF.QUATTRO.1 wird von GRAFIK.EDITOR automatisch geladen

AS.DOUBLE.HIRES.DEMO
T.AS.DOUBLE.HIRES
AS.DOUBLE.HIRES
(1) Double-Hires für Applesoft; (2) Heft 10/85, S. 14; (3) IIe mit 64K-Karte oder IIc; (4) DOS 3.3; (5) RUN AS.DOUBLE.HIRES.DEMO

T.CHARSET
CHARSET
DEMO.CTRL
CTRL.DISABLE
T.ENABLE.DISABLE
HTAB1.BUG
INVERSE.BUG
T.INT.IIE.NEU
(1) Beispiel- und Testprogramme zu den neuen IIe-ROMs; (2) Heft 10/85, S. 22; (3) IIe mit neuen ROMs (Enhanced IIe); (4) DOS 3.3 oder ProDOS; (5) BRUN CHARSET; RUN DEMO.CTRL; RUN CTRL.DISABLE; RUN HTAB1.BUG; RUN INVERSE.BUG; (6) T.INT.IIE.NEU gibt alle geänderten Interpreter-Stellen als Assemblerlisting wieder

EPROM
T.EPROMMER
EPROMMER
(1) Hilfsprogramm zum EPROM-Programmiergerät; (2) Heft 10/85, S. 40; (3) II+ oder IIe mit EPROM-Gerät; (4) DOS 3.3; (5) RUN EPROM; (6) EPROMMER enthält die Firmware des fertigen Geräts

CRUNCH.TEXT
MOVER.TEXT
CRUNCHER.TEXT
(1) Freigabe doppelt belegter Speicher unter Pascal; (2) Heft 10/85, S. 50; (3) IIe mit 64K-Karte oder IIc; (4) Pascal 1.2 128K-Version; (5) E(xecute) CRUNCH (nach dem Assemblieren und Linken, s. S. 51)

PUZZLE
PUZZLE.PDL
(1) Verschiebespiel; (2) Heft 10/85, S. 65; (3) II+, IIe oder IIc mit Maus (PUZZLE) oder Paddles (PUZZLE.PDL); (4) DOS 3.3; (5) RUN PUZZLE oder RUN PUZZLE.PDL

UCSD.TEXT
TURB.PAS
(1) Pascal-Kompaktkurs für Applesoft-Programmierer; (2) Heft 11/85; (3) II+, IIe oder IIc; (4) UCSD.TEXT für Apple Pascal 1.1/1.2 und TURB.PAS für Turbo-Pascal 3.0 oder älter; (5) E(xecute) oder R(un); (6) Vorher UCSD.TEXT mit GETPAS auf Pascal-Format und TURB.PAS mit APDOS auf CP/M-Format konvertiert

Hüthig-FACHBUCH-TIP

Apple Assembler
Tips und Tricks

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

„Apple Assembler" wendet sich an alle, die bereits Anfängerkenntnisse der 6502-Programmierung haben – z.B. aufgrund des Buches „Apple Maschinensprache" – und nunmehr ein Nachschlagewerk für ihren Apple II Plus/IIe/IIc suchen, in dem alle wichtigen ROM-Routinen sowie eine Vielzahl sonstiger Hilfsprogramme in einer systematischen Form zusammengestellt werden. Insgesamt umfaßt dieses Buch über 40 Utilities, darunter mehrere völlig neuartige Programme wie Double-Lores, Double Hires, Screen-Format u.a.
Der erste Teil enthält ein Repetitorium der wichtigsten Befehle, Adressierungsarten und sonstigen Besonderheiten des 6502.

Im zweiten Teil werden alle Adressen des Monitors zusammengestellt, die für Assembler-Programmierer von Nutzen sein können. Darüber hinaus findet der Leser Unterroutinen für hexadezimale Addition/Subtraktion/Multiplikation/Division, Binär-Hex-ASCII-Umwandlung usw.
Der dritte Teil befaßt sich mit der Speicherverwaltung der Language Card und der IIe-64K-Karte und enthält Move-Programme zum Verschieben von Daten in die und aus der Language Card sowie der 64K-Karte.
Der vierte Teil ist dem Applesoft-ROM gewidmet und listet eine große Anzahl nützlicher Interpreter-Adressen. Bei den Utility-Programmen liegt das Schwergewicht auf Fließkommamathematik einschließlich Print Using.

Der letzte Teil behandelt den Text- und Graphikspeicher. Neben einem professionellen Maskengeneratorprogramm werden auch Routinen zur Double-Lores- und Double-Hires-Grafik vorgestellt.

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

BESTELLCOUPON

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

Buchtitel

Name

Straße

Ort

Unterschrift

Pascal 1.2

Eine Richtigstellung

von Bernard Condrau

Der vorliegende Bericht bezieht sich auf den Testbericht aus Peeker 3/85, S. 65ff. „Pascal 1.2 – Evolution statt Revolution" von Claus Rautenstrauch (1) und soll zahlreiche Falschaussagen und Ungereimtheiten aufdecken. Die Reihenfolge richtet sich nach derjenigen von (1) und soll keine Einstufung der Wichtigkeit der einzelnen Möglichkeiten von Pascal 1.2 sein.
„..." (1) kennzeichnet Zitate aus (1).

0. Redaktioneller Hinweis

Wir hatten die Pascal-1.2-Version am 14.11.1984 bei der Firma Apple in München zwecks Testbericht angefordert, jedoch trotz wiederholter Erinnerungsschreiben erst 7 Monate später erhalten, so daß wir natürlich den Testbericht von Claus Rautenstrauch nicht überprüfen konnten. Da Claus Rautenstrauch Mitarbeiter des Applevertragshändlers Hunstig in Münster ist und auch dessen Pascal-Programm DMP-Charger geschrieben hat, bestand eigentlich kein Grund zur Skepsis. Als jedoch unsere Pascal-Experten D. und J. Geiß sofort nach Erscheinen des Testberichts auf gravierende Fehler hinwiesen, wurden wir eines anderen belehrt. In Zukunft werden wir deshalb applespezifische Fachbeiträge von Vertragshändlern nicht mehr unbesehen übernehmen können, denn bei Spezialfragen der nachfolgenden Art scheinen Applehändler möglicherweise überfordert zu sein. Bevor jetzt ein Sturm der Entrüstung losbraust, mögen sich die Applehändler zunächst einmal fragen, warum der Pascal-1.2-Beitrag von keinem einzigen Applehändler und auch nicht von der Firma Apple kritisiert wurde.

1. Das %-Präfix

„Mit dem %-Präfix findet man einen File immer, wenn er „online" ist, unabhängig davon, ob er in Drive 1 oder Drive 2 gespeichert ist" (1).
Dies ist falsch. Richtig verhält sich das %-Präfix wie folgt: Wird der File auf der gleichen Unit nicht gefunden, so erfolgt der übliche Input/Output-Fehler #10 („File not found"). Dieses Präfix ist eine außerordentliche Hilfe beim Erstellen von sog. „Programm-Disketten", da so beim Zugriff auf Datenfiles der Volume-Name (der Diskette, der Profile o.ä.) nicht bekannt sein muß. Beispiel: Das Programm FORMAT.CODE befindet sich auf der Diskette DRUCKER:. Die Anweisung
RESET(file,'%FORMAT.INFO')
im Hauptprogramm von FORMAT-.CODE sucht den File FORMAT-.INFO auf der Diskette DRUKKER:, gleichgültig, ob diese in Drive 1,2,3,... steckt.

2. Unitnummern

„Neu ist auch die Tatsache, daß jetzt auch die Unit-Nummern 13 bis 20 und 128 bis 143 zur Verfügung stehen" (1)
stimmt nicht genau. Die Units #128 bis #143 wurden bereits mit der Version 1.1 eingeführt (neu gegenüber Version 1.0); die Anwendungsweise hat sich gegenüber Version 1.1 nicht geändert. Neu sind die Units #13 bis #20 als blockorientierte Units für „große" Massenspeicher wie z.B. ein Festplattenspeicher (Profile) oder Diskettenlaufwerke mit erhöhter Kapazität. Grundsätzlich könnten auch normale Disketten angesteuert werden. Diese sind aber nicht durch die eingebauten Disk-Routinen ansprechbar, sondern es muß ein eigenes Treiber-Programm (Driver) für diese Drives installiert werden (vgl. dazu (3)). Ein Driver, welcher diesen Unit-Nummern zugeordnet wird, wird vom System (einschließlich Filer) genau gleich behandelt wie Unit #4, 5, 9-12. Nicht so Unit #128-143: Sie können vom Programm nur mit den Unit-Prozeduren (UNITWRITE, UNITREAD, UNITCLEAR, UNITSTATUS) angesprochen werden. Diese Units eignen sich aber nicht nur (eigentlich sogar überhaupt nicht) zum „Verstecken von Daten" (1), sondern zum Einrichten von Treibern, welche besondere Aufgaben erfüllen (z.B. EPROM-Brenner, A/D-Karten usw. ansteuern).

3. Die Swapping-Optionen

Swapping-Option 2:
„Diese Option soll auch noch nicht benutzt werden" (1).
Diese Aussage ist unbegründet, da sie keinesfalls unsauber implementiert ist. In (2) wird lediglich darauf hingewiesen, daß der mit Swap2 reservierte Platz in zukünftigen Anwendungen nicht unbedingt freigegeben wird.

4. Die Disketten-Formattierung

Interessant ist, daß Claus Rautenstrauch davon spricht, daß der neue Formatter „nur" 35 Tracks formattiert. Wie von der Redaktion richtig angemerkt, sind nämlich nicht alle Duodisk- oder Apple-II-Laufwerke fähig, 40 Tracks zu positionieren (mechanische Sperre). Außerdem wären dann nicht mehr alle Apple-II-Modelle im Disketten-Format zueinander kompatibel. Daß die Firma Apple mit einem derartigen Formatierungsprogramm die Fremddrive-Hersteller nur begünstigen würde, ist ebenfalls klar.

5. Gleiche Volume-Namen

Grundsätzlich gilt für alle P-Systeme (Version 1.0, 1.1, 1.2), daß niemals zwei gleichnamige Volumes „on-line" sein sollten, da in vielen Fällen ein Directory vom System nur nach seinem Volume-Namen auf Gültigkeit untersucht wird. Die Aussage „Bei der Version 1.1 konnte man diese Warnung ignorieren, wenn man mit Unit-Nummern arbeitete" (1) entspricht nicht den Gegebenheiten.

6. Die neue Unit CHAINSTUFF

Die in (1) erwähnte Prozedur SWAPGPOFF gibt es nicht, wie aus (2) oder aus dem Interface-Text der Unit klar ersichtlich ist.

7. Die Funktion REMSTATUS aus der APPLESTUFF-Unit

Diese Funktion erwartet nicht ausschließlich eine Super-Serial-Card in Slot 2, sondern eine Apple-Communication-Card oder eine Firmware-Protocol-Card.

8. Bildschirmsteuerung

„Weiterhin kann man mit Ctrl-F=CHR(6) den Cursor unsichtbar und mit Ctrl-E= CHR(5) wieder sichtbar machen." (1)
Dies ist tatsächlich eine neu eingeführte Möglichkeit zur Kontrolle des Cursors unter Pascal 1.2. Durch einen Fehler im ROM-residenten Bildschirm-Treiber des Apple IIe (Revision B) arbeiten diese Ctrl-Zeichen jedoch nicht, wenn die Original-Apple-IIe-80-Zeichen-Karte eingesteckt ist (im 40-Zeichen-Modus des Apple II oder Apple IIe ohne Probleme). In Listing 1 ist ein Programm dargestellt, mit dem man eine Eingabe von Ctrl-F ab Tastatur simulieren kann. (Red. Hinweis: Mit den neuen ROMs des „Enhanced IIe" funktionieren Ctrl-E und Ctrl-F jetzt einwandfrei. us)
Leider funktioniert diese Methode nicht zum Abschalten des Cursors vor der Prozedur GOTOXY. Vorsicht: die verwendeten Speicherstellen sind abhängig von der P-System-Version und müssen richtig behandelt werden. Das Erkennen der aktuellen BIOS-Version wird in Abschnitt „11.Peeken der Version" besprochen. Die Ctrl-Zeichen Ctrl-O (inverse Darstellung CHR(15)) und Ctrl-N (normale Darstellung CHR(14)) sind Eigenheiten des Treibers der 80-Zeichen-Karte, wie sie auch von Apple für den Apple IIe angeboten wird. Der im Apple IIe installierte Bildschirm-Treiber erkennt ein Ctrl-O als Kommando zur inversen Darstellung und ein Ctrl-N als Kommando zur normalen Darstellung. Wenn eine P-Version 1.1 oder 1.2 auf dem Apple IIe mit dieser 80-Zeichen-Karte läuft, kann der Bildschirm mit diesen Ctrl-Zeichen gesteuert werden. Man kann jedoch nicht davon ausgehen, daß andere Fabrikate von 80-Zeichen-Karten die gleichen Steuerzeichen benutzen wie die Apple-Karte. Ist keine 80-Zeichen-Karte installiert, so wird derjenige Bildschirm-Treiber verwendet, der im BIOS des Systems (auf dem File SYSTEM.APPLE) integriert ist. Unter Version 1.2 kann auf die besprochenen Sonderzeichen ebenfalls zugegriffen werden. Ein weiteres Merkmal der Bildschirmsteuerung sollte hier noch zur Sprache kommen, und zwar wie die Treibersoftware für die Apple-80-Zeichen-Karte im Apple IIe installiert ist: Sie befindet sich ROM-resident auf der Hauptplatine des Apple IIe, auch wenn man diese Karte überhaupt nicht kauft oder verwendet. Die Karte selbst trägt nur den Speicher und zwei Steuerbausteine. Damit der Treiber in allen Betriebssystemen lauffähig ist, wurde er so geschrieben, daß er weitgehend unabhängig von dem jeweiligen Betriebssystem ablaufen kann. Es existiert nun ein Flag, welches dem Treiber mitteilt, ob beim Schreiben über das Zeilenende hinaus automatisch auf die nächste Zeile gesprungen werden soll. Unbegreiflicherweise wird dieses Flag beim Booten des P-Systems (1.1 oder 1.2) auf „wahr" gesetzt, so daß dieser automatische Sprung ausgeführt wird. Dies hat mehrere Konsequenzen:
1. Unschöne Darstellung bei „alten" Programmen, welche 80 oder mehr Zeichen in einer Zeile schreiben.
2. Besteht im Editor eine Zeile aus mehr als 79 Zeichen (einschließ-

lich Leerzeichen), so stimmt die dargestellte Cursor-Position auf dem Schirm nicht mehr unbedingt mit der wirklichen Position überein.
3. Ändert man dieses Flag mit einem Poke an die betreffende Adresse (vgl. Listing 2), so führt dies bei Verwendung der Unit TURTLEGRAPH zu vorerst unerklärlichen System-Abstürzen (Grund: Die Unit verwendet die gleichen Zero-Page-Adressen wie der Bildschirm-Treiber; ist das Flag auf „falsch" gesetzt, so werden diese Adressen nach Ausführen der Grafik-Routinen nicht mehr richtig initialisiert = Software-Fehler im ROM!).
Ein neuer Bildschirm-Treiber, welcher alle obengenannten Fehler vermeidet und außerdem noch etwa 4 mal so schnell schreibt wie der eingebaute Treiber, ist über die Firma des Verfassers erhältlich.

9. Die Prozedur UNITSTATUS

Diese Prozedur wurde mit Version 1.1 eingeführt und nicht erst mit Version 1.2. Mit Hilfe dieser Prozedur kann beispielsweise die Funktion KEYPRESS ohne Verwendung der APPLESTUFF-Unit oder einer Assembler-Routine programmiert werden (Listing 3). Beim Testen der „Apple-Keys" oder der Shift-Taste (Modifikation auf einem Apple IIe Revision B: Lötpunkt auf die vorbereitete „Brücke" X6 neben dem Tastatur-Anschluß auf der Hauptplatine) können verschiedene Wege gewählt werden (Tabelle 1). Zur Tastaturabfrage mit UNITSTATUS dürfen die Unit #1 und #2 verwendet werden, da für Unit #1 und #2 vom System immer der gleiche Treiber angesprochen wird (gilt für alle Versionen).

10. Bugs

Diese beseitigten Fehler aus dem System 1.1 waren für mich der ausschlaggebende Faktor, auf das System 1.2 umzustellen. Neben dem völlig unzulänglichen alten SEEK (vgl. (1)) sind noch andere Dinge bemerkenswert, z.B.:
Compiler: Falsches Laden von Datensegmenten bei Verwendung der Resident-Option; verkehrte Reihenfolge der Initialisierung bei geschachtelten Units; zu frühe Freigabe von Symboltable-Speicher; falsche Verarbeitung von negativen Long-Integer-Zahlen und von der Konstante -32768; falsche Verarbeitung von String-Konstanten mit einer Länge größer als 80 Zeichen; Zufallsfehler bei Identifiers, welche mit H,J,K,Q,X,Y oder Z beginnen usw.
Input/Output: File-Input/Output funktioniert nicht korrekt für die Units #4 bis 12, wenn ein neuer Treiber für die betreffende Unit am System angeschlossen ist; Zugriff vom Pascal auf DOS-formatierte Disketten ohne Fehlermeldung (IORESULT=0, „File found"!) usw.
Turtlegraph: YSKIP in der Prozedur DRAWBLOCK wird nicht verarbeitet; DRAWBLOCK arbeitet falsch, wenn nur ein Teil des Blocks in das aktuelle Window hineinpaßt.
Allgemeines: Wechseln der Bildschirmseiten im 40-Zeichen-Modus (Ctrl-A) während des Compilierens kann zu Fehlern führen, mehrfaches Wechseln sogar zur Zerstörung des Directory (!); kein „Stack-overflow"-Test beim Laden von Datensegmenten einer Intrinsic Unit usw.

11. Peeken der Version

Etwas unklar erfolgte hier die Auflösung der Informationsbytes, welche Interpreter-Version auf der Maschine arbeitet. Besseren Aufschluß darüber sollte Tabelle 2 geben.

12. Speicherverwaltung

Daß Programme durch Verwendung von ungepackten, also größeren Dateistrukturen schneller werden, ist anhand des folgenden Beispiels zu verstehen:

```
VAR
ALFA: PACKED ARRAY [0..79] OF
      CHAR;
BETA: ARRAY [0..79] OF CHAR;
```

Die Variable ALFA benötigt 80 Bytes an Variablen-Speicherplatz; der Zugriff auf ein Element aus dem Array braucht jedoch etwas mehr Programmcode-Speicherplatz und ist auch etwas langsamer als ein Zugriff auf ein Element von BETA. BETA benötigt jedoch 160 Bytes an Variablen-Speicherplatz, obwohl mit (konventionellem) Speicherzugriff nicht mehr Informationen abgelegt werden können als in ALFA. Wird ein Programm identisch von Pascal 1.1 auf Pascal 1.2 128K übernommen, ohne daß die Datenstrukturen geändert werden, so läuft dieses Programm in Pascal 1.2 etwas langsamer ab, da ständig zwischen P-Code-Speicher und Variablen-Speicher umgeschaltet werden muß. Näheres siehe (1). Leider werden vom Pascal 1.2 nicht immer größtmögliche Betriebssystemteile in den Speicher geladen; der Editor käme nämlich unter dem 128K-System ohne Segment-Swapping aus. Außerdem hat der 1.2-Editor immer noch den gleichen Fehler wie unter Version 1.1: Er gibt etwa 3000 Bytes an nutzbarem Speicher zu wenig frei.

13. Die Program-Library

„Da man aber weiterhin nur 16 Segmente im Speicher haben darf, müssen diese zusätzlichen Segmente als Intrinsic Units in eine Library eingebaut werden" (1)
ist natürlich falsch. Version 1.1 erlaubte im Zugriff maximal 24 Benutzer-Segmente, da die Segmente #0, 2 bis 6, 30 und 31 vom System reserviert wurden. Diese Segmente konnten sich auch alle gleichzeitig im Speicher befinden. Auch die von Claus Rautenstrauch angesprochenen Intrinsic Units befinden sich ja während der Programmausführung im Speicher, falls sie nicht extra mit der Compiler-Noload-Option ausgelagert wurden. Version 1.2 erlaubt nun deren 50 (!) Segmente gleichzeitig im Zugriff. Wenn man die Segmente #58 bis 63 noch dazu nimmt, welche die Firma Apple für zukünftigen System-Gebrauch reserviert, sind es sogar 56 Segmente (wie beim Pascal-System des Apple III). Bei Version 1.2 sind ebenfalls nur 16 Segmente innerhalb eines Codefiles oder eines Library-Files möglich. Durch die Möglichkeiten der Library-Name-Files fällt diese Beschränkung jedoch nicht weiter ins Gewicht. Nicht möglich ist es allerdings, die Libraries auf verschiedene Disketten zu verteilen und dann auf deren Namen mit dem %-Präfix zuzugreifen, wie bereits in Abschnitt 1 eingehend erläutert wurde. Ebenfalls etwas verfälscht wird in (1) der Zugriff auf die einzelnen Libraries beschrieben. Man muß den Zugriff auf diese unterscheiden:
a) während der Programmausführung:
Dann muß neben dem Codefile noch ein Library-File auf der gleichen Diskette verfügbar sein oder die SYSTEM.LIBRARY auf der Boot-Diskette. Beispiel: Die Diskette DRUCKER: beinhaltet einen File FORMAT.CODE und einen Library-File oder einen Library-Name-File FORMAT.LIB. Im 0-ten Block des Codefiles liest das System vor der Ausführung des Programms die Information, welche Intrinsic-Segment-Nummern vom Programm benötigt werden. Die erstgefundene Unit mit übereinstimmender Nummer wird verwendet, ohne daß der Unit-Name in Betracht gezogen wird. Am besten versichert man sich immer, daß eine spezielle Unit-Nummer nur einmal für eine Intrinsic Unit verwendet wird (dies gilt auch für Datensegmente).

b) beim Compilieren:
Der Compiler sucht nach wie vor jede Unit im File SYSTEM.LIBRARY. Will man den Compiler darauf hinweisen, daß die benötigten Units im File DRUCKER:FORMAT.LIB gesucht werden sollen, so muß die entsprechende Option im Textfile gesetzt werden (z.B. USES {$U DRUCKER:FORMAT.LIB} PRINTUNIT;).

14. Kompatibilität

„Man kann die System-Files beider Systeme mischen" (1)
stimmt leider nicht. Version 1.2 läßt die Ausführung von System-Files der Version 1.1 nicht zu (Man kann das Betriebssystem hintergehen, indem man die Files nicht mit Tastendruck, sondern über das eXecute-Kommando ausführt. Dies kann aber zu bösen Überraschungen führen, vor denen gewarnt sei). Folgende Fälle können zu fehlerhaften oder von der Version 1.1 verschiedenen Resultaten führen, falls man sie unter Version 1.2 benutzt:

– Die Verwendung der Prozedur UNITSTATUS: Unter Version 1.1 wurden als Antwort 2 Bytes in die Status-Variable geschrieben, unter Version 1.2 aber 8 Bytes. Wird nun diese Prozedur übernommen und besitzt sie eine zu klein definierte Status-Variable, so können unvorhersehbare Folgen auftreten (vgl. Listing 3).
– System-Variablen: Umfang und Standort haben sich geändert, daher führen Pokes oder Peeks an absolute Speicher-Adressen nicht zum Ziel.
– Der benötigte Speicherplatz der lokalen Daten einer Prozedur, welche String-Konstanten enthält, ist größer unter Version 1.2 128K.
– Programme, welche mit Varianten-Rekords (vgl. Listing 3) auf den Codefile im Speicher zugreifen.
– Assembler-Routinen, welche Adressen im BIOS oder im Interpreter verwenden.
– Da der 80Store-Switch (Zugriff auf Text- und Grafik-Seiten) in Version 1.2 nicht immer genauso gesetzt ist wie in den Versionen 1.0 und 1.1, kann es bei Routinen, welche in den Bildschirm „poken", zu Schwierigkeiten kommen (vgl. (5)).
– (Assembler-)Routinen, welche eine genau definierte Zeitspanne

während der Ausführung voraussetzen.
– eventuell weitere Fälle, welche ich selbst noch nicht kenne.

Quellenangabe

(1) Claus Rautenstrauch, Pascal 1.2 – Evolution statt Revolution, Peeker 3/85

(2) Apple Pascal 1.2 Update Manual

(3) Barry Haynes, Attach-Bios Document for Apple II Pascal 1.1

(4) Apple Pascal Language Reference Manual und Apple Pascal Operating System Reference Manual

(5) Apple IIe Reference Manual

(6) Apple IIe Reference Manual Addendum: Monitor ROM Listings

Tabelle 1

Möglichkeiten, wie die Abfrage der Tasten Open-Apple, Solid-Apple und Shift in einem Pascal-Programm vorgenommen werden kann (B = Button):

	Apple-stuff-Unit	Programmiertes Peek	Unitstatus (Listing 3)	Speziell	B-Wert der nicht-gedrückten Taste
Open-A (= B Ø)	B(Ø)	$CØ61(-16287)	2. Wort	High-Bit gesetzt	B frei
Solid-A (= B 1)	B(1)	$CØ62(-16286)	3. Wort	---	B frei
Shift (= B 2)	B(2)	$CØ63(-16285)	4. Wort	---	B gedrückt

Beispiel für das gesetzte High-Bit:
Open-Apple nicht gedrückt: ORD('A')=65
Open-Apple gedrückt: ORD('A')=193 (=65+128)

Tabelle 2

Tabelle 2.1: Erkennen der Hardware unter Version 1.2:
Adresse $BF31 (-16591) gibt über die Hardware Auskunft.

	Bit 7	Bit 6	Bit 1	Bit Ø
Es handelt sich um einen IIe	1	Ø	Ø	Ø
IIe mit 8Ø-Z/Z-Karte (Apple)	1	Ø	Ø	1
IIe mit 8Ø-Z/Z-Karte + 64k	1	Ø	1	1
II	Ø	Ø	Ø	Ø

Tabelle 2.2: Erkennen der Software:
Adresse $BF21 (-166Ø7) gibt Auskunft über die verwendete Version:

	Bit 7-2	Bit 1	Bit Ø
UCSD-Pascal II Version 1.Ø	Ø	Ø	Ø
UCSD-Pascal II Version 1.1	Ø	1	Ø
UCSD-Pascal II Version 1.2	Ø	1	1

Tabelle 2.3: Adresse $BF22 (-166Ø6) gibt Auskunft über die Interpreter-Version:

	Bits 7	6	5	4	3	2	1	Ø
Bedeutung für Version 1.2								Ø
Pascal-Entwicklungssystem (normal)								1
Pascal-Laufzeitsystem (speziell für Software-Entwickler)							1	
Real-Zahlen nicht unterstützt						1		
Set-Operationen nicht unterstützt					1			
Interpreter für 48K-Systeme			Ø	1				
Interpreter für 64K-Systeme			Ø	Ø				
Interpreter für 128K-Systeme			1	Ø				
Console-Output (Unit #1/2) geht über Treiber auf die Texts. ($4ØØ-$7FF)	Ø							
Console-Output (Unit #1/2) geht über Treiber auf Grafiks. ($2ØØØ-$3FFF) wird vom Standard-Treiber nicht unterstützt	1							

	$BF22
Bedeutung für Version 1.Ø	Ø
keine zu unterscheidenden Versionen	

	$BF22
Bedeutung für Version 1.1	1
Pascal-Entwicklungssystem (normale Version)	
Pascal-Laufzeitsysteme:	
48k-Interpreter:	
Alles unterstützt	2
Set-Operationen nicht unterstützt	3
Real-Zahlen nicht unterstützt	4
Set und Real nicht unterstützt	5
64k-Interpreter:	
Alles unterstützt	6
Set-Operationen nicht unterstützt	7
Real-Zahlen nicht unterstützt	8
Set und Real nicht unterstützt	9

Listing 1

Programm zum Abschalten der Bildschirmausgabe während der Programmausführung. Das Programm stützt sich darauf, daß der Standard-Apple-Treiber zur Tastaturabfrage benutzt wird.
Bit 6 in der entsprechenden Speicherstelle ($BF15 resp. $FA) gibt dem System Auskunft darüber, ob die Ausgabe an den Bildschirm erfolgen soll.
Leider ist dieses Verfahren nicht brauchbar, um den Cursor bei GOTOXY-Befehlen abzuschalten, da auch die Prozedur GOTOXY als Spezialzeichen-Ausgabe an den Bildschirm im Treiber implementiert ist.

```
PROGRAM FLUSH_CRT;
{alle nachfolgenden Programme von B. Condrau}

CONST
 VERSION  = -166Ø7; {$BF21 Versions-Byte}
 CONFLGS1 = -16619; {$BF15 CONsole FLaGS:
                     Info ueber Flush, Start/Stop,}
 CONFLGS2 =    25Ø; {$ØØFA Auto-Follow (Ctrl Z),
                     Pagel/2 (Ctrl A)}
 VERS1Ø = Ø; {Versions-Identifikation fuer Pascal 1.Ø}
 VERS11 = 2; {Versions-Identifikation fuer Pascal 1.1}
 VERS12 = 3; {Versions-Identifikation fuer Pascal 1.2}
TYPE
 WORD=PACKED ARRAY [Ø..1] OF Ø..255;
 VARIANT=RECORD CASE INTEGER OF
  Ø:(I:INTEGER);
  1:(P:↑WORD);
 END;
VAR
 CH:CHAR;

FUNCTION PEEK(ADRESSE:INTEGER):INTEGER;

VAR
 QUICK:VARIANT;
BEGIN
 QUICK.I:=ADRESSE; {Adresse laden}
 PEEK:=QUICK.P↑[Ø];{Wert dieser Adresse holen}
END;

PROCEDURE POKE(ADRESSE,WERT:INTEGER);

VAR
 QUICK:VARIANT;
BEGIN
 QUICK.I:=ADRESSE; {Adresse laden}
 QUICK.P↑[Ø]:=WERT MOD 256;{1 Byte
  in diese Adresse speichern}
END;

PROCEDURE FLUSH_ON;
 {Ausgabe ausschalten}
BEGIN
 IF PEEK(VERSION)=VERS12 THEN
  POKE(CONFLGS2,ORD(ODD(PEEK(CONFLGS2))
   OR ODD(64)))  {CONFLGS2 OR #4Ø}
 ELSE
  IF PEEK(VERSION)=VERS11 THEN
   POKE(CONFLGS1,ORD(ODD(PEEK(CONFLGS1))
    OR ODD(64)));{CONFLGS1 OR #4Ø}
END;

PROCEDURE FLUSH_OFF;
{Ausgabe einschalten}
BEGIN
 IF PEEK(VERSION)=VERS12 THEN
  POKE(CONFLGS2,ORD(ODD(PEEK(CONFLGS2))
   AND ODD(255-64)))  {CONFLGS2 AND #ØBF}
 ELSE
  IF PEEK(VERSION)=VERS11 THEN
   POKE(CONFLGS1,ORD(ODD(PEEK(CONFLGS1))
    AND ODD(255-64)));{CONFLGS1 AND #ØBF}
END;
```

```
BEGIN
 {Ein Beispiel fuer die Anwendung}
 PAGE(OUTPUT);
 WRITELN;
 WRITE('Wollen Sie die nächste Zeile am Bildschirm sehen?');
 READ(CH);
 IF CH='N' THEN FLUSHON;
 WRITELN;
 WRITE('Die Ausgabe an den Bildschirm ist EINgeschaltet.');
 FLUSHOFF; {VORSICHT: Unbedingt wieder zurueckschalten}
 WRITE('...ok...');
END.
```

Listing 2

```
Programm zum Verändern des MODE-Bytes, d.h.
automatischer Sprung am Zeilenende auf die nächste Zeile.
Bit Ø im MODE-Byte gibt dem System Auskunft darüber,
ob der Sprung ausgeführt werden soll.
Vgl dazu auch (6). Zu PEEK und POKE vgl. Listing 1.


PROGRAM WRAP_AROUND;

CONST
 MODE = 1275; {$4FB}
TYPE
 WORD=PACKED ARRAY [Ø..1] OF Ø..255;
 VARIANT=RECORD CASE INTEGER OF
  Ø:(I:INTEGER);
  1:(P:↑WORD);
 END;
VAR
 CH:CHAR;

FUNCTION PEEK(ADRESSE:INTEGER):INTEGER;

VAR
 QUICK:VARIANT;
BEGIN
 QUICK.I:=ADRESSE; {Adresse laden}
 PEEK:=QUICK.P↑[Ø];{Wert dieser Adresse holen}
END;

PROCEDURE POKE(ADRESSE,WERT:INTEGER);

VAR
 QUICK:VARIANT;
BEGIN
 QUICK.I:=ADRESSE; {Adresse laden}
 QUICK.P↑[Ø]:=WERT MOD 256;{1 Byte
  in diese Adresse speichern}
END;

BEGIN
 PAGE(OUTPUT);
 WRITELN('Sie verwenden einen Apple //e');
 WRITELN('mit der Apple-8Ø-Zeichen-Karte.');
 WRITELN('Wollen Sie den automatischen Sprung');
 WRITELN('am Zeilenende ein- oder ausschalten?');
 WRITE  ('E(inschalten, A(usschalten, ESC:');
 READ(CH);
 WRITELN;
 IF CH='E' THEN POKE(MODE,ORD(ODD(PEEK(MODE))
  AND ODD(254)));{Loescht Bit Ø}
 IF CH='A' THEN POKE(MODE,ORD(ODD(PEEK(MODE))
  OR TRUE));{Setzt Bit Ø}
 FOR CH:='!' TO 'ß' DO WRITE(CH:2);
END.
```

Listing 3

```
Funktion KEYPRESS in Pascal ohne Assembler-Code.
Verwendung von
  UNITSTATUS(unit,status<variable>,control<word>):
  unit:    Nummer der Unit (hier 1 oder 2)
  status:  TYPE KEYSTAT:Variable, in welcher
           die Antwort gespeichert werden
           soll (8 Byte lang für Unit 1 oder 2); s.u.
  control: Kontroll-Nummer, Bedeutung:
           control=Ø: Antwort in status unbestimmt,
            IORESULT=Ø nach UNITSTATUS
            (bedeutet: Bereit für Ausgabe an den Bildschirm)
           control=1: Antwort in status.NUM=Ø:
            Kein Zeichen gedrückt.
           Antwort in status.NUM>=1: JA,
            es wurde ein Zeichen gedrückt.
            Bem: Der ROM-residente Treiber des IIe gibt die
            Anzahl der gedrückten Zeichen nicht richtig zurück!
           control>=2: Ungültiges Kontrollwort:
            Rückkehr mit IORESULT=3
            ("Illegal Operation" - Unzulässige Operation)

PROGRAM KEY_CHECK;

TYPE
 KEYSTAT=RECORD
  NUM:INTEGER;  {Anzahl der gedrueckten Zeichen}
  OPEN:BOOLEAN; {Open-Apple-Taste gedrueckt (nur 1,2)}
  SOLID:BOOLEAN;{Solid-Apple-Taste gedrueckt (nur 1.2)}
  SHIFT:BOOLEAN;{Shift-Taste gedrueckt (nur 1.2)}
 END;
VAR
 CH:CHAR;
 I:INTEGER;

FUNCTION KEYPRESS:BOOLEAN;

VAR
 STATUS:KEYSTAT;

BEGIN
 STATUS.NUM:=Ø;
 UNITWRITE(1,STATUS,1);
  {erzwingt, dass die Tastatur abgefragt wird; da dieses
   Byte=Ø gesetzt ist wird kein Zeichen geschrieben}
 UNITSTATUS(1,STATUS,1);
  {falls keine Taste gedrueckt wurde, beinhaltet STATUS.NUM
   immer noch Ø}
 KEYPRESS:=STATUS.NUM<>Ø;
END;

BEGIN
 PAGE(OUTPUT);
 WRITELN('Drücken sie eine Taste um aufzuhören');
 WHILE NOT KEYPRESS DO
  BEGIN
   WRITE('.');
   I:=Ø;
   WHILE (I<1ØØØ) AND NOT KEYPRESS DO I:=I+1;
  END;
 READ(KEYBOARD,CH); {Die gedrueckte Taste lesen}
 WRITELN;
 WRITE('Danke');
END.
```

Schwierigkeiten mit den Graf-quattro-Cursoren.

Es haben sich bei der Redaktion Leser gemeldet, die Schwierigkeiten mit den Graf-quattro-Cursoren hatten. Obwohl die Routinen getreulich abgetippt wurden, erschien nach den vorgeschriebenen CALLs eine Ansammlung von „wild gewordenen Linien" und nicht die versprochenen +- und x-Cursoren. Hierzu gibt es eine einfache Erklärung, die ich in meinem Beitrag zu erwähnen vergessen habe (mea culpa): Die Cursoren sind Shapes, und Shapes brauchen zwei Parameter, nämlich ROT und SCALE. Also, bevor Sie die Cursoren in Betrieb setzen, den Befehl SCALE = 1 nicht vergessen. Dieser Befehl setzt nur den Zero-Page-Pointer $E7 (231) auf den Wert 1, so daß auch ein POKE 231,1 zu demselben Ergebnis führt. Im Teil 4 auf S. 6 in diesem Heft ist dieser Patch nicht erforderlich.

Ich hoffe, hiermit dem Leser manchen Frust zu ersparen.

N. G. Barbieri

Was ist Logo?

Eine Kurzeinführung

von Kurt Rudl

Zusammenstellung der entsprechenden Anweisungen

BASIC	Logo
PRINT A	PR THING "A bzw. PR :A
PRINT "ABC"	PR "ABC
PRINT " 3-4"	PR "\ 3\-4
B = 5	MAKE "B 5
Z$ = "XYZ"	MAKE "Z "XYZ
X$ = " XYZ"	MAKE "X "\ XYZ
A = B	MAKE "A THING "B oder
	MAKE "A :B

Nicht nur Turtlegraphic

Schon von den natürlichen Sprachen her ist bekannt, daß man nicht wörtlich übersetzen kann. Diese Schwierigkeit tritt auch bei Programmiersprachen auf, besonders, wenn sie von sehr verschiedener Struktur sind wie BASIC und Logo. Begriffe, die der einen Sprache entnommen werden, passen nicht auf die andere. Es müssen Kompromisse geschlossen werden. Da auf Grund der Verfügbarkeit BASIC einen großen Bekanntheitsgrad aufweist, soll es als Ausgangspunkt dienen. Logo kennen viele dem Namen nach, und oft wird auf die entsprechende Frage geantwortet, daß es eine Sprache für Kinder sei. Diese irrige Meinung rührt sicher von der Turtlegraphic – oder wie es moderner heißt, der Igelgrafik – her. Dabei handelt es sich aber nur um einen kleinen, sehr speziellen Anwendungsbereich von Logo. Die Hauptbedeutung liegt vielmehr darin, daß es sich bei Logo um einen Abkömmling von LISP handelt. LISP ist von der Struktur her grundverschieden zu Sprachen wie BASIC, FORTRAN, ALGOL, Pascal und Ada.

Diese Verschiedenheit von Logo und LISP zu den gängigen Programmiersprachen hat mitunter zu der Meinung geführt, man möge von vorn anfangen und alles, was man von BASIC her weiß, vergessen. Nur wer kann schon alles vergessen und wer ist gewillt, von vorn anzufangen (zumal BASIC ja „so schön bequem" ist).

Logo gibt es in verschiedenen Versionen. Wie wenig Normierungsversuche helfen, kann man an Pascal sehen. Wenn die Praxis es erfordert, übergeht man gern die Norm und bedient sich der Hilfsmittel, die außerhalb der Norm zur Verfügung gestellt werden – wie man z.B. an UCSD-Pascal sehen kann. Es gibt auch deutsche Versionen von Logo. Ob das im Endeffekt Vorteile bringt, wage ich zu bezweifeln. Englisch ist nun einmal die Grundlage für alle Computersprachen, zumal jeder in der Schule diese Sprache lernt. Ob ich den Befehl DRUCKEZEILE oder PRINT lernen muß, bleibt sich gleich. Ich finde es gut, wenn in der „Computerei" durch das Amerikanisch eine gewisse Internationalität hergestellt wird. Deshalb neige ich zur englischen Version von Logo, auch wenn es Argumente für die Verwendung der deutschen Version geben sollte. Sicherlich lohnt es nicht, dies zu einer Streitfrage werden zu lassen.

Taschenrechnermodus

Logo kann man wie BASIC im Taschenrechnermodus verwenden (= Eingabe von sofort ausführbaren Befehlen über die Tastatur). Hier sind die Unterschiede zwischen beiden Sprachen ziemlich gering.

Falls sich jemand die BASIC-Eingaben ohne Zwischenräume gar nicht erst angewöhnt hat, treten kaum Probleme auf. (Dies ist bei Applesoft im Gegensatz zu MBASIC grundsätzlich zulässig.) Dem „PRINT" bzw. „?" entsprechen in Logo „PRINT" bzw. „PR". *PR 3+4, PR 3+4*7, PR 3/4* usw. zeigen, daß sich Logo genauso verhält wie BASIC.

Im Taschenrechnermodus kann man die vorgegebenen Bibliotheksfunktionen durchprobieren. Dabei tritt viel Bekanntes auf: ARCTAN (ATN), SIN, COS, INT und SQRT (SQR).

Allerdings findet man kein „EXP" und auch kein „LOG". Das ist zwar erstaunlich, muß aber hingenommen werden. Vielleicht sind diese Funktionen vorhanden, wenn demnächst ein Rechner erscheint, der Logo an Stelle von BASIC benutzt. (J. Tramiel will dies mit seinem neuen Atari versuchen.)

Die BASIC-Anweisung *SQR(2)* wird in Logo zu *SQRT 2* bzw. *SQRT(2)*. „INT" verhält sich in Logo und BASIC etwas unterschiedlich, da Logo nur die Nachkommastellen abschneidet, was bei negativen Zahlen zu einem anderen Ergebnis führt. Bei der Sinus- bzw. Cosinusfunktion ist zu beachten, daß in Logo das Argument im Gradmaß erwartet wird und nicht im Bogenmaß wie in BASIC. Entsprechendes gilt auch für den „ARCTAN". In Logo gilt *ARCTAN(1)* = 45, in BASIC *ATN(1)* = PI/4. Übrigens, auch *ARCTAN -1* ergibt etwas anderes als in BASIC. Der berühmte Lehrsatz des Pythagoras in BASIC *PRINT SQR (3 ↑ 2 + 4 ↑ 2)* wird in Logo zu *PR SQRT(3*3 + 4*4)* – den Potenzierungsoperator „ ↑ " gibt es nicht.

Ausgabe von Strings

Die Unterschiede zwischen BASIC und Logo merkt man, wenn es sich um die Ausgabe von Strings handelt. *PRINT "ABCD"* wird in Logo zu *PR "ABCD*. (Das Fehlen des letzten Hochkommas ist dabei nicht als allzu großer Unterschied zu sehen.) Wenn man aber vor dem ersten Buchstaben ein Leerzeichen einfügt, weiß Logo plötzlich nicht mehr, was es machen soll. In BASIC bereitet *PRINT "ABCD"* keine Probleme, in Logo führt *PR "ABCD* zu einer Fehlermeldung. Der Grund liegt darin, daß das Leerzeichen genau wie die eckigen und runden Klammern, die Größer/Kleiner-Symbole und das Gleichheitszeichen noch zusätzlich die Trennfunktion ausführen. Dieselbe Fehlermeldung würde also auch bei *PR "=ABCD* auftreten. Dem Logo-Interpreter muß mitgeteilt werden, daß nur das Zeichen allein und nicht die zusätzliche Trennfunktion genommen werden soll. Das geschieht dadurch, daß vor dem mit der Trenneigenschaft versehenen Symbol ein Ctrl-Q eingegeben wird. Auf dem Bildschirm erscheint der sog. Backslash „\" oder „Ö" (nicht zu verwechseln mit dem Divisionsoperator „/"). Der BASIC-Befehl *PRINT " ABCD"* wird damit in Logo zu *PR "\ ABCD*.

Wertzuweisung

Bekanntlich erfolgt dies in BASIC z.B. mit *A=3*. Dies hat ALGOL-Programmierer schon immer geärgert, deshalb schreibt man dort wie auch in der jüngeren Sprache Pascal a:=3. In Logo ist das Gleichheitszeichen wie bei den ALGOL-Sprachen ein Vergleichszeichen. Um einen Wert in einen Speicherplatz mit der Bezeichnung A zu schreiben, benutzt Logo den MAKE-Befehl. *A=3* wird in Logo zu *MAKE "A 3*. Natürlich kann man dann den Wert der Variablen nicht mit *PR "A* ausgeben, denn dann wird lediglich A auf den Bildschirm geschrieben. Man muß schon sagen: „Gib den Wert im Speicher A aus". Dafür gibt es die Anweisung „THING". Mit *PR THING "A* wird 3 ausgegeben, wie es in BASIC mit *PRINT A* erfolgen würde. Da das Schreiben von *THING "* zu umständlich erschien, wurden die beiden Angaben durch den Doppelpunkt „:" ersetzt. *PR :A* führt zum gleichen Ergebnis wie *PR THING "A*.

Beispiele:

```
MAKE "A 5
MAKE "123 7
PRINT THING "A
PRINT THING "123
PRINT :A
PRINT :123
```

Aus den Beispielen ist zu sehen, daß auch reine Ziffernkombinationen als Speicherbezeichnung verwendet werden können.

Wenn Strings in BASIC abgespeichert werden sollen, geschieht das in entsprechend gekennzeichneten Variablen, z.B. *Z$="BASIC"*. In Logo ist eine besondere Kennzeichnug der Variablen, die Strings aufnehmen sollen, nicht erforderlich. In Logo schreibt man wie folgt: *MAKE "Z "BASIC*. Mit *PR :Z* (also *PR THING "Z*) erfolgt die Ausgabe auf den Bildschirm. Das funktioniert natürlich nur, solange keine Zeichen mit Trennungseigenschaft verwendet werden. Die Anweisung *X$="3+3"* muß in Logo also lauten: *MAKE "X "3\+4*. Durch die Verwendung des Backslash wird die Trenneigenschaft des (+)/(-)-Zeichens aufgehoben. Die scheinbar komplizierte Verwendung von Variablen in Logo führt bei weitergehenden Anwendungen zu Vorteilen (siehe hierzu die **Zusammenstellung der entsprechenden Anweisungen**).

Man kann sich die bereits definierten Variablen und deren Wert ausgeben lassen. Das geschieht mit Hilfe der Anweisungen „PONS"

oder „POALL". Dabei listet „POALL" außer den Variablen auch noch die Prozeduren auf. Es lohnt sich, mit den verschiedenen Anweisungen zu experimentieren, um sich mit der zunächst befremdenden Schreibweise vertraut zu machen.
Grundbegriffe in Logo sind Wort und Liste. Dabei ist das Wort mit dem String in BASIC vergleichbar, sofern keine Zeichen mit Trenneigenschaft auftreten. Der Begriff Liste ist erheblich komplizierter, so daß hierfür nur ein Beispiel angegeben wird:
MAKE "A [1 5 ASD [E RTZ]].
Diese Liste kann mit *PR :A* ausgegeben werden. (Die eckigen Klammern entsprechen dem „Ä" und „Ü".)

Prozeduren
Für den BASIC-Programmierer ist der Begriff der Prozedur nicht so geläufig wie für den Pascal-Programmierer. Aber auch zwischen Pascal und Logo gibt es Unterschiede im Prozedur-Begriff. Formalparameter gibt es in BASIC nur bei den Funktionen; bei den Subroutinen werden keine Parameter – wenn man von einigen BASIC-Dialekten absieht – benutzt. Wenn einfache Prozeduren gewählt werden, ergeben sich auch für den BASIC-Programmierer keine Schwierigkeiten. Eine Prozedur wird zwischen „TO" und „END" eingeschlossen. Hinter „TO" muß der Prozedur-Name stehen. Eine einfache Prozedur sieht dann wie folgt aus:

```
TO BEISPIEL
PR "BEISPIEL
PR [ AUCH EINE LISTE KANN
AUSGEGEBEN WERDEN ]
END
```

Der Aufruf dieser Prozedur erfolgt durch Angabe des Namens. Durch *BEISPIEL* wird die Ausführung der in dieser Prozedur aufgeführten Befehle veranlaßt.

Als Beispiel einer Prozedur mit Variablen wird die Berechnung der Hypotenuse nach dem Lehrsatz des Pythagoras angeführt:

```
TO PYTH :A :B
PR SQRT :A*:A + :B*:B
END
```

Der Aufruf der Prozedur muß jetzt zwei Angaben, nämlich die Aktualparameter enthalten, er lautet z.B. *PYTH 3 4*.
Nun ist unbestreitbar, daß die bisherige Anwendung den Aufwand nicht lohnt. Die Flexibilität von Logo besteht aber darin, daß man die Prozedurbezeichnungen wieder in andere Prozeduren einsetzen kann. Zur Veranschaulichung soll wieder das Beispiel zur Berechnung der Hypotenuse eines rechtwinkligen Dreiecks dienen:

```
TO OBERPROGRAMM :C :D
(PR [ WERT DER ERSTEN KATHETE: ] :C)
(PR [ WERT DER ZWEITEN KATHETE: ] :D)
PR [ WERT DER HYPOTENUSE :]
PYTH :C :D
END
```

Die Ausgabe dieses Beispiels zeigt Schwächen in der Anordnung. Wer sich weiter mit Logo beschäftigt, wird leicht Abhilfe finden.

Weiterführende Literatur
Der vorliegende Artikel versuchte, in die Denkweise von Logo unter Benutzung von BASIC-Begriffen einzuführen. Er soll Anregung für die Benutzung von Fachliteratur sein. Viele Fragen mußten notgedrungen offen bleiben, z.B. die Behandlung der Kontrollstrukturen. Es gibt eine Reihe von Büchern über Logo. Wer sich nicht mit der Igelgrafik begnügen will, sollte das Buch von Senftleben: „Programmieren in Logo" zur Hand nehmen. Viele Fragen, die hier nicht einmal angeschnitten werden konnten, werden dort behandelt. Wer sich mit der deutschen Version von Logo beschäftigen will, findet in Hoppe/Löthe: „Problemlösen und Programmieren mit Logo" eine reichhaltige Sammlung von Anwendungen.
Eines wird man bei der Beschäftigung mit Logo ganz bestimmt feststellen: So wie man bei der häufigen Benutzung einer Fremdsprache beginnt, in dieser zu denken, so wird sich bei der Anwendung von Logo ein ganz neuer Programmierstil einstellen, der mit der Ausgangssprache BASIC nur noch sehr wenig gemein hat.

Puzzle mit der Maus

von Joachim Mette

Das Programm **PUZZLE** ist die Apple-II-Version des Mac-Puzzles. Es wurde sogar weiterentwickelt, so daß nicht nur auf einem 4 * 4 Feld gespielt werden kann, sondern auch auf einem 3 * 3 oder 5 * 5 Feld. Auf Grafik wurde verzichtet, damit das Programm klein und übersichtlich bleibt. Wer keine Apple-Maus besitzt, kann dieses Programm auch mit Joystick betreiben.

Leider ist das Programm recht langsam. Wer über einen Applesoft-Compiler verfügt, sollte es daher compilieren.

Sinn dieses Spiels ist es, die auf dem Spielbrett verteilten Steine in fortlaufender Numerierung anzuordnen. Dazu fährt man mit der Maus auf einen Stein, der an das Leerfeld angrenzt. Durch Klicken verschiebt sich der Stein dorthin – eine neue Lücke hat sich aufgetan. Dieses Verfahren soll so lange fortgeführt werden, bis alle Steine sortiert sind und sich die Lücke am unteren, rechten Spielfeldrand befindet.

Noch eine Bemerkung: Während der Abfrage der Spielfeldgröße wird in der Zeile 210 eine Zufallszahl erzeugt. Dies ist notwendig, da der Apple beim Einschalten immer die gleichen Zahlen erzeugt und somit immer dasselbe Puzzle erscheinen würde.
Eine Erläuterung des Programmablaufs findet sich in **Tabelle 1.**

Tabelle 1

Die Beschreibung der einzelnen Programmschritte:

```
 1ØØ- 14Ø Initialisierung des Programmes und der Apple-Maus;
 15Ø- 22Ø Abfrage der Spielfeldgröße;
 23Ø- 29Ø Spielfeld mit Zahlen besetzen;
 3ØØ- 32Ø Initialisierung der richtigen Werte für
          verschiedene Spielfeldgrößen;
 33Ø      Bildschirm löschen;
 34Ø- 36Ø Sprung in die Unterprogramme: Ausgabe Spielfeld,
          zufällige Ausgangsposition herstellen, Ausgabe der
          Feldwerte auf Bildschirm;
 37Ø- 38Ø Abfrage der Maus und Umrechnung der Mauswerte;
 39Ø- 42Ø Mausposition auf Bildschirm ausgeben;
 43Ø      Abfrage, ob Mausknopf gedrückt wurde: wenn nein,
          dann weiter bei 36Ø;
 44Ø      Abfrage, ob Feld der Mausposition Null ist: wenn
          ja, dann weiter bei 77Ø;
 45Ø- 47Ø Abfrage, ob in der Spalte ein Leerfeld ist: wenn
          ja, dann weiter bei 53Ø;
 48Ø- 5ØØ Abfrage, ob in der Zeile ein Leerfeld ist: wenn
          ja, dann weiter bei 65Ø;
 51Ø      Abfrage, ob Unterprogrammaufruf: wenn ja, dann
          zurück;
 52Ø      kein Leerfeld in Zeile oder Spalte: gehe zu 36Ø;
 53Ø      Abfrage, ob Leerfeld über Mausposition: wenn ja,
          dann weiter bei 6ØØ;
 54Ø- 57Ø herunterschieben der Puzzleteile;
 58Ø      Abfrage, ob Unterprogrammaufruf: wenn ja, dann
          zurück;
 59Ø      gehe zu 36Ø;
 6ØØ- 63Ø heraufschieben der Puzzleteile;
 64Ø      gehe zu 58Ø;
 65Ø      Abfrage, ob Leerfeld rechts von der Mausposition:
          wenn ja, dann weiter bei 72Ø;
 66Ø- 69Ø Puzzleteile nach rechts verschieben;
 7ØØ      Abfrage, ob Unterprogrammaufruf: wenn ja, zurück;
 71Ø      gehe zu 36Ø;
 72Ø- 75Ø Puzzleteile nach links verschieben;
 76Ø      gehe zu 7ØØ;
 77Ø- 81Ø Überprüfen, ob das Puzzle fertig ist:
 82Ø      wenn fertig, dann gehe zu 1Ø5Ø;
 83Ø      wenn nicht fertig, dann gehe zu 36Ø;
 84Ø- 9ØØ Ausgabe der Feldwerte auf dem Bildschirm;
 91Ø- 97Ø Ausgabe des Spielfeldes;
 98Ø-1Ø4Ø zufällige Ausgangsposition herstellen;
1Ø5Ø-1Ø7Ø abschalten der Apple-Maus;
1Ø8Ø-1Ø9Ø Frage: "Noch ein Spiel", wenn ja, dann wieder
          nach 1ØØ;
 11ØØ     Ende.
```

PUZZLE

```
 1ØØ CLEAR : HOME
 11Ø IF PEEK (5Ø188) < > 32 AND PEEK (5Ø427) < > 214 THEN
     HOME : VTAB 12: PRINT "Dieses Programm benötigt die
     Apple-Maus": END
 12Ø PRINT CHR$ (4)"PR#4": PRINT CHR$ (1)
 13Ø PRINT CHR$ (4)"PR#Ø"
 14Ø PRINT CHR$ (4)"IN#4"
 15Ø VTAB 4: HTAB 1: PRINT "Welche Spielfeld-Größe?"
 16Ø VTAB 1: INPUT "";XM,YM,KM
 17Ø VTAB 4: HTAB 28
 18Ø IF XM < 34Ø THEN PRINT "3 * 3":DR = 2
 19Ø IF XM > = 34Ø AND XM < 68Ø THEN PRINT "4 * 4":DR = 3
 2ØØ IF XM > = 68Ø THEN PRINT "5 * 5":DR = 4
 21Ø X = RND (1)
 22Ø IF ABS (KM) < > 1 AND ABS (KM) < > 2 THEN 16Ø
 23Ø FOR J = Ø TO DR
 24Ø FOR I = Ø TO DR
 25Ø L = L + 1
 26Ø A(I,J) = L
 27Ø NEXT
 28Ø NEXT
 29Ø A(DR,DR) = Ø
 3ØØ IF DR = 2 THEN AB = 345:AC = 8
 31Ø IF DR = 3 THEN AB = 257:AC = 15
 32Ø IF DR = 4 THEN AB = 2Ø6:AC = 24
 33Ø HOME
 34Ø GOSUB 91Ø
 35Ø GOSUB 98Ø
 36Ø GOSUB 84Ø
 37Ø VTAB 1: INPUT "";XM,YM,KM
 38Ø X = INT (XM / AB):Y = INT (YM / AB)
 39Ø VTAB 4 + 4 * Y: HTAB 1Ø + 5 * X: INVERSE : IF A(X,Y) <
     = 9 THEN PRINT " ";
 4ØØ IF A(X,Y) = Ø THEN PRINT " ": GOTO 42Ø
 41Ø PRINT A(X,Y)
 42Ø NORMAL
 43Ø IF ABS (KM) < > 1 AND ABS (KM) < > 2 THEN 36Ø
 44Ø IF A(X,Y) = Ø THEN 77Ø
 45Ø FOR J = Ø TO DR
 46Ø IF A(X,J) = Ø THEN 53Ø
 47Ø NEXT
 48Ø FOR I = Ø TO DR
 49Ø IF A(I,Y) = Ø THEN 65Ø
 5ØØ NEXT
 51Ø IF FL = 1 THEN RETURN
 52Ø GOTO 36Ø
 53Ø IF J > Y THEN 6ØØ
 54Ø FOR L = J TO Y
 55Ø IF L = Y THEN A(X,L) = Ø: GOTO 58Ø
 56Ø A(X,L) = A(X,L + 1)
 57Ø NEXT
 58Ø IF FL = 1 THEN RETURN
 59Ø GOTO 36Ø
 6ØØ FOR L = J TO Y STEP - 1
 61Ø IF L = Y THEN A(X,L) = Ø: GOTO 58Ø
 62Ø A(X,L) = A(X,L - 1)
 63Ø NEXT
 64Ø GOTO 58Ø
 65Ø IF I > X THEN 72Ø
 66Ø FOR L = I TO X
 67Ø IF L = X THEN A(L,Y) = Ø: GOTO 7ØØ
 68Ø A(L,Y) = A(L + 1,Y)
 69Ø NEXT
 7ØØ IF FL = 1 THEN RETURN
 71Ø GOTO 36Ø
 72Ø FOR L = I TO X STEP - 1
 73Ø IF L = X THEN A(L,Y) = Ø: GOTO 7ØØ
 74Ø A(L,Y) = A(L - 1,Y)
 75Ø NEXT
 76Ø GOTO 7ØØ
 77Ø L = Ø:M = Ø
 78Ø FOR B = Ø TO DR: FOR A = Ø TO DR
 79Ø L = L + 1
 8ØØ IF A(A,B) = L THEN M = M + 1
 81Ø NEXT : NEXT
 82Ø IF M = AC AND A(DR,DR) = Ø THEN 1Ø5Ø
 83Ø GOTO 36Ø
 84Ø FOR A = Ø TO DR
 85Ø FOR B = Ø TO DR
 86Ø VTAB 4 + 4 * A: HTAB 1Ø + 5 * B: IF A(B,A) < = 9 THEN
     PRINT " ";
 87Ø IF A(B,A) = Ø THEN PRINT "  ": GOTO 89Ø
 88Ø PRINT A(B,A)
 89Ø NEXT : NEXT
 9ØØ RETURN
 91Ø FOR A = 2 TO (4 * DR + 6) STEP 4
 92Ø VTAB A: HTAB 9: FOR B = 1 TO (5 * DR + 4): PRINT "-";:
     NEXT : NEXT
 93Ø FOR A = 3 TO (4 * DR + 5): FOR B = 8 TO (5 * DR + 13)
     STEP 5
 94Ø IF (A = 6) OR (A = 1Ø) OR (A = 14) OR (A = 18) THEN
     96Ø
 95Ø VTAB A: HTAB B: PRINT "!"
 96Ø NEXT : NEXT
 97Ø RETURN
 98Ø FL = 1
 99Ø VTAB 1: HTAB 1Ø: PRINT "einen Moment bitte"
1ØØØ FOR A = 1 TO 4ØØ
1Ø1Ø X = INT ( RND (1) * (DR + 1)):Y = INT ( RND (1) * (DR
     + 1))
1Ø2Ø GOSUB 45Ø
1Ø3Ø NEXT
1Ø4Ø FL = Ø: RETURN
1Ø5Ø PRINT CHR$ (4);"IN#Ø"
1Ø6Ø PRINT CHR$ (4);"PR#4": PRINT CHR$ (Ø)
1Ø7Ø PRINT CHR$ (4);"PR#Ø"
1Ø8Ø VTAB 1: HTAB 1Ø: PRINT "Noch ein Spiel? ";: INPUT JN$
1Ø9Ø IF LEFT$ (JN$,1) = "J" OR LEFT$ (JN$,1) = "j" THEN 1ØØ
11ØØ VTAB 22: END
```

Bei der Verwendung von Joysticks müssen folgende Zeilen geändert werden:

```
 11Ø-14Ø löschen
 16Ø XM = PDL(Ø) * 4: YM = PDL(1) * 4: KM = INT ( PEEK ( -
     16287) / 16Ø) OR INT ( PEEK ( - 16286) / 16Ø)
 22Ø IF KM < > 1 THEN 16Ø
 37Ø XM = PDL(Ø) * 4: YM = PDL(1) * 4: KM = INT ( PEEK ( -
     16287) / 16Ø) OR INT ( PEEK ( - 16286) / 16Ø)
 43Ø IF KM < > 1 THEN 36Ø
 82Ø IF M = AC AND A(DR,DR) = Ø THEN 1Ø8Ø
1Ø35 VTAB 1: CALL - 868
1Ø5Ø-1Ø7Ø löschen
```

Buchbesprechungen

Lerne BASIC auf dem Apple
Programmiertechnik für Groß und Klein
von Edward H. Carlson
1985, 322 S.,k zahlr. Abb., kart., DM 38,–
Markt & Technik, Haar
Dieses Buch – eine Übersetzung des amerikanischen „Kids and the Apple" – ist für Kinder und Jugendliche gedacht, die behutsam in die Grundlagen der Applesoft-Programmierung eingeführt werden sollen. Jeder der 33 Abschnitte gliedert sich in Lehrer-, Schüler- und Übungsteil. Aus pädagogischen Gründen wird nur eine Teilmenge der möglichen Applesoft-Befehle behandelt. Beispielsweise wird auf die HGR-Grafik überhaupt nicht eingegangen. Befehle wie LEFT$ usw. gelten bereits als „fortgeschrittenes Programmieren".
Inhalt
Teil 1: Einführung – Teil 2: Grafik (nur Lores!), Spiele und so weiter – Teil 3: Fortgeschrittenes Programmieren – Teil 4: Anhang

Mathematik auf dem Apple II, IIe, IIc
Fertige Programme, Anregungen und Erläuterungen in BASIC
von G. Daubach und D. Herrmann
1984, 219 S., kart., DM 42,–
IWT-Verlag, Vaterstetten
Dieses Buch kann allen Liebhabern ernsthafter und weniger ernsthafter mathematischer Knobeleien wärmstens empfohlen werden. Alle Probleme, auch die sehr zeitaufwendigen wie „Pi auf 1000 Stellen" usw., sind in Applesoft-BASIC gelöst, so daß Applesoft-Anhänger hier voll auf ihre Kosten kommen. Das Buch wird durch eine Fülle historischer Reproduktionen aus alten Mathematikbüchern aufgelockert, die indessen teilweise gute Lateinkenntnisse voraussetzen.
Inhalt
Einleitung – Mehr-Register-Arithmetik – Zahlentheorie – Kombinatorik – Algebra – Geometrie – Numerische Mathematik – Anhang

Trainingsbuch zu APPLESOFT-BASIC
von Dr. Renate Prust
1984, 350 S., kart., DM 39,–
Data-Becker, Düsseldorf
Dieses „Trainingsbuch" wendet sich an „blutige" Anfänger, die ihren Apple gerade erworben haben und noch keinen anderen BASIC-Dialekt kennen. Jedes der extrem einfachen Programme wird sehr umfangreich kommentiert, so daß für viele grundlegende Befehle wie GOSUB, HTAB, VTAB usw. kein Platz mehr bleibt; diese Befehle werden dann in dem Kapitel „Weitere APPLESOFT-Sprachelemente" ohne Beispiele kurz erwähnt. Leider ist dieses Buch wie viele andere Data-Becker-Bücher in Schreibmaschine „gesetzt", wobei sogar auf manchen Seiten mit dem Kuli nachgeholfen wurden (z.B. S. 38). Gerade ein Anfängerbuch sollte typographisch besser aufbereitet werden.
Inhalt
Grundlagen – Die wichtigsten Befehle – Schleifen – Wiederholung von Programmteilen – Indizierte Variablen – Weitere APPLESOFT-Sprachelemente – Fehlermeldungen – Schlüsselwörter – Lösungen der Aufgaben

APPLE II für Technik und Wissenschaft
von Rolf Drewes
1984, 268 S., kart., DM 49,-
Data-Becker, Düsseldorf
Dieses Buch enthält eine Reihe mathematischer, chemischer und physikalischer Programme, die allesamt in Applesoft geschrieben sind. Die Programmbeschreibungen sind knapp, aber ausreichend. Technische Zeichnungen sind mit dem Kuli erledigt worden (s.S. 250 usw.). Offenbar wurde das komplette Manuskript unverändert abfotografiert. Dafür sind 49,- DM als Ladenpreis zu viel.
Inhalt
Einleitung – Programmiersprachen und Programmierung – Ein- und Ausgabe – Tastatur, Bildschirm, Floppy – Wichtigste Naturkonstanten – Programme aus der Mathematik – Programme zu grafischen Darstellungen – Programme zur Datenverarbeitung – Programme aus der Chemie – Programme aus der Physik und Technik – Spiel – Atomphysik

Macintosh
Anwenderhandbuch
von Charles B. Duff
1984, 156 S., kart., DM 39,80
McGraw-Hill, Hamburg
Bei dieser deutschen Übersetzung des amerikanischen Titels „Introducing the Macintosh" handelt es sich wahrscheinlich um das erste Buch, das im letzten Jahr zum Macintosh erschien und das nach wie vor als Kurzeinführung für Mac-Anwender, die sich einen groben Überblick verschaffen wollen, empfohlen werden kann. Allerdings sind zu den meisten beschriebenen Programmen inzwischen neuere Releases erschienen. Interessant ist der Vergleich zwischen Macintosh und IBM-PC ab S. 145ff., wobei übrigens das Buch selbst nicht auf dem Macintosh, sondern auf dem IBM-PC geschrieben wurde.
Inhalt
Der Macintosh stellt sich vor – Das Innenleben des Macintosh – Die Anwenderprogramme des Macintosh – Die Toolbox – und wie sie sich erweitern läßt

Macintosh
Ein Computer und seine Mitwelt
von Key B. Hacker
1984, 195 S., kart., DM 48,-
Vieweg & Sohn, Braunschweig
Dies ist eines der umstrittensten Bücher über den Macintosh. Manfred Schürmann alias „Keyboard Hacker" kann eindringlich und anregend schreiben, daran besteht kein Zweifel. In Abwandlung von Goethes Ausspruch muß man jedoch sagen: „Das ‚wie' bedenke! Mehr bedenke ‚was'!" So schreibt Schürmann wiederholt in eloquenten Tiraden, daß ihn beispielsweise der langsame Diskettenzugriff stört; eine Begründung wird jedoch nicht gegeben. So ist dieses Buch weniger für Programmierer und Techniker denn für all diejenigen gedacht, die den Macintosh psychologisch hinterfragen wollen. Interessant ist außerdem das Kapitel über die „Meinungsmache", in dem aufgewiesen wird, daß deutsche Zeitschriften bereits zu einem Zeitpunkt Lobpreisungen über den Macintosh verbreiteten, als das Gerät noch keinem vorlag. Ein für Insider lesenswertes Buch.
Inhalt
MAC's Welt, in der wir leben – How to use – Was man Schwarz auf Weiß besitzt – Ein „verunglücktes" Titelbild – Sag' mir, was du meinst – Die vergessene Kapitelüberschrift – Schau, was kommt von draußen rein – Meinungsmache über den Mac – Persönliche Beichte – Bilder statt Worte

Das Apple Macintosh Buch
von Josef Steiner
1985, 359 S., zahlr. Abb., kart., DM 52,-
Markt & Technik, Haar
Dieses Buch wendet sich überwiegend an Anwender und weniger an Programmierer. Dementsprechend liegt das Schwergewicht auf der Handhabung von Anwenderprogrammen wie Macwrite, Macpaint, Multiplan usw. Einige wenige Macintosh-Interna sind auf den letzten Seiten erwähnt. Leider ist das Buch wie die meisten anderen Markt-&-Technik-Bücher mit der Schreibmaschine „gesetzt", was nicht gerade der modernen, elektronischen Satztechnik entspricht.
Inhalt
Einführung – Die Bedienung des Macintosh – Auspacken und Aufstellen – Der Macintosh-Schreibtisch – Die Schreibtischutensilien – Arbeiten mit Dokumenten – Arbeiten mit Disketten – Kommandoabkürzungen und Drucken – Anwendungsprogramme für den Macintosh – Texte verarbeiten – Zeichnen, Malen und Gestalten – Rechnen und Kalkulieren – Geschäftsgrafiken erstellen – Daten verwalten und auswerten – Kommunikation – Programmieren mit dem Macintosh – BASIC – Pascal – FORTH – Zusatz-Hardware – Informationen für Spezialisten – Die Systemsoftware des Macintosh – Das Innere des Macintosh – Anhang – Stichwortverzeichnis

Leserbriefe

ProDOS-CONVERT

Beim Gebrauch des ProDOS-CONVERT-Programms in Verbindung mit einem Duo-Disk-Laufwerk tritt häufig eine unangenehme Tatsache zutage. Man startet das Convert-Programm in Laufwerk 1 und legt die DOS-3.3-Diskette in Laufwerk 2. Nach der korrekten Einstellung der Werte meldet sich das Programm beim Aufruf des Tranfers von DOS 3.3, S6, D2 nach ProDOS mit „NO DEVICE CONNECTED" für Laufwerk 2. Es bleibt nichts anderes übrig, als die Disketten zu vertauschen und dem Programm mitzuteilen, daß sich die DOS-3.3-Diskette in Laufwerk 1 befindet. Ich hatte die Möglichkeit, auf verschiedenen Gerätezusammenstellungen diesen Sachverhalt zu testen. Auf einem Apple II+ (64K) und auf einem Apple IIe, jeweils mit zwei Disk-II-Laufwerken bestückt, gabe es diese Probleme nicht. Hingegen auf einem Apple IIe mit Duo-Disk-Laufwerken sowie auf einem Apple IIc mit zweitem Laufwerk treten diese Probleme auf. Hiermit scheint sich Apple mal wieder ins eigene Fleisch geschnitten zu haben.

Jörg Wunschhofer, Beckum

Anm.d.Red.: Das CONVERT-Problem ist bekannt und tritt mit Gewißheit dann auf, wenn sich der Lesekopf im Laufwerk 2 auf der innersten Spur 34 befindet, was z.B. dann der Fall ist, wenn man zuvor unter DOS 3.3 eine Diskette mit einem Programm kopiert hat, das von Spur 0 nach Spur 34 aufwärts kopiert. Wenn man sich nicht an der CONVERT-Fehlermeldung stört und den Konvertierungsvorgang exakt dreimal zu starten versucht, dann zieht ProDOS den Lesekopf wieder zurück, und weitere Konvertierungsvorgänge funktionieren normal. Das Problem liegt genauer gesagt weniger am CONVERT denn am ProDOS selbst. Unser eigenes Konvertierungsprogramm DOSTOPRO aus „ProDOS für Aufsteiger", Bd. 2, S. 187 umgeht das Problem, indem es vor dem eigentlichen Einlesevorgang der DOS-Datei dreimal Block 0 zu lesen versucht. Dies überlistet den ProDOS-Bug. CONVERT hat übrigens noch andere Bugs; z.B. darf ein DOS-Dateiname i.d.R. keine Leertasten enthalten, da sonst ebenfalls ein I/O-Fehler angezeigt wird. Ferner kann man den DOS-Namen aus CONVERT heraus nicht auf 15 Zeichen kürzen, so daß zwei Dateien, deren Namen in den ersten 15 Zeichen identisch sind, nicht hintereinander kopiert werden können. All diese Bugs sind in unserem DOSTOPRO nicht „implementiert". us

Basis 108

Der positiven Meinung von Tim Berndt aus Büdelsdorf (Peeker 7/85, S. 73) kann ich mich nur anschließen. Auch meine Erfahrungen mit der Fa. Basis in Münster waren bisher äußerst erfreulich. Anfragen wurden immer beantwortet, und man war bemüht, mir stets weiterzuhelfen.

Seit 1983 bin ich im Besitz eines Basis 108 mit CP/M+ (3.0). Unterdessen ist noch die 256K-BASRAM-Karte (Pseudo-Floppy) von der Fa. Basis dazugekommen, die einwandfrei funktioniert.

Zwei kleinere Probleme konnte ich noch nicht eliminieren, zu denen vielleicht ein BASIS-Besitzer die Lösung gefunden hat.

1. Es war mir bisher nicht möglich, Wordstar im Pseudo-Floppy C: einzusetzen. Ein Aufruf zur Bearbeitung von Files auf den vorhandenen Laufwerken A: oder B: führte jedesmal zum Absturz.
2. Da der Basis 108 z.B. das Löschen einer Zeile mit ESC M, Löschen des Bildschirmes mit ESC *, Löschen bis Seitenende mit ESC Y und vieles mehr mit veränderbaren Bildschirmfunktionen ermöglicht, ist ein Arbeiten mit einem angepaßten „Wordstar" oder „Turbo Pascal" sehr komfortabel. So läßt sich „Wordstar" bei 026Dh:02 1B 54 zum „erase to end of line" und bei 0274h:02 1B 4D zum „delete screen line" bewegen. Leider funktioniert das Einfügen einer Zeile mit ESC L nicht, wenn sich der Cursor auf der letzten Zeile befindet. Dasselbe gilt für „Turbo-Pascal". Wie mir die Fa. Basis mitteilte, liegt ein Fehler im CP/M+ 3.0 vor, an deren Behebung gearbeitet wird.

Vielleicht haben noch andere BASIS-108-Besitzer das Bedürfnis, Anregungen weiterzugeben oder Kontakte zu anderen BASIS-108-Besitzer zu knüpfen. Sofern ein Bedürfnis besteht, würde ich mich gerne als Kontaktadresse zur Verfügung stellen.

Rolf Gachnang,
Neue Jonastr. 81,
CH-8640 Rapperswil,
Schweiz

Erfahrungen mit Macintosh

Ich besitze seit ca. einem Vierteljahr einen Macintosh und möchte auf einige Fehler der Systemsoftware aufmerksam machen, die meines Wissens noch nicht allgemein bekannt sind, einem neuen Benutzer aber einiges Kopfzerbrechen machen können. Nachdem ich fünf Jahre lang der zufriedene Besitzer eines Apple II Plus gewesen bin, bin ich von der minderen Qualität der Dokumentation (es steht eigentlich überhaupt nichts in Handbüchern) und Software (deutsche Version) dieses Apple-Produktes höchst unangenehm überrascht, obwohl ich dennoch die Maschine nicht so generell verdammen möchte wie Herr Stiehl: Ich hoffe noch...

Von vielen frühen deutschsprachigen Macintosh-Benutzern beklagt und inzwischen anscheinend von Apple berichtigt, hatte die Schreibtischdatei, welche die gesamte Pseudo-Subdirectory-Organisation enthält, ursprünglich im deutschen System den Namen „Schreibtisch". Im amerikanischen System heißt sie „Desktop". Offensichtlich hat hier ein Mitarbeiter des Hauses Apple, München bei der Übersetzung der Finder-Resource-Datei ein wenig zuviel des Guten getan! Will man eine Diskette des einen Systems von dem jeweils anderen System aus lesen (Catalog genügt), so wird dabei die Organisation der Dateien in „Ordner" (Subdirectories) vollkommen zerstört. Interessanterweise liefert aber Apple nach wie vor auch die fehlerhafte Version des Finders aus! Bei meinem im April gekauften Macintosh befindet sich auf der Diskette „Write:" ein Finder (1.1g), der seine Schreibtischorganisation in einer Datei mit Namen „Schreibtisch" ablegt, auf der Diskette „Paint:" aber ein anderer Finder (auch Version 1.1g), der statt dessen eine Schreibtischdatei namens „Desktop" erstellt. Da die Schreibtischdateien das Attribut „unsichtbar" haben, ist ein unerfahrener Benutzer hier eventuell äußerst frustrierenden Erlebnissen ausgesetzt, die sein Vergnügen mit dem Macintosh nachhaltig schmälern könnten.

Apple liefert an seine Händler ein sog. „Localizer"-Programm, das es erlaubt, in eine ausländische Systemdatei die deutsche Tastaturbelegung zu installieren. (Warum Apple nicht einfach einen anderen Decoder in die deutschen Tastaturen einsetzt, ist sowieso ein Rätsel). Dieses Programm ändert auch die INTL-Resource in der Systemdatei. Dabei werden noch ein paar mehr Dinge als die Tastaturbelegung verändert, z.B. auch die Währungs- und Datenformate. Leider wird hierbei der sechste Wochentag als „Sonnabend" und nicht als „Samstag" initialisiert. Der Finder verwendet aber die ersten drei Buchstaben des Wochentags als Abkürzung mit dem Effekt, daß der Sonntag nicht mehr vom Samstag unterscheidbar wird. Wirklich traurig, daß einer Firma wie Apple so etwas passieren muß! Überhaupt ist es sehr fraglich, ob der Benutzer eines ansonsten englischen Systems wirklich den Wochentag und die Uhrzeit „...UHR" ausgerechnet auf deutsch sehen will. Mich persönlich ärgert das.

Das bei Macintosh-Pascal angewandte Kopierschutzverfahren ist zumindest fragwürdig zu nennen. Ich habe bisher bei meinen übrigen Computern nie mit Originalprogrammen gearbeitet, sondern stets Sicherungskopien angeferigt, wozu ich nach dem Urheberrecht berechtigt bin. Dieses auch dann, wenn herstellerseits eine zweite Diskette geliefert wurde, weil ich einfach ein unangenehmes Gefühl bei der Benutzung von Originaldisketten habe. Die Originaldisketten befinden sich dabei aber in meinem persönlichen Besitz. Nun scheint aber Macintosh-Pascal auch bei einer mit einem Nibble-Kopierprogramm angefertigten Kopie zu merken, daß es nicht von der Originaldiskette läuft (Prolok?), und retourniert den überraschten Benutzer nach einer gewissen Zeit unter Datenverlust in den Finder! Deshalb ist Macintosh-Pascal auch nicht auf einer Harddisk installierbar und läuft nicht von einer RAM-DISK aus. Diese Methode des Programmschutzes ist meiner Meinung nach höchst unfein, zumal es bei Macintosh-Pascal bis zu einer Stunde dauern kann, bevor der Schutzmechanismus aktiv wird, und die Datenverluste erheblich sein können. Ich gehe davon aus, daß der überwiegende Teil der Personen, die mit Programmkopien arbeiten, auch das Originalprogramm gekauft haben, und fühle mich von der Firma Apple irgendwie betrogen!

Weiterhin weist die mir gelieferte Version von MacWrite (2.20) einen empfindlichen Mangel auf: Entgegen der Beschreibung in der mitgelieferten Dokumentation (S. 39 ff.) ist es möglich, bei hängendem Einzug die Absätze zu numerieren. Zwar erscheinen die Zahlen an der richtigen Stelle auf dem Bildschirm, aber beim Ausdruck erlebt man dann ein kleines Wunder: Was auf dem Drucker erscheint, ist nicht identisch mit dem Schirmbild! Die Zahlen fehlen. Hier liegt ein schwerer Verstoß gegen die Macintosh-Philosophie vor „What you see is what you get"!

Traurig ist auch, daß diverse Programme, wie z.B. Microsoft-Word, anscheinend nicht die von Apple definierten Ein/Ausgaberoutinen verwenden, sondern die Tastatur direkt lesen (evtl. sogar notgedrungen; meine Version von RMaker erlaubt z.B. nur Groß- und Kleinbuchstaben als Befehlstastenäquivalent für Menükommandos). So darf man als Benutzer der deutschen Tastatur dann rätseln, welcher Buchstabe wohl auf der amerikanischen Tastatur durch „option-square open bracket" generiert wird, wenn ein Programm diese Tasten plus der Kommandotaste gleichzeitig sehen will. Aber sogar Programme aus dem Hause Apple selbst fallen unangenehm auf: Die MAUG-ß-Prerelease-Version von Andy Hertzfelds „Switcher" hat die Kommandosequenzen Command-„(","),"„/", was auf einer deutschen Tastatur zu „ü","+" und „Return" wird.

Es gibt noch viele weitere Kleinigkeiten, die mich an meinem Macintosh ärgern, aber das sind die wichtigsten. Trotzdem möchte ich mich (noch) nicht von ihm trennen, denn bis jetzt habe ich die Hoffnung nicht aufgegeben, daß eines fernen Tages wirklich fehlerfreie und schnelle Software kommt, die die Fähigkeiten der Maschine voll ausnutzt. Das Potential ist vorhanden. Eigentlich müßte zumindest LisaPascal bald auf dem Macintosh laufen, wo es doch den Macintosh XL nun nicht mehr gibt.

Schlußendlich möchte ich Ihnen zu Ihrem überaus gelungenen Magazin herzlich gratulieren! Allerdings glaube ich in den beiden letzten Ausgaben eine gewisse Kommerzialisierung beobachten zu können. Auf zwei Seiten ein Spiel („Pyramid Pitty") zu beschreiben, das dann nur mit einer Peeker-Sammeldiskette erworben werden kann, ist schon reichlich unverschämt. Sogar in Ihrer ausländischen Konkurrenz „Nibble" werden solche Beiträge ausdrücklich als Anzeige deklariert. Bitte behalten Sie den Stil Ihrer ersten Ausgaben bei! Es gibt schon genügend „Publikumszeitschriften mit EDV-Charakter" auf dem Markt!

Michael Franz, CH-Zürich
Amn.d.Red.: Zu „Pyramid-Pitty" s. Heft 8/85, S.68. us

Wordstarpatch für Epson FX-80

Dieser Beitrag (aus Heft 7/85 und 8/85) ist eine wertvolle Hilfe, die ich sofort ausprobiert habe. Ich möchte Sie aber auf eine Schwierigkeit aufmerksam machen, die mir einiges Kopfzerbrechen bereitet hat.

Bei der verwendeten Druckerinit-Sequenz unter $0E67 (PSINIT) schaltete der WORDSTAR den FX-80 in den französischen Zeichensatz um! Abhilfe brachte erst die Verkürzung auf: 02 1B

40, wonach nach dem Zählbyte (02) einfach ein ESC § (= Klammeraffe ASCII 64) gesetzt wird. Nach meinen Erfahrungen reicht das völlig aus, um den Drucker zu normieren.
Ich empfehle übrigens, bei der „Patcherei" auch gleich die meist lästige Seitennumerierung auszuschalten, so daß nicht immer wieder vor den Dateien ein .op eingegeben werden muß.
Label ITPOPN ($03D3): 00 = EIN, FF = AUS. Auch das bezieht sich auf die Version 3.0.
Dieter Charchot, Hamburg

Pascal-Directory
Die Abfrage für die Gültigkeit eines Pascal-Directory (s. Heft 1/85, S. 64ff.) lautet richtig geklammert:
(dFirstB1k=0) AND
(MiscInfo.UserKind=Booker)
OR ((MiscInfo.UserKind IN [AQuiz, PQuiz]) AND dFKind=SecureDir))
OR ((MiscInfo.UserKind=Normal) AND (dFKind=UntypedFile))
AQuiz heiß meiner Meinung nach „awkwards".
Ansonsten ein guter Artikel.
Edgar Fuss, Bonn

Erfahrungen mit Balfer-Interface
Zu dem Leserbrief von Herrn H.-D. Siewert im Peeker, 6/85 (s.a. 5/85, S. 12ff.) möchte ich hier gerne meine subjektiven Erfahrungen mit der Balfer-Karte schildern. Ich habe mir diese Karte bestellt, da ich sowieso eine universelle E/A-Karte benötigte und natürlich erfreut war, daß der Peeker zu dieser Karte auch noch weitere Utilities veröffentlicht. Ich habe mich für die Fertigversion entschieden, die auch gleich ein paar Tage nach der Bestellung geliefert wurde. Die Karte wird gut verpackt in Alufolie und Luftpolsterfolie verschickt, die Platine ist sauber geätzt und gelötet (im Gegensatz zu anderen Karten, die ich schon gesehen habe). Auf der Karte befindet sich ein 2K-RAM, das den Slotbereich Cn00-CnFF sowie den Bereich C800-CEFF abdeckt und durch ein 2K-EPROM ersetzt werden kann, außerdem der E/A-Baustein, der VIA 6522. Der (fertigen) Karte war noch eine Bauanleitung und ein kleines Testprogramm für die Funktion des VIA beigelegt, die man dann mit dem Oszillographen überprüfen kann. Diese beiden DIN-A4-Blätter waren jedoch die ganze Information, die mit der Karte kam! Über die Programmierung des sehr universellen VIA-6522-Bausteins wird man im unklaren gelassen. Ich finde, bei einer so universell zu gebrauchenden Karte sollte man auch ein wenig über die Programmiermöglichkeiten informiert werden. Ich kaufte mir also das Buch „6502 Anwendungen" von R. Zaks, in dem die E/A-Bausteine des 6502 sehr genau und auch mit einigen Beispielprogrammen versehen, beschrieben sind. Zusammen mit diesem Buch, das allerdings mit weiteren 38,- DM zu Buche schlägt, kann man die vielseitigen Möglichkeiten des Interfaces voll ausnutzen. Man kann sich Hardwarezusätze bauen, die Karte als serielle und parallele Schnittstelle programmieren oder sie auch für Interruptsteuerungen benutzen.
Auch in „MC", 4/83 befindet sich ein universelles E/A-Programm, wie die Fa. Balfer in einem Info mitteilte. Dieses Programm braucht man nur noch, um die speziellen Treiber für seriell oder parallel zu ergänzen. Mit diesen beiden Zusatzquellen läßt sich die Karte sehr gut nutzen, doch man muß von vornherein die zusätzliche Ausgabe von ca. 45,- DM einkalkulieren. In diesem Fall finde ich, ist die Karte durchaus lohnenswert, da man sie – mit entsprechenden Kenntnissen aus dem Buch – sehr universell als Drucker- und Modeminterface programmieren kann.
Frieder Mahr, Esslingen

Softswitches
Die bisherige Arbeit der Peeker-Redaktion finde ich sehr gut. Vielleicht könnten Sie mal Artikel bringen, die in anderen Zeitschriften sowieso tabu sind und auch in der Fachliteratur nur vereinzelt beschrieben werden. Dazu gehören z.B. eine Aufstellung der Softswitches mit Erkärung, ähnlich wie die Aufstellung der Assembler-Pseudo-Opcodes in der letzten Ausgabe. Ein weiteres Thema , was extrem zeitkritisch und deshalb schwierig ist, ist das Lesen und Schreiben einer Diskette auf unterster Ebene. So ist in dem Buch „Apple Assembler – Tips und Tricks", Seite 68, ein kurzes Programm zur Prüfung des Schreibschutzes; weitere Erklärungen fehlen jedoch. Für Bastler, zu denen ich auch gehöre, wäre es sicherlich schön, wenn Sie einmal Hardwareänderungen oder -Erweiterungen vorstellen könnten. Ein weiterer Vorschlag wäre ein Lehrgang über die logischen Bausteine des Apple mit einer kurzen Erklärung.
Robert Anmond, Trier
Anm.d.Red.: Das Ihnen bereits vorliegende „Apple Assembler" enthält S. 82-166 und 199-226 nur „Softswitch"-Programme. Wir werden trotzdem für den Peeker einige (leichtere) Softswitch-Übungsprogramme einplanen. us

Pascal-Idsearch
Ich hätte zu dem Artikel „Die versteckte Prozedur Idsearch" von Dieter Geiß im Peeker, Heft 8/85 etwas zu bemerken. Ich finde die Idee, etwas tiefer in die Arbeitsweise von Apple-Pascal einzudringen, sehr gut, und ich freue mich auf die weiteren Artikel in dieser Serie.
Aber in diesem ersten Teil befindet sich meiner Meinung nach ein grober Verstoß gegen die Übersichtlichkeit, die sonst in der Programmiersprache Pascal vorhanden ist. Es geht mir da um die beschriebenen Seiteneffekte bei Verwendung der Prozedur „Idsearch". Es wird nur die Variable „Symcursor" in der Parameterliste genannt, aber die Variablen „sym", „OP" und „Id" werden auch verändert.
Da der Compiler bei dieser Prozedur nicht den Typ der Parameter kontrolliert und im P-Code bei der Übergabe eines Record an eine Prozedur nur die Adresse der ersten Variablen übergeben wird, kann man das Programm durch Verwendung einer Variablen vom Typ Record viel übersichtlicher gestalten.
type Id_Typ = record
Symcursor: integer;
Sym: integer;
Op: integer;
Id: Alpha
end;
var Idrecord: Id_Typ;
Der Aufruf der Prozedur wäre dann also:
Idsearch (Idrecord, S);
Wie gesagt, im P-Code ändert sich nichts, diese Änderung dient nur der Übersichtlichkeit des Quelltextes.
Noch etwas zur Arbeit des Compilers: Schreibt man in der obigen Typendeklaration die Integer-Variablen nur durch ein Komma getrennt, so wird deren Reihenfolge durch den Compiler umgekehrt. Möchte man also Kommata verwenden, so hätte man zu deklarieren
type Id_Type = record
Op, Sym, Symcursor: integer;
Id: Alpha
end;
Ansonsten war der Artikel sehr gut, und ich hoffe auf weitere Beiträge, die auch auf den P-Code eingehen.
Armin Köllner, Bochum

Diskettenprobleme
Als engagierter „Leser der ersten Stunde" habe ich einige Probleme mit dem Peeker, die ich mit dem Wunsch nach Lösung hiermit vortragen möchte:
1. Das ProDOS-Patch-Programm aus Heft 1/2 85 versagt seinen Dienst. Bei allen ProDOs-Versionen (1.0, 1.0.1 und 1.1.1) meldet es sich nach dem Patch-Versuch in Zeile 399 mit der Meldung „Fehler – nicht die erwarteten Daten!".
Als Abhilfe würde ich vorschlagen, die Spuren und Sektoren für die Patch-Eingriffe mit den entsprechenden Hex-Codes (original und geändert) zu veröffentlichen, um die Änderungen per Disk-Editor vornehmen zu können. (*Anm.d.Red.: In Zeile 570 sollte es heißen CMD = 2: GOSUB 800. us*)
2. Daß Probleme mit dem Diversi-DOS 2-C von der Sammeldiskette #6 auftreten können, haben Sie ja schon in Heft 8/85 angedeutet. Ich würde aber trotzdem gerne erfahren, wie ich die Programme HELLO und ASMDIV benutzen kann, ohne daß sich nach dem Hello-Start die Aufforderung meldet, die Diversi-Diskette erneut zu booten. Die beiden Programme verlangen augenscheinlich bereits ein original Diversi-DOS im Speicher, das ja bislang nicht vorliegen kann. (*Anm.d.Red.: Siehe nächsten Leserbrief. us*)
3. Mit den Peeker-Sammeldisketten gibt es leider Probleme. Offensichtlich bedingt durch unpraktische Verpackung und den rauen Postversand sind die Diskettenhüllen so stark an den Kanten zusammengedrückt, daß sich die Disketten selbst durch das Laufwerk in der Hülle nur noch schwer drehen lassen. Meist hilft dann nur noch das Aufschneiden der Hülle und nach vorsichtiger Entnahme der Scheibe deren sorgfältiges Aufbiegen. Auch hier bitte ich um eine Problemlösung. (*Anm.d.Red.: In solchen Fällen erfolgt kostenloser Umtausch. Wir haben inzwischen den Hersteller BASF gewechselt. us*)
Wolfgang Geisendörfer, Meppen

Wichtig, wichtig, wichtig!
Nachfolgenden Brief lesen!

Diversi-DOS 2-C
Es ist festzustellen, daß das Diversi-DOS 2-C von der Sammeldiskette #6 lauffähig ist, wenn man einen Trick anwendet.

Schritt 1:
DOS 3.3 booten
10 PRINT „Hallo"
INIT START
Schritt 2:
Auf diese neu initialisierte DOS 3.3-Diskette die Dateien HELLO und ASMDIV von der Sammeldiskette #6 kopieren.
Schritt 3:
POKE 40312,32
RUN HELLO
Option 8: BSAVE Patch File
Option 9: Exit to Basic
Schritt 4:
Diskette neu booten
NEW
BRUN PATCH
CALL -151
BFD6: 4C 44 B7
RUN HELLO
(Achtung: CALL -151, BFD6: 4C 44 B7 gilt nicht für die ursprüngliche System-Master 1980, sondern nur für die etwas neuere System-Master 1982. Anm. d. Red.)
Schritt 5:
Option 4: Make Copies of this Disk
Option 9: Exit to Basic

Und als Ergebnis dieser Bemühungen hält man eine bootfähige Diversi-DOS 2-C Master-Disk in Händen. Allerdings hat es bei mir mehrere Tage gedauert, bis ich diese Lösung fand.
Arndt Kay Marczoch, Münster
Anm.d.Red.: Herzlichen Dank, Herr Marczoch! Keiner hat an diesen verblüffend einfachen Trick gedacht, und ich schon gar nicht... Anhand der Originaldiskette kann ich bestätigen, daß mit dem geschilderten Verfahren tatsächlich eine lupenreine 2C-Version entsteht. Trotzdem bitte die $30.00 an DSR überweisen. us

Biteditor
Nachdem ich Peeker nun seit der zweiten Ausgabe (die erste war leider nicht mehr zu bekommen) mit großem Interesse lese, habe ich mich sehr über die Veröffentlichung des Biteditors gefreut. Da die Fa. Jeschke, Kelkheim auch nach mehreren Anfragen trotz ihres Angebotes, für ihre 80-Zeichen-Karte einen deutschen Zeichensatz zu liefern offenbar nicht in der Lage ist, konnte ich mit Hilfe Ihres schönen Programms endlich die fehlenden Zeichen erstellen.

Dazu ein Tip: Da bei der 80-Zeichen-Darstellung die horizontalen Teile der Zeichen bei den billigeren Monitoren die vertikalen Anteile fast überstrahlen, kann man auch anstatt der doppelten vertikalen Linien die horizontalen Linien mit Zwischenräumen versehen. Dadurch ergibt sich eine gestochen scharfe Abbildung der Zeichen als Punktmatrix. Nachteile ergeben sich in der inversen Darstellung. Da ich die 80-Zeichen-Karte hauptsächlich für die Textverarbeitung mit dem Applewriter benutze, ist das für mich kein Nachteil.
Beispiel Buchstabe „B":

```
*-*-*
*---*
*---*
*-*-*
*---*
*---*
*-*-*
```

Detlev Gerrets, Jevenstedt

Stiftung Warentest
Zunächst möchte ich Ihnen meine Begeisterung für den Peeker mitteilen, den ich seit der ersten Nummer abonniert habe. Mir gefällt die Mischung von Programmen und Artikeln, auch wenn sie für mich als Anfänger häufig ein so hohes Niveau haben, daß ich manches (noch) nicht verstehe. Hier wäre mein Wunsch, daß noch mehr einführende Artikel, vielleicht sogar mit Übungsbeispielen im Heft oder auf der Sammeldiskette kommen. Dabei halte ich die Blockform (wie bereits angekündigt) für geeigneter als eine Serie, die sich über 5 oder sogar 10 Hefte hinzieht.
Geradezu wohltuend ist die kritische Distanz zur Firma Apple oder zu anderen Anbietern, die für mich den Peeker zu einer ähnlichen Instanz wie die Zeitschrift „TEST" von der Stiftung Warentest macht, der ich mich bei Kaufentscheidungen eher anvertraue als wohlklingenden Werbeversprechen oder Prospekten. Insofern möchte ich Sie ermuntern, Testberichte so objektiv und kritisch wie möglich zu gestalten und neben den Vorzügen eines Produkts oder Buches o.ä. gerade die Schwachpunkte auch darzustellen. Laien wie ich bemerken diese ja häufig erst, wenn alles zu spät und das Geld ausgegeben ist. Außerdem wirkt eine kritische Beschreibung als „sanfter Druck" auf den Anbieter, sein Produkt zu optimieren, wovon unterm Strich dann alle etwas haben.
Die Zeitschrift „Apple's" hat für mich in vieler Hinsicht diesen Kredit ebenso verspielt wie zahlreiche andere Computer-Zeitschriften, bei deren Lektüre ich mich häufig des Eindrucks nicht erwehren kann, daß die Artikel direkt aus den Marketing-Abteilungen aufs Papier gehen, nicht notwendigerweise zum Nutzen der Endverbraucher.
Ich habe gehört, daß der Apple IIe jetzt mit verändertem ROM hergestellt wird und daß es für Apple IIe's mit altem ROM Nachbausätze zum Preis von etwa DM 300,- geben soll. Vielleicht könnte das mit Für und und Wider mal in einem Artikel besprochen werden (*Anm.d.Red.: Siehe dieses Heft. us*).

Da ich den Peeker als Arbeits- und Informationsmittel nutze (und das auch in den nächsten Jahren tun möchte), hätte ich folgende Wünsche, die zumindest mir das formale Arbeiten erleichtern würden:
1. Ein Jahres-Inhaltsverzeichnis, in dem nach Autoren, Artikeln und Sachgebieten alle Beiträge einschließlich der Errata und späteren Ergänzungen (auch abgedruckte Leserbriefe, die ja nicht im Inhaltsverzeichnis erscheinen) geordnet sind. Optimal wäre es als Hardcopy und als Diskette. Vielleicht bringt der Peeker ja nochmal ein Sortierprogramm dazu, das den „Zugriff" weiter erleichtert (*Anm.d.Red.: Kommt im Dezember-Heft. us*)
2. Sammelordner mit Stabmechanik, damit die Hefte jahrgangsweise abgelegt werden können.
3. Bei den Peeker-Artikeln würde ich mir standardmäßig kurze Zusammenfassungen am Anfang wünschen, um entscheiden zu können, ob für mich das Durcharbeiten in Frage kommt oder nicht. Diese in Fachzeitschriften üblichen „Summaries" sind bezüglich schneller Orientierung sehr hilfreich.
Gunter Kase, Hamburg

Übertragungsrate Megaboard 22K/s
Wenn eine neue Zeitschrift erscheint, ist die erste Ausgabe meist sehr gut, aber dann geht es mehr oder weniger schnell abwärts. Dies ist beim Peeker nicht der Fall. Es wurden sogar manche Details verbessert. Zuerst gab es gar kein Inserentenverzeichnis, dann war es irgendwo versteckt. Jetzt befindet es sich am Ende der Zeitschrift, wo es sofort gefunden werden kann. Der Peeker ist also nach wie vor hervorragend, und das meist recht hohe Niveau der Artikel sollte beibehalten werden. Die Sonderteile in Form eines „Mini-Buches" bringen auch für Fortgeschrittene Vorteile. Man muß nämlich nicht viele Hefte absuchen, wenn man etwas nachschlagen möchte.
Ich bin kein Informatik-Lehrer und finde Ihre Bewertung der Länge des Sourcefiles im Pascal-Wettbewerb absurd. So etwas gehört in die Steinzeit der Computertechnik, als die Programme um jeden Preis kurz gehalten werden mußten, weil die Hardware noch teuer war. Der Programmiersprache Pascal sollte man so etwas nicht antun.
Im Artikel „Die AP33-Megawrap RAM-Karte" in Heft 7/85 baten Sie um Meßergebnisse mit der Profile-Festplatte. Da ich zwar keine Profile, dafür aber eine Festplatte von BASF und den Megaboard-Controller von Frank & Britting besitze, habe ich damit die Übertragungsgeschwindigkeit gemessen. Die Übertragungsdauer des SPEEDTEST beträgt bei Read und Write 46.3s. Hieraus ergibt sich eine Übertragungsgeschwindigkeit von 22 K/s.
Zu Ihrem Test „MEGACORE" im selben Heft möchte ich folgendes ergänzen: Die Übertragungsrate auf Systemebene habe ich mit dem beigelegten Programm ermittelt. Sie beträgt bei Read und Write ca. 49000 Bytes/s.
Da Sie mit Ihrer Behauptung, ein 140K-Laufwerk sei wie der Kofferraum des alten VW-Käfer, vollkommen recht haben, ist die Kompatibilität zu 640K-Laufwerken interessant. Da die 640K-Laufwerke kaum teurer sind als 140K-Laufwerke, haben Sie sich zu einem zweiten Standard entwickelt. Deshalb wäre es sinnvoll, bei solchen Tests die Kompatibilität zu 640K-Laufwerken auch zu testen. Ich beziehe mich nun auf die Patchdiskette „PATCH 165" von Ehring elektronik. Unter dem Betriebssystem DOS können 640K-Laufwerke und Megaboard nicht sinnvoll gemeinsam verwendet werden, da auf der Harddisk nur 140K-Laufwerke simuliert werden können. Die mit dem Harddisk-Controller gelieferten Programme zum Kopieren von Files und ganzen Disketten arbeiten nicht mit 80 Tracks und den zusätzlichen Drives 3 und 4 für die Rückseiten der Disketten und die entsprechenden Programme auf der Patchdiskette „PATCH 164" können die verschiedenen Volumes auf der Harddisk nicht verwalten.
Unter CP/M und UCSD-Pascal können Megaboard und 640K-Laufwerke ohne Einschränkungen kombiniert werden, wenn man die Anpassung der Betriebssysteme auf 35-Track-Disketten vornimmt, bevor man die Systeme auf die Harddisk kopiert.
Ulrich Allgeier, Stuttgart

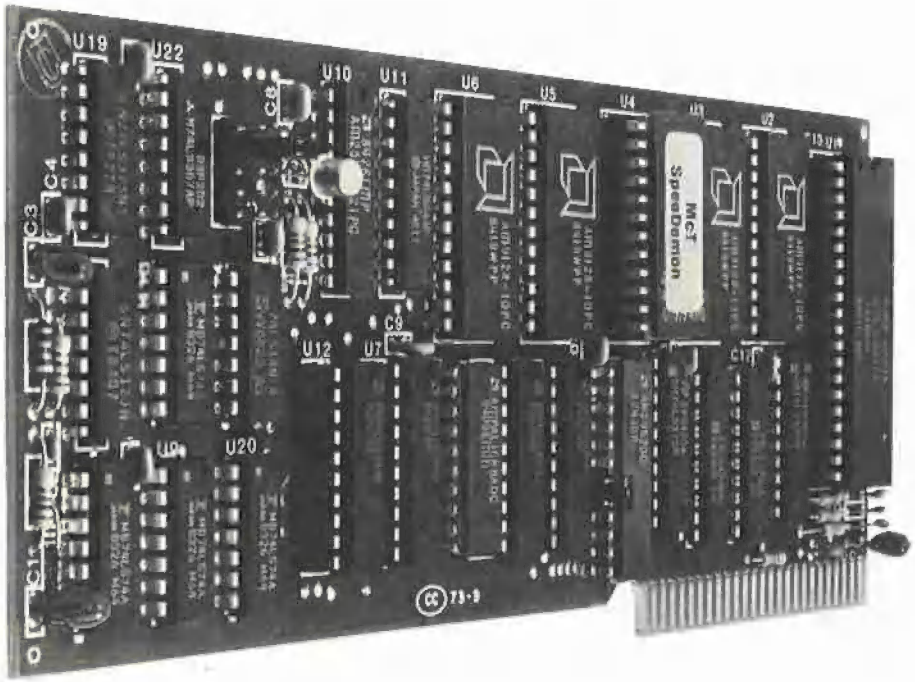


Foto Alvarez Softline

Speedemon-Karte

getestet von Ulrich Stiehl

Dies ist eine Karte, die mit einem 4-MHz-65C02C bestückt ist und Programme auf dem Apple IIe bzw. II Plus ähnlich wie die Accelerator-Karte beschleunigt, die bereits im Peeker, Heft 1/84, ausführlich besprochen wurde. Die Speedemon-Karte, die mir kurzfristig zur Verfügung gestellt wurde, läßt sich mit der Accelerator-IIe-Karte wie folgt vergleichen:

Preis: Beide Karten haben etwa den gleichen Preis, d.h. jeweils ca. DM 1100-, bis DM 1300,-, je nach Bezugsquelle.

Installation: Die Installation ist bei der Speedemon etwas problemloser als bei der Accelerator, weil bei letzterer auf der Karte kleine Schalter umgelegt und/oder Plastikstecker entfernt werden müssen, je nachdem, welche zeitkritischen Interface-Karten (Disk-Controller usw.) oder RAM-Karten sich im Apple befinden. Dafür läßt sich die Accelerator dann auch flexibler als die Speedemon anpassen, die von einer bestimmten Grundkonfiguration ausgeht (z.B. Disk-Controller in Slot 6 usw.) Im übrigen ist bei beiden Karten keine spezifische Software zur Aktivierung des 65C02C-Prozessors erforderlich. Einfach Karte in einen Slot stekken, Apple einschalten, fertig!

Reset: Die Accelerator IIe hat Reset-Probleme. Wenn sich z.B. ein RAM-Disk-Driver auf der 64K-Karte des IIe befindet und man just in dem Augenblick des RAM-Disk-Zugriffs Ctrl-Reset drückt, gelangt die Accelerator nicht mehr in den definierten Kaltstart-Softswitch-Zustand. So kann nach einem solchen Reset beispielsweise der Bereich $0200-$BFFF der unteren 64K lese/schreibfähig sein, während die Bereiche $0000-$01FF und $D000-$FFFF der oberen 64K weiterhin aktiv sind, womit der Apple „sich verabschiedet". Außerdem wird bei der Accelerator eine lese/schreibfähige Language-Card der unteren 64K nicht durch Ctrl-Reset abgeschaltet. All dies steht im Widerspruch zur normalen Ctrl-Reset-Funktion beim Apple IIe. Bei der Speedemon konnte durch entsprechende Tests ermittelt werden, daß man nach Ctrl-Reset stets wieder in den definierten Kaltstart-Softswitch-Zustand gelangt, bei dem die 64K-Karte und die Language-Card der unteren 64K abgeschaltet sind.

Geschwindigkeit: Bei der Speedemon fällt auf, daß sie offenbar bei RAM-Karten und beim Bildschirmspeicher der 80-Zeichenkarte (wegen des Bereichs $0400-$07FF der 64K-Karte) *sofort* auf 1 MHz heruntertaktet. So läßt sich z.B. zeigen, daß das AUXMOVER-Demo auf der MMU-Diskette (Hüthig-Software-Service), welches 8 HGR-Bilder von der 64K-Karte kaleidoskopartig in schneller Folge in den unteren HGR-Bereich überträgt, bei der Accelerator eine Datenübertragungsrate von ca. 78K/s aufweist, während es die Speedemon nur auf ca. 62K/s bringt. Ferner fällt beim Applewriter IIe auf, daß man mit der Speedemon erheblich langsamer als mit der Accelerator scrollen kann. Genaue Messungen wurden hier nicht vorgenommen, doch dürften sich ähnliche Werte wie bei dem AUXMOVER-Demo ergeben.

Zusätzliches RAM: Die Accelerator IIe (nicht so die alte Accelerator II) verfügt über 80K schnelles RAM. Zudem werden die zusätzlichen 16K RAM (80K – 64K = 16K) als Pseudo-ROM benutzt, in das eigene ROM-Inhalte, z.B. Forth usw., kopiert werden können (s. Peeker, 1/84). Diese Möglichkeiten entfallen bei der Speedemon, die praktisch fast gar kein eigenes RAM hat. (Sie soll laut einer amerikanischen Zeitschrift über einen kleinen CMOS-RAM-Baustein verfügen, in den die jeweils abzuarbeitenden Befehle kopiert werden. Das Anleitungsfaltblatt schweigt sich jedoch hierüber aus.)

Fazit: Angesichts gleicher Preise (früher war die Accelerator IIe noch erheblich teurer) fällt eine Empfehlung für die eine oder andere „Beschleunigungskarte" schwer. Ich persönlich würde mir – wenn ich heute erneut vor einer Kaufentscheidung stünde – wieder die Accelerator IIe kaufen, weil das Pseudo-ROM vielfältige zusätzliche Möglichkeiten für Assemblerprogrammierer eröffnet. Ist man jedoch an dem im Peeker, 1/84, beschriebenen Pseudo-ROM nicht interessiert, so sollte man die Speedemon erwerben, weil hier die geschilderten Reset-Probleme nicht auftreten.

Nachtrag

Nachdem der obige Testbericht bereits fertiggestellt worden war, erwarb ich für die Peeker-Redaktion die Speedemon-Karte von der Firma Softline in Oberkirch. Ich hatte inzwischen Gelegenheit, genauere Untersuchungen und detailliertere Zeittests anzustellen. Die Speedemon-Karte verfügt über einen 4K-Cache-Speicher, den man sich wie ein Fenster in dem 64K-Gesamtspeicher vorstellen kann. Wenn z.B. eine Programmschleife ab $1000 *zum ersten Mal* abgearbeitet wird, „versteckt" (cacher = franz. verstekken) die Speedemon den RAM-Inhalt ab $1000 in ihrem eigenen Cache-Speicher. Wenn die Programmschleife dann *zum zweiten Mal* durchlaufen wird, so wird direkt auf die Programmkopie im Speedemon-Cache-Speicher und nicht mehr auf den Apple-RAM-Speicher zugegriffen. Angeblich sollen in dem 4K-Cache-Speicher mehrere Programmteile gepackt werden können, die sich in verschiedenen, weit auseinanderliegenden Bereichen des Apple-RAM-Speichers befinden. Diese Theorie konnte jedoch trotz zahlreicher Tests nicht erhärtet werden, denn in jedem der diversen Testprogramme war die Ausfüh-

rungszeit die gleiche. Dann las ich jüngst im Juli-Heft von „Apple Assembly Lines", S. 19, daß der Lesebefehl (LDA usw.) schneller als der Schreibbefehl (STA usw.) ausgeführt würde: „Stores into memory always slow down to a 1 MHz rate, because the stores must be performed in real RAM, not just cache RAM." Hier irrte jedoch der Autor B. Sander-Cederlof. Für
(a) 6502,
(b) Accelerator IIe und
(c) Speedemon
ergaben sich folgende Werte (s = Sekunde) in einem Schreib- und Lesetest, der aus Platzgründen hier nicht abgedruckt werden kann:
1. Schreibtest
(a) 144s
(b) 49s (Faktor 2,94)
(c) 41s (Faktor 3,51)
2. Lesetest
(a) 135s
(b) 41s (Faktor 3,29)
(c) 53s (Faktor 2,55)
3. Schreib- und Lesetest
(a) 279s
(b) 90s (Faktor 3,1)
(c) 94s (Faktor 2,97)
Daraus folgt, daß die Speedemon im Gegensatz zur Accelerator für STA-Speicherzugriffe (Faktor 3,51) weniger Zeit benötigt als für LDA-Speicherzugriffe (Faktor 2,55). Während die effektive Taktfrequenz bei der Accelerator IIe nur geringfügig (+/- 0,4) pendelt, schwankt sie bei der Speedemon erheblich (+/- 1,0). Besonders eklatant ist der Unterschied zwischen Lesen und Schreiben, wenn beispielsweise aus einer RAM-Karte in den 64K-Apple-RAM-Speicher gelesen werden soll, wie bereits der obige RAM-Disk-Test zeigte.

Die 16K-Akku-Karte LC-85

getestet von Harald Grumser

Es dürfte mittlerweile nur noch wenige Apple geben, die nicht mit einer Speichererweiterung ausgerüstet sind, zumal der Apple IIe bereits in der Grundausstattung über 64K verfügt. Daher stößt die Besprechung einer 16K-Karte sicher nicht unbedingt auf Interesse – wenn es sich dabei um eine herkömmliche Speicherweiterung handelt.
Die von der Firma Ing.-Büro Fricke, Berlin entwickelte und vertriebene Karte LC-85 zeichnet sich durch ihre Batterie-Pufferung aus, die es ermöglicht, daß die Daten auch nach Ausschalten des Rechners nicht verlorengehen. Darüber hinaus bietet diese Speichererweiterung eine weitere Option: Durch Umstellen eines Schalters auf der Karte wird deren Inhalt zum Pseudo-ROM.

Die LC-85 als Language-Card

In einem II Plus kann die Karte an Stelle einer normalen LC eingesetzt werden, wobei sich der Einbau durch einfaches Einstecken von anderen Karten abhebt, die z.T. eine Verbindung zu einem der IC-Sockel auf der Mutterplatine des Rechners erfordern. Im IIe erhält man zur bereits eingebauten LC einen weiteren Speichergewinn um 16K (in jedem freien Slot), wodurch sich die Kapazität auf 80K erhöht (in Verbindung mit einer erweiterten 80-Zeichenkarte 144K). Die Ansteuerung geschieht in gewohnter Weise. Durch Ändern der Softswitch-Adressen in bestehenden Programmen (z.B. Bit $C0F0 zum Selektieren der Bank 2 in Slot 7) wird die Karte sofort angesprochen und verhält sich kompatibel zu andern LCs (mit Ausnahmen: s.u).
Die beim Kauf mitgelieferte Diskette enthält u.a. einen RAM-Disk-Driver für 60 Spuren, der die Karte in eine Diskette verwandelt, die über S1 (unabhängig vom tatsächlichen Slot) angesprochen wird. Diese RAM-Disk bleibt beim Ausschalten des Rechners erhalten, so daß mit einem ebenfalls mitgelieferten Programm beim erneuten Start nur das DOS gepatcht werden muß, um wieder Zugriff auf die Daten zu bekommen. Bei dieser Möglichkeit möchte man fast schon von einer kleinen physischen Diskette sprechen. Die Übertragungsgeschwindigkeit dieses RAM-Disk-Drivers wird durch die Tatsache geschmälert, daß bei jedem Sektorzugriff eine Driver-Routine in den Eingabepuffer gelegt wird, um von der Karte wieder zum DOS zu springen. Mit einer ebenfalls mitgelieferten Routine kann das DOS jederzeit wieder in den alten Zustand versetzt werden, um z.B. eine neue Diskette zu initialisieren.
Wer auf eine RAM-Disk zugunsten von Ineger-BASIC verzichten möchte, erhält als II-Plus-Besitzer die Gelegenheit, die zweite Sprache in der LC-85 abzulegen, ohne diese beim Einschalten des Rechners laden zu müssen. Über den hierzu erforderlichen Patch im DOS gibt die Bedienungsanleitung Auskunft.

Die LC-85 als Pseudo-ROM

Eine völlig neue Art der Anwendung bietet die Möglichkeit, die LC-85 als Pseudo-ROM einzusetzen. Dazu wird ein Schalter auf der Karte in On-Stellung gebracht und beim nächsten Einschalten des Rechners werden die Apple-ROMs desaktiviert und an deren Stelle tritt der Inhalt der Speicherkarte.
Dadurch wird es möglich, ohne EPROM-Brenner eigene Programme beim Start des Rechners ablaufen zu lassen, oder Modifikationen im Monitor oder Interpreter auszuführen, die dann für den Rechner wie im ROM erscheinen. Da die Karte bei einem Reset im Gegensatz zu anderen LCs nicht desaktiviert wird, kann z.B. auch der Apple II Plus auf einem IIe simuliert werden, um Programme auf beiden Rechnern zu testen. Dies erreicht man dadurch, daß der II-Plus-Monitor und -Interpreter (die Applesoft-Interpreter der beiden Geräte sind bisher identisch) in der LC-85 abgelegt wird.
In ähnlicher Weise kann verfahren werden, um die Vorzüge des neuen Interpreters auszunutzten (s. Aufsatz „Die neuen IIe-ROMs" in diesem Heft). Hierzu überträgt man den alten Monitor des IIe in die LC-RAMs (der neue Monitor benutzt den CX-Bereich, der unseres Wissens bisher mit keiner RAM-Karte simuliert werden kann) und fügt den neuen Interpreter hinzu. Dieser arbeitet dann mit Ausnahme des SHLOAD-, STORE- und RECALL-Befehls anstandslos.

Wer die Firmware des Apple nach eigenem Geschmack ändern will, hat nun die Möglichkeit, diese Änderungen fest einzubauen. So kann z.B. die doppelte Grafikauflösung als fester Bestandteil des Interpreters implementieret werden. (Eine der ersten Änderungen dürfte der Titel beim Kaltstart sein, der bei $FB09-$FB10 liegt.) Dem Interpreter-Kenner eröffnen sich hiermit ungeahnte Möglichkeiten, all die kleinen Unzulänglichkeiten zu bereinigen, die den täglichen Umgang mit Applesoft erschweren. Wird aus Kompatibilitätsgründen die Original-Firmware gebraucht, genügt ein Softswitch, um die Apple-ROMs wieder zu aktivieren.

Technische Anmerkungen

Die LC-85 behält bei einem Reset ihren aktuellen Softswitch-Zustand bei. Dadurch unterscheidet sie sich stark von den normalen LCs. Diese Eigenschaft bringt erhebliche Vorteile mit sich, kann aber auch zu Schwierigkeiten führen. So wird damit in jedem Fall erreicht, daß das Pseudo-ROM nach Reset tatsächlich als „ROM" erhalten bleibt. Auf der anderen Seite hat der gewohnte Griff zur Reset-Taste, um den Rechnerzustand zu normalisieren, nur noch beschränkte Wirkung. Macht man sich diese Tasache bewußt, so überwiegen jedoch die Vorteile.
Beim Einschalten des Rechners versetzt sich die Karte im Pseudo-ROM-Modus stets in einen definierten Zustand: Bank 2 wird selektiert und die Karte im schreibgeschützten Zustand gelesen. Wäre dies nicht der Fall, könnten seltsame Dinge geschehen.
Was das An- und Abschalten des RAMs anbelangt, reagiert die LC-85 wie jede andere Language-Card. Dies bringt eine eingeschränkte Anwendung als ROM mit sich. Ist die Karte aktiv, kann keine andere LC angesprochen werden. Dies bedeutet für den Pseudo-ROM-Modus, daß keine Programme gestartet werden können, die ihrerseits auf eine 16K-Speichererweiterung zugreifen. Die System-Master-Diskette des DOS 3.3 lädt somit kein Integer-BASIC (vorausgesetzt, die LC-85 steckt nicht in Slot 0, was in diesem Zusammenhang zum Überschreiben des Inhalts führen würde). Somit muß die Karte desaktiviert werden, bevor man z.B. den Applewriter bootet; d.h., daß manche Programme bei aktivem Pseudo-ROM nicht mehr kalt gestartet werden können.
Leider sind diese Hinweise nicht in der Bedienungsanleitung enthalten, so daß hier oft nur das altbewährte Trial-and-error-Verfahren weiterhalf.

Fazit

Wenn man von einigen technischen Eigenarten absieht, die nicht immer wünschenswert sind, stellt diese Karte eine lohnende Bereicherung der Apple-Interna dar. Mit etwas Phantasie lassen sich völlig neue Lösungswege für Probleme beschreiten, die bisher nur recht umständlich zu bewältigen waren.
Das verblüffenste Argument für diese Karte dürfte ihr Preis sein: Für DM 175,- lohnt sich selbst die Anschaffung als reine Language-Card, die darüber hinaus auch noch mit einer Diskette zum sofortigen Einsatz geliefert wird.

Zwei neue Apple-IIe-Kompatible

SCSes

Die Firma SCS Sander Computer Systeme in Remscheid importiert zwei neue Apple-IIe-Kompatible, die in den nachfolgenden Berichten vorgestellt werden sollen. Das SCSe-Modell wird im Apple-II-Gehäuse und das SCSes-Modell im IBM-Gehäuse geliefert.

Apple-IIe-Kompatibler SCSe

getestet von Jürgen Geiß

Die Hardware

Man sieht gleich auf den ersten Blick, daß es sich hier um einen Nachbau handeln muß. Zwar hat das Gehäuse die typische beige Farbe, sieht aber sonst nach billigem Plastik aus. Außerdem ist das Gehäuse eine Kopie des schon legendären Apple II, der auf der Rückseite 3 breite Schlitze für das Herausführen der Kabel einiger Interfacekarten aufweist. Ebenfalls nur apple-II-kompatibel ist der Joystick-Anschluß. Hier ist nur der leere Sockel auf der Platine vorgesehen, nicht aber die Steckbuchse, die beim IIe neben dem Kassettenausgang liegt.
Zur Tastatur ist leider auch einiges zu sagen: Wie bei vielen Kompatibilen macht die Tastatur nicht gerade einen sehr stabilen Eindruck. An der Leertaste kann gewackelt werden; sie sitzt nicht so fest wie bei seinem Vorbild. Außerdem werden deutsche Benutzer enttäuscht sein: Es fehlt der Umschalter zwischen amerikanischer und deutscher Tastatur. Das liegt daran, daß es sich um den Nachbau des amerikanischen Apple IIe handelt.

Klappt man den Gehäusedeckel auf, der ebenfalls „nur" apple-II-kompatibel ist, so entdeckt man einige Unterschiede. Erfreulicherweise wurde beim Netzteil nicht gespart. Während das Apple-IIe-Netzteil bei 5 Volt nur 2,5 Ampere liefert, bringt es der SCSe auf ganze 5 Ampere. Damit dürfte ein problemloses Arbeiten möglich sein, auch wenn alle 7 Slots belegt sind. Überhaupt macht die Platine einen recht sauber verarbeiteten Eindruck, so daß sich kaum Funktionsstörungen aufgrund einer defekten Platine ergeben dürften. Die Platine besitzt ebenfalls sieben Slots und einen Auxiliary Slot für die 64K-Erweiterung. Nur sitzt dieser nicht unterhalb von Slot 3 wie beim europäischen Apple IIe, sondern ganz links neben Slot 1 wie

beim amerikanischen Apple IIe. Benutzt man die 64K-Erweiterung nicht, so sollten alle 7 Slots gleichwertig sein. Leider konnte keine einzige Interfacekarte wie Drucker, Diskette oder 80-Zeichen-Ultraterm in Slot 3 zum Laufen gebracht werden. Sobald dieser Slot belegt ist, gibt der Rechner manchmal den Geist auf. (Anm.d.Red.: Dieses Problem hat auch der Original-Apple-IIe. In Verbindung mit den neuen IIe-ROMs wurden hier Verbesserungen vorgenommen. us) Dies ist aber kein großer Nachteil, da ja meistens mit der Apple-IIe-80-Zeichenkarte im Auxiliary Slot gearbeitet wird und dann Slot 3 ohnehin nicht mehr benutzt werden soll.

Zum Testen lag noch die firmeneigene 80-Zeichenkarte mit den zusätzlichen 64K RAM vor. War diese eingesteckt, so verhielt sich der Rechner genau wie ein IIe mit 128K. Die Double-Hires-Grafik funktionierte nicht auf Anhieb, da auf der mitgelieferten 80-Zeichenkarte die Brücke fehlte, die Pin 55 mit Pin 50 verbindet. 2 Lötstellen sind aber vorgesehen, und wer diese mit etwas Geschick und einem Lötkolben verbindet, kann sich der Double-Hires erfreuen. Wer jetzt aber glaubt, er könne 16 Farben gleichzeitig darstellen, der muß enttäuscht werden. Das Video-Signal des SCSe liefert nur die amerikanische NTSC-Norm, und mit einem deutschen Farbfernsehgerät, das ja bekanntlich mit der PAL-Norm arbeitet, sieht man nur rosa-graue Schlieren. Soweit zur Hardware.

Die Software

Zur ihr kann nur Erfreuliches gesagt werden. DOS 3.3 läuft ohne Probleme, ebenso ProDOS. Das will schon etwas heißen, denn ProDOS reagiert auf Nachbauten nicht gerade freundlich. Sowohl Pascal 1.1 als auch 1.2 (64K und 128K Version) liefen anstandslos, was die Funktionsfähigkeit der erweiterten 80-Zeichenkarte bewies. Die Software-Kompatibilität scheint also 100%ig zu sein.

Abschließend kann man sagen: Wer weder auf Farbe noch auf die umschaltbare deutsche Tastatur Wert legt, ist mit dem SCSe gut bedient, zumal alle Softwarepakete einwandfrei laufen und das Gerät voll ausgebaut werden kann, ohne daß das Netzteil in die Knie geht. Den Preis von ca. 1000,- DM ist der SCSe sicher Wert.

Apple-IIe-Kompatibler SCSes

getestet von Harald Grumser

Nachdem Apple-II-Plus-Kompatible in ähnlichen Stückzahlen wie der Original-Apple-II-Plus verkauft worden sind, kommt nun auch der Apple IIe in den Ruhm, einigen Computerherstellern als Vorbild zu dienen.

Einer dieser IIe-Kompatiblen wird von der Firma Sander Computer-Systeme importiert und über den Fachhandel vertrieben. Dieses Gerät, das weniger als die Hälfte des Originals kostet, kann gegen Aufpreis in einem IBM-ähnlichen Gehäuse unter der Bezeichnung SCSes bezogen werden. Zum Lieferumfang gehört außer dem Computer, der in der gleichen Grundaustattung wie sein Vorbild geliefert wird (also keine eingebaute 64K-Karte oder Z80-Prozessor), das Buch „Apple II leicht gemacht" von Joseph Kascmer und außerdem bei der IBM-Gehäuse-Version ein kleines Handbuch zur Benutzung der externen Tastatur.
Zum Test standen darüber hinaus zwei Teac-kompatible Laufwerke zur Verfügung, die als weitere „Preisbrecher" bezogen werden können.

Die Tastatur

Die externe Tastatur kann wahlweise in deutscher oder amerikanischer Belegung bezogen werden. Der zusätzliche 10er-Block (mit Delete-Taste) kann programmiert werden, wobei die Speicherung durch den Batteriepuffer nach Herstellerangaben 5 Jahre lang erhalten bleiben soll. Zu den normalen Tasten stehen weitere 10 Funktionstasten zur Verfügung, die zum einen als Reset-Taste dienen (F1 und F2 zusammen) und auch die apple-spezifischen Apfeltasten ersetzen. Die restlichen Funktionen sind vordefiniert (z.B. CATALOG, CALL -151 usw.). Eine zusätzliche Alternate-Taste ermöglicht die Eingabe von 21 Applesoft-Kommandos über einen Tastendruck (z.B. Alt-G = GOTO). Außerdem kann ein akustisches Signal wahlweise eingestellt werden. Als nachteilig erweist sich die fehlende Key-Roll-over-Einrichtung. Beim Betätigen mehrerer Tasten wird die letzte nicht mehr erkannt, was sich bei sehr schneller Schreibweise negativ auswirken kann.

SCSe

Software

Der Computer selbst stand in erster Linie bezüglich seiner Kompatibilität auf dem Prüfstand. Die 80-Zeichenausgabe (eine 64K-Karte wurde mitgeliefert) funktionierte auf Anhieb, und auch die Ausgabequalität konnte sich sehen lassen. Die nachempfundenen Apfeltasten wurden vom Applewriter erkannt, und auch andere Standard-Programme und Betriebssysteme (Pascal, ProDOS und CP/M 2.2 in Verbindung mit der Z80-Karte) liefen einwandfrei. Die Installierung verschiedener RAM-Disks brachte keine Überraschungen mit sich und arbeitete erwartungsgemäß.
Da bei Importgeräten dieser Preisklasse nicht auf die nationalen Märkte eingegangen werden kann, mußte auf eine umschaltbare deutsch-amerikanische Bildschirmausgabe verzichtet werden.

Der hier vorgestellte Kompatible SCSes vermag in der Version mit externer Tastatur nicht jene Zuverlässigkeit auszustrahlen, die der Original-IIe vermittelt, was zeigt, welche Rolle die Tastatur bei der subjektiven Bewertung eines Rechners spielt. Stellt man seine Leistungen jedoch in Relation zu seinem Anschaffungspreis, so ist er durchaus erwägenswert. Für den, der nicht täglich mehrere Stunden am Rechner verbringt oder gesteigerten Wert auf die volle Ausnutzung der speziellen Fähigkeiten eines Markencomputers legt, empfiehlt sich die Anschaffung eines (fast) Kompatiblen; das sporadische Tennisspiel erfordert auch keinen teuren Puma-Schläger.

Es bleibt zu hoffen, daß die Firma Apple diese Herausforderung erkennt und sich aus ihrer Gefälligkeit erhebt, um vielleicht doch noch mit einem neuen IIx ihren gewelkten Lorbeerkranz zu erfrischen.

Firmenmitteilungen

4.000 Frei-Programme
INTUS Lern-Systeme AG, spezialisiert auf computer-unterstützte Lernprogramme, bietet über 4.000 Frei-Programme (Public Domain) für Apple II Computer an. Das sind Programme, die von ihren Autoren freigegeben worden sind (kein Copyright mehr). Sie stammen hauptsächlich aus den USA. Es handelt sich um Lern- und andere Schulprogramme, Geschäftsprogramme, Hilfsprogramme, Spiele, Grafik/Kunst, Luftfahrt, Musik, Gesundheit/Essen, Astronomie, Psychologie usw. Auf jeder Diskette sind 10 bis 30 Programme enthalten.

Die Vermittlungsgebühr beträgt DM 14,- je Diskette. Die Programmliste kann für DM 10,- (im voraus in bar oder Briefmarken) bezogen werden.

Ein gleicher Service ist für Programme vorgesehen, die auf Macintosh lauffähig sind.

Quelle: Intus Lern-Systeme, Waldshut-Tiengen

Buchhaltungsprogramm BUCH
Ganz neue Vertriebswege beschreitet die Firma Röntgen Software mit Ihrem Buchhaltungsprogramm BUCH. Jeder Interessent kann sich das Programm für 8 Tage zur Ansicht bestellen und es in aller Ruhe austesten. Bei Nichtgefallen wird das Programm einfach zurückgeschickt. Auch sonst ist die neueste Version V2.4 des erprobten Buchhaltungsprogramms interessant: Gewinn-und-Verlust-Rechnung, automatische Umsatzsteuerberechnung, Ausgabe der Kontenblätter, Betriebsübersicht, Journal und Kassenführung wird ebenso einfach und sicher erledigt wie Kreditoren- und Debitorenkonten. Eine ausführliche Einführung macht es selbst EDV-Laien möglich, sich schnell und zuverlässig in das Programm einzuarbeiten. Der Preis liegt mit DM 595,- weit unter dem vergleichbarer Produkte. BUCH läuft auf Apple II+, IIe und IIc.

Quelle: Röntgen Software, Edelstetten

Präsentationsgrafiken mit (G)GROGRAF
Seit Ende 1984 vertreibt die Firma SanData, Apple-Händler in Nürnberg, ein Grafikprogramm namens ROGRAF; seit Juni 85 ist eine erweiterte Version GROGRAF hinzugekommen. „(G)ROGRAF ist das elektronische Zeichenbrett für jeden Apple", verspricht der Prospekt – Grund genug, sich das Angebot einmal näher anzuschauen.

Zuerst die Gemeinsamkeiten und Unterschiede: Beide Programme kommen ohne Maus, Tablett oder Pad aus. Durch Tastensteuerung kann ein Fadenkreuz in präzisen Schritten über den Bildschirm bewegt werden, und mit einfachem Tastendruck können grafische Grundelemente wie Punke, Geraden, Rechtecke, Dreiecke, Kreise und Kreissegmente abgerufen werden. Außerdem können waagrechte und senkrechte Beschriftungen angebracht, Flächen schraffiert, Rechteckflächen ausgefüllt, invertiert und gelöscht werden. Freihandzeichnungen sind nicht möglich. Selbstverständlich können die erstellten Zeichnungen auf Diskette gepeichert und auf Matrixdrucker ausgedruckt werden. Als besondere Stärke der Programme wird die Möglichkeit bezeichnet, eine Bibliothek von grafischen Symbolen ähnlich einer Zeichenschablone anzulegen. Diese „Makrofähigkeit" ist, wie die Erfahrung zeigt, auch als Zeichenhilfe für manche Anwendungsfälle geeignet. Der einzige, aber wichtige Unterschied zwischen den beiden Versionen besteht in der Größe der erstellbaren Grafiken: ROGRAF arbeitet nur auf Bildschirmformat, wobei noch eine Zeile für Bedienerführung genutzt wird; die nutzbare Zeichenfläche beträgt 280 * 184 Punkte. GROGRAF hingegen nutzt eine mehr als dreimal so große Fläche von 448 * 360 Punkten, von denen immer nur ein Ausschnitt von ROGRAF-Größe sichtbar ist und bearbeitet werden kann; dieser Ausschnitt kann beliebig verschoben werden. Druckt man das Bild mit Matrixdrucker, so füllt das eine etwa DIN-A5-Format, das andere eine halbe Postkarte. Trotzdem werden weiterhin beide Versionen angeboten, denn auch das kleine Format ist für viele Anwendungen ausreichend, benötigt für Drucken und Speichern nur ein Viertel der Zeit, auf der Diskette nur ein Drittel des Speicherplatzes je Grafik und kommt zur Not auch mit einem Diskettenlaufwerk aus. Da die Handhabung der Programme ansonsten identisch ist, ist die Anschaffung des Pakets aus beiden Programmen empfehlenswert, insbesondere da das Paket auch die Möglichkeit bietet, Teilbilder zwischen beiden Systemen zu übertragen. ROGRAF kostet ca. DM 300,-.

Quelle: SanData, Nürnberg

Melco-Duckpuffer von Watanabe
Immer dann, wenn an einem Computer – gleich welcher Größenordnung – ein Printer oder Plotter läuft, steht der Rechner still und kostet Geld. Mit dem Melco-Printer/Plotter-Buffer läßt sich hier leicht und preiswert Abhilfe schaffen, läßt sich Zeit und Geld sparen. Nehmen wir an, ein Computer ist täglich 8 Stunden in Betrieb und der dazugehörige Printer läuft 3 Stunden. Dies sind 3 Stunden, in denen der Rechner zwar läuft, aber nicht benutzt werden kann. Nehmen wir weiter an – machen wir es billig – die Rechnerstunde kostet mit allem Drumherum DM 100,-, so sind es immerhin DM 300,-, die der Printer verbraucht, während der Rechner steht. Wenn man die Übertragungszeit zwischen Rechner und Printer auf 20% der bisherigen Zeit verkürzt, spart man DM 240,- an Rechnerzeit. Die Melco-Printer/Plotter-Buffer können die Übertragungszeiten tatsächlich auf unter 20% reduzieren. Sie übernehmen die Daten vom Rechner je nach Buffertyp mit 500 oder 10.000 cps und geben sie in einer dem Printer angepaßten Geschwindigkeit weiter. Je nach Bedarf stehen Buffer mit verschiedenen Kapazitäten bis hinauf zu 2 MByte und mit den Standard-Schnittstellen zur Verfügung.

Quelle: Watanabe Gmbh, Herrsching

Digitalisiertabletts höchster Präzision
Die MICROGRID-Serie von Summagraphics zeichnet sich durch die hohe Auflösung von 40 Linien/mm und die ausgezeichnete Genauigkeit von +/-0,12 mm aus. Diese hohe Genauigkeit kann dank eines automatischen Laserinterferometer-Tests garantiert und belegt werden. Die Digitizer sind dünn, leicht, an den Ecken abgerundet und haben eine blendfreie Oberfläche. Sie sind in verschiedenen Größen von 12" x 12" (305 x 305 mm) bis 42" x 60" (1067 x 1523 mm) lieferbar, wahlweise mit Stylus oder mit 3, 4 oder 16 Tasten Cursor. Für die größeren Ausführungen gibt es Ständer, so daß man die Tabletts in verschiedenen Winkeln aufstellen kann. Alle Systeme haben eine duale RS-232C-Schnittstelle, eine 8-Bit-parallel-Schnittstelle, zwei Transducerports und eine Selbstdiagnose. Busoptionen wie IEEE, RS 449, 16 Bit-parallel oder spezielle OEM-Interfaces und Speichererweiterung für Applikationssoftware sind lieferbar.

Die hohe Zuverlässigkeit erreicht Summagraphics durch Kontrolle und Testlauf auf der Komponentenebene, auf der Subsystemebene und durch einen computergesteuerten Endtest.

Quelle: nbn Elektronik GmbH, Herrsching

Neuer Apple-Kompatibler
Täglich wird das Angebot guter und preiswerter Computer größer. Trotz der vielen neuen Rechner ist der Apple IIe der Dauerbrenner des Computermarktes. Die zur Verfügung stehende Software ist in ihrer Vielfalt einmalig. Noch immer werden neue Erweiterungskarten angeboten.

Zwischenzeitlich ist dieser Rechner nur noch als „Nachempfindung" erhältlich. Die Firma CCTR GmbH bietet einen C/PM-fähigen Kompatiblen an, von dem sie behauptet, daß dies der Apple IIe ist, den die Firma Apple versäumte zu bauen. Der Rechner bietet eine deutsche, IBM-PC-ähnliche, freistehende Tastatur mit 12 Funktionstasten, 2 Laufwerke mit jeweils 40 Spuren, 7-Ampere-Schaltnetzteil, 64K-RAM, Z80- und 6502-CPU. Preis: DM 1.699,-.

Quelle: CCTR GmbH, Maintal

Flugsimulator mit deutscher Anleitung
Mit dem Flugsimulator kann man realitätsnah auf 80 Flugplätzen starten und landen. Dieses Programm hatte bisher einen einzigen Nachteil: Um die vielen Möglichkeiten – z.B. Funknavigation oder Verändern der Wettervariablen oder Kunstflug – wirklich nutzen zu können, mußte man schon sehr gut in Englisch sein. Für alle die vielen, die damit nicht hundertprozentig zurechtkamen, hat „Softline" ein deutsches Handbuch herausgebracht. Titel: „Fliegen mit dem Heimcomputer – Tips und Hilfen für das Training mit dem Flight Simulator II". Autor ist der bekannte Luftfahrt-Journalist E.U. Adler, der für große Zeitschriften und Zei-

tungen schreibt und begeisterter Bildschirmflieger ist. Ihm ist es gelungen, alle fliegertechnichen Fragen wie auch alle Sprachprobleme für neue wie altgediente Computer-Piloten verständlich zu machen.
Quelle: Softline, Oberkirch

OKI-Drucker für Apple II
Eine gute Nachricht für alle Besitzer von Apple-Computern und solche, die es werden wollen: OKIDATA liefert jetzt voll apple-kompatible Versionen der Drucker OKIMATE 20 sowie MICROLINE 192/193. Es bietet sich so z.B. die Möglichkeit, Schönschriftqualität bei hoher Druckgeschwindigkeit mit den Matrix-Druckern der neuen MICROLINE-Serie für den Apple II+, IIe oder IIc zu realisieren.
Quelle: OKIDATA GmbH, Düsseldorf

Macprint-Satzsystem
Vom Manuskript bis zur reprofähigen Druckvorlage in 30 Minuten. Dies wird laut Angabe des Vertreibers Bense KG durch eine integrierte Hard- und Softwarelösung ermöglicht, die in der Grundkonfiguration aus einem Apple-Macintosh-Computer, einem Klarschriftleser, einem Laserdrucker und spezieller Software für den mehrspaltigen Seitenumbruch unter Berücksichtigung von Grafiken besteht. Die einzelnen Komponenten des Systems werden über das lokale Netzwerk Appletalk verbunden. Damit ist eine Erweiterung auf bis zu 32 Arbeitsplätze möglich. Eine arbeitsfähige Grundkonfiguration kostet ca. DM 100.000,-.
Quelle: Bense KG, Coesfeld

Macprolog2-Interpreter
Im Rahmen ihrer Forschungstätigkeit im Technologiezentrum Dortmund hat die Bense KG unter dem Namen Macprolog2 einen PROLOG-Interpreter für den Apple-Macintosh fertiggestellt. PROLOG wird in Expertensystemanwendungen wie Krankheitsdiagnosen, Rechnerkonfiguration, Fehlererkennung in Schaltkreisen usw. eingesetzt. Voraussetzung für den Einsatz ist ein Macintosh-Computer mit 512K RAM und 2 Diskettenlaufwerken.
Quelle: Bense KG, Coesfeld

Turbo-Lader Graph
Turbo-Lader Graph ist ein windowfähiges Grafik-System, das speziell für Anwendungen mit Turbo-Pascal im Rahmen der Turbo-Lader-Serie von Lauer & Wallwitz entwickelt wurde und für alle Apple-II-Rechner und Kompatible verfügbar ist, auf denen Turbo-Pascal implementiert ist. Der Turbo-Lader Graph ist mit allen Turbo-Pascal-Versionen direkt lauffähig. Mit Ausnahme der zeitkritischen Operationen ist das System in Turbo-Pascal geschrieben. Sowohl die Pascal- als auch die kommentierten Assembler-Quelltexte werden dem Nutzer mitgeliefert, um ihm so die Anpassung an die eigenen Bedürfnisse zu ermöglichen. Das umfassende Programmpaket ist inklusive der umfassenden Dokumentation in deutscher Sprache direkt beim Hersteller Lauer & Wallwitz für DM 198,- erhältlich.
Quelle: Lauer & Wallwitz, Wiesbaden

Sound Sampling System
Die Firma ACT hat das Sound Sampling System DX-1 der Firma Decillionix, USA, in ihren Vertrieb aufgenommen. Das DX-1 ist ein Sound Sampler für den Apple-II-Computer, mit dem sich Naturklänge mit einer Abtast-Rate von 0,78 kHz bis 36 kHz aufnehmen, im Arbeitsspeicher des Rechners editieren und über 5 Oktaven abspielen lassen. Die Klänge können verändert, auf Disketten abgespeichert und wieder geladen werden. Die maximale Länge eines Klanges beträgt 10 Sekunden. Es wird auch eine große Palette fertiger Klänge auf Disk angeboten. Das Zubehör-Programm „Echo" macht aus dem Apple II ein digitales Hall- und Echogerät. Das Programm „Splash" erlaubt die Darstellung von Sound- und Hüllkurve auf der hochauflösenden Grafikseite des Apple II in sechs verschiedenen Darstellungsformen.
Quelle: ACT Computersysteme, Düsseldorf

Apple dementiert „Heute Journal"
„Apple steht nicht an der Spitze der 400 Heimcomputer-Hersteller, die von der Pleite bedroht sind", erklärt Ralph M. Deja von der Apple Computer GmbH, München, zu dem TV-Fim „Pleitegeier über dem Silicon Valley" im „Heute Journal" des Zweiten Deutschen Fernsehens (ZWF) vom 24. Juli. Wie der Geschäftsführer der deutschen Vertriebstocher des amerikanischen Mikrocomputerpioniers zum ZDF-Film erläutert, gibt es weder 400 Heimcomputerhersteller, noch zählt Apple überhaupt zu den Heimcomputer-Herstellern. Richtig ist dagegen, daß es etwa 400 Mikrocomputer-Hersteller gibt und daß in der Branche der Auslese-prozeß begonnen hat.
Quelle: Apple Computer GmbH, München

Apple GmbH erhöht Stammkapital
Die Apple Computer GmbH, München, hat das Stammkapital von 50.000,- DM auf 7,5 Mill. DM erhöht. Wie der Geschäftsführer der deutschen Vertriebsgesellschaft des amerikanichen Mikrocomputerpioniers, Ralph M. Deja, erklärt, unterstreicht die kräftige Kapitalerhöhung die Schlüsselstellung des deutschen Marktes. „Zusammen mit Frankreich nimmt die Bundesrepublik die Führungsrolle innerhalb der europa-weiten Apple-Aktivitäten ein". Die Führungsposition wird ausgebaut werden. Vor allem in die Bereiche Vertrieb, Service und Schulung wird weiter investiert.
Quelle: Apple Computer GmbH, München

Apple-Händler-Netz umstrukturiert
Im Rahmen der weltweiten Reorganisation macht Apple das Macintosh-Computersystem auch für externe Entwickler und Zulieferer zugänglicher und strukturiert das Fachhändler-Netz um. Wie Ralph M. Deja, Geschäftsführer der deutschen Vertriebstochter des amerikanischen Mikrocomputerpioniers, jetzt auf einem Fachhändler-Seminar ausführte, wird Apple in Zukunft stärker vom Markt als vom Produkt her gesteuert. Das bedeutet nicht, daß Apple die Rolle als Technologieführer in der PC-Branche aufgegeben und nicht mehr die fortschrittlichsten Produkte bauen wird. Das bedeutet aber, daß Apple als marktorientiertes Unternehmen die Produkte künftig noch gezielter dem tatsächlichen Nutzen der Kunden anpassen wird. „Wir werden genauer hinhören, um die Bedürfnisse unserer Kunden zu erfahren, welche Produkte und Lösungen sie von Apple erwarten. Wir werden die Brücken zu externen Entwicklern verstärken, um dem Markt mehr Lösungen zu geben. Gleiches gilt für die Anbieter von ergänzender Hardware, die den Gebrauchsnutzen unserer Geräte für den Kunden wesentlich steigern. Wir kehren damit zu einem Verhalten zurück, das schon dem ersten Apple II zu seinem phänomenalen Aufstieg und Durchbruch verhalf. Wir beseitigen die rechtlichen und andere Hindernisse, damit andere ihre Produkte und Lösungen an Apple anbinden können. Wir werden auch stärker als bisher internationale Kommunikationsprotokolle und Standards berücksichtigen und die Koexistenz von Apple II und Macintosh mit anderen System-Umgebungen weiter ausbauen".
Apple-System-Händler verkaufen keine Heim- und Hobby-Computer, sondern zielgruppenorientiert professionelle Systeme wie den Apple II oder den Macintosh. Mit der Umstrukturierung zu einem marktgerechten Fachhändler-Netz wird die Zahl der Händler bis Ende 1986 von derzeit über 300 auf rund 180 reduziert.
Quelle: Apple Computer GmbH, München

MAUS: Mac an Hochschulen
Das Apple-Projekt MAUS (Macintosh-Apple-Universitäts-System) breitet sich auch in Europa aus. Um auch deutschen Studenten die Informationswelt offen zu halten, hat Apple das Projekt MAUS entwickelt. Damit können die Hochschulen ihren vorhandenen Lehrmitteletat effektiver ausschöpfen. Denn mit MAUS wird den Hochschulen ein händlerähnlicher Status eingeräumt, womit sie Apple-Produkte für den Eigenbedarf zu wesentlich ermäßigten Preisen beschaffen können. Das gilt nicht nur für Geräte in Lehre und Ausbildung, sondern auch für Forschung und Verwaltung.
Quelle: Apple Computer GmbH, München

Mac-Leasing für Studenten
Jeder Student ab dem Vordiplom oder ab der Zwischenprüfung an deutschen Hochschulen und Fachhochschulen soll sich seinen eigenen Macintosh leisten können. Aus diesem Grund startet die Apple Computer GmbH, München, zum 1. Juli 1985 das Sonderprogramm „Studmac". Wie die deutsche Tochtergesellschaft des kalifornischen Mikrocomputerpioniers anläßlich der Hannover-Messe 1985 bekanntgab, können Studierende über autorisierte Apple-Händler und Applerent ein Macintosh-System für rund 275,- DM im Monat leasen.
Quelle: Apple Computer GmbH, München

Leasing von Apple-Computern
Zu günstigen Konditionen und mit einem umfangreichen Leistungspaket sind Apple Computer-Systeme, Peripheri und Software jetzt auch über Leasing zu haben. Nach Abschluß eines Leasingvertrages bei Apple-Fachhändlern kann jeder Kunde seinen Apple sofort mitnehmen. Außer den Steuervorteilen bietet der Leasingvertrag eine Reihe von zusätzlichen Leistungen. In der Leasingrate eingeschlossen sind 12 Monate Garantie, Schwachstrom- und Diebstahlversicherung und zusätzlich eine Risiko-Lebensversicherung, die den

Privatkunden im Schadensfall abgesichert.
Quelle: Apple Computer GmbH, München

BTX für Apple IIc
Rechtzeitig zur diesjährigen Hannover-Messe 1985 ist das komplette BTX-Softwarepaket „Telesoft" für den Apple IIc auf den Markt gekommen. „Telesoft", das den Apple IIc zu einem professionellen BTX-Arbeitsplatz macht, wird von der Apple Computer GmbH in Verbindung mit der Baud BTX-Agentur, München, angeboten. Das BTX-Softwarepaket setzt sich aus einem Grund- und fünf Anwendungsmodulen zusammen. Telesoft ist FTZ-zugelassen; damit darf das Gesamtsystem ohne zusätzliche BTX-Tastatur betrieben werden.
Quelle: Apple Computer GmbH, München

Teletex für den Apple II
Der Apple II ist jetzt auch teletexfähig. Wie die Apple Computer GmbH ausführt, wird der Apple II zusammen mit der UTC-Box zu einem Teletexgerät. Die Kombination aus Apple II, dem Apple Matrix-Drucker, der UTC-Box und dem Programmpaket TTX Pack ist jetzt von der Deutschen Bundespost zugelassen worden. Preis ca. DM 7.000,-.

Mac-Pressekonferenz bei Apple in München

von Harald Grumser

Am 13. August lud die Firma Apple zu ihrer ersten Fachpressekonferenz in die Zentrale nach München ein, die zukünftig in regelmäßigen Abständen stattfinden soll, um über neueste Produkte und applespezifische Belange zu berichten.
Die Themen dieser ersten Fachpressekonferenz beschränkten sich – von der Diskussion grauer Märkte abgesehen – ausschließlich auf Macintosh-Produkte von Nicht-Apple-Firmen.

Graue Märkte
Ein für die Firma Apple derzeit vordergründiges Problem scheinen die grauen Märkte darzustellen, die jährlich erhebliche Verluste in finanzieller Hinsicht als auch im Ansehen der Firma einbringen. So wurde dieser Markt in zwei Kategorien eingeteilt:
Zum einen tauchen immer wieder Originalgeräte auf dem freien Markt auf, die z.T. um 10% unter den Händlerrabattpreisen liegen und somit eine ernste Konkurrenz zu den von autorisierten Apple-Händlern verkauften Geräten darstellen. Diese Billiggeräte werden oft „auf die Schnelle verkauft" und enthalten keine Garantieansprüche. Auch entfällt die nach Angaben der Firma Apple fachkundige Kundenbetreuung bei den autorisierten Händlern. Recherchen haben ergeben, daß diese Apples z.T. unter Rückerstattung der Mehrwertsteuer ausgeführt und dann wieder auf grauen Wegen in die Bundesrepublik eingeführt werden.
Die zweite Gruppe der Geräte liegt in der Vielzahl von Nachbauten, „die vom Original kaum zu unterscheiden sind." So wurde von einigen Fällen berichtet, bei denen enttäuschte Kunden in München angerufen hatten, weil Programme oder Zusatzkarten auf diesen Geräten nicht liefen. Oftmals wäre diesen Kunden nicht bewußt, daß sie einen billigen Nachbau erworben hatten. Daher „kann vor diesen – den Ruf der Firma Apple schädigenden – Geräten nur nachdrücklich gewarnt werden".

Omni-Drive von Corvus
Zum zweiten wurde das Omni-Drive-System der Firma Corvus vorgestellt, das in der Niederlassung in München bereits im Appletalk-Netz als zentraler Daten-Pool installiert wurde.
Diese Festplatte (lieferbar als 45,1-, 16,6-, 11,1- oder 5,5-MByte-Platte) kann im Netz die Kosten für externe Massenspeicher enorm senken, ist aber auch einzeln einsetzbar. So kostet 1 MByte Speicher bei der 45M-Corvus nur noch ca. DM 615,- inkl. MwSt. (= DM 27700,- : 45,1). Der Anschluß der Platte erfolgte über das lokale Netz Appletalk, das bis zu 32 Einheiten miteinander verbinden kann, die dann auch auf einen Drucker (z.B. Laserwriter) zugreifen können. Die Software für alle Geräte wird auf der Platte abgelegt, so daß kein Diskettenwechsel mehr nötig wird. Die Installierung dieses Netzes ist durch bloßes Zusammenstecken der einzelnen Komponenten möglich, wobei für jeden Mac nur das Verbindungskabel für ca. DM 200,- gekauft werden muß. Die benötigte Software wird mitgeliefert.

Quartet von ABC/Haba
Die Firma ABC (Advanced Business Computerproducts) stellte das von Haba Systems entwickelte und von ABC ins Deutsche übertragene Programm Quartet vor, das Tabellenkalkulation, grafische Darstellung, Textverarbeitung und „Datenbank" in sich vereint. Dabei wurde dieses Programm in erster Linie für den kleinen Betrieb (es läuft bereits auf dem 128K-Mac) konzipiert, bei dem nicht riesige Datenmengen oder ein umfangreicher Befehlssatz im Vordergrund stehen, sondern eine schnelle und leichte Handhabung erwünscht ist. So dürften die 62 Spalten mit je 999 Zeilen genauso ausreichend sein wie die drei grafischen Darstellungsmöglichkeiten (Torten-, Linien- und Balkengrafik). Für Texte können Fenster mit bis zu 1000 Zeichen angelegt werden. Die „Datenbank" verwaltet die Spalten als einen Datensatz, wobei nach zwei Kriterien sortiert werden kann.
Das Ausdrucken der Tabellen und Grafiken kann nur in einer einzigen Druckqualität erfolgen, wobei die Unterteilung in Einzelbilder, die später zusammengeklebt werden können, erst beim Druck ersichtlich wird.
Der Preis für das deutsche Programm beträgt ca. DM 1100,- inkl. MwSt.

Pagemaker von ABC/Aldus
Von ABC wurde auch das Programm Pagemaker der Aldus Corporation, Seattle, ins Deutsche übersetzt, mit dessen Hilfe der Umbruch von einzelnen Seiten einer Broschüre erstellt werden kann. Wenn diese Seiten dann mit dem Laserwriter ausgedruckt werden, erhält man Schriftqualitäten, die für anspruchsvollere Hausdrucksachen in jedem Fall ausreichen und selbst als reprofähige Vorlagen für kleine Auflagen von konventionell gedruckten Broschüren dienen können.
Die Ausgabe kann auf jedem Drukker erfolgen, der die Übertragungssprache Postscript versteht – also auch der Imagewriter.
Hierzu werden zunächst „Master Pages" angelegt, die immer wiederkehrende Elemente (Titel, Spaltenbegrenzungen, Kolumnenköpfe) enthalten können. In diese „Satzspiegel" werden dann Mac-Dokumente wie Grafiken, Tabellen und Texte einmontiert. Während dieser Montage können die einzelnen Teile vergößert oder verkleinert werden, um z.B. Abbildungen an die Spaltenbreite anzupassen.
Wenn ein Stück nicht auf die Seite paßt, kann es auf dem Ablageboard deponiert werden, um später in der nächste Seite Platz zu finden.
Zur optischen Aufbereitung steht eine „Toolbox" zur Verfügung, mit deren Hilfe grafische Element wie Linien, Rechtecke und Ovale in die Gestaltung aufgenommen werden können.
Auch soll die nachträgliche Bearbeitung von Schriftstücken möglich sein, was in der zur Demonstration vorgelegenen Version noch nicht vorgesehen war.
Der Preis liegt für die deutsche Version bei ca. DM 2100,-.

Excel von Microsoft
Die Firma Microsoft hatte Gelegenheit, ihr neues Tabellenkalkulationsprogramm Excel vorzustellen. Mit diesem Programm soll der Macintosh (mit 512K) zu ähnlichem Ruhm gelangen wie seinerzeit der Apple II mit dem Programm Multiplan (verständlicherweise wurde nicht auf Visicalc verwiesen).
Das ganz auf die Maussteuerung abgestimmte Programm, das Tabellenkalkulation, Business-Grafik und Datenverwaltung umfaßt, bietet umfangreiche Möglichkeiten, um Tabellen mit bis zu 16000 Zeilen und 256 Spalten zu erzeugen und sofort grafisch umzusetzen, wobei Änderungen in der Tabelle auch gleichzeitig in der Grafik vorgenommen werden.
Besondere Beachtung verdienen die Makros, die häufig zu wiederholende Aufgaben bewältigen können. Dabei sind keine Programmierkenntnisse erforderlich, vielmehr kann bei der Eingabe einer Befehlssequenz „mitgeschnitten werden", d.h. alle Befehle werden in einem Makro abgelegt, das später noch editiert werden kann. Damit bietet sich die Möglichkeit, branchenspezifische Lösungen als Makros zu erstellen – der völlig unbedarfte Anwender kann so, von jedem Ballast befreit, sofort in die Tabellenkalkulation einsteigen.
Diese Möglichkeit könnte von Händlern oder kleineren Software-Häusern aufgegriffen werden, um auch spezielle Kundenstämme befriedigen zu können.
Die Grafiken, die sehr unterschiedliche Darstellungsformen zulassen, können vor dem Ausdruck mit Laserwriter oder Imagewriter zu Repräsentationszwecken mit Texten versehen werden.
Die Datenkompatibilität mit anderen Programmen ist gewährleistet.
Die englische Version wird voraussichtlich in diesem Monat in den USA erhältlich sein. Die deutsche Version soll in einigen Monaten folgen. Der Preis wird in Deutschland etwas über DM 1600,- liegen.

Inserentenverzeichnis peeker 10/85

Demnächst im Peeker

Grafik

Trickfilmgrafik für Spielprogramme
Mit einem Sprite-Editor

Drucker

Großformatiger HGR-Ausdruck

Imagewriter kurz und bündig
Die wichtigsten Steuerbefehle

DOS 3.3

Disk-Chirurg
Ein Programm zur Überprüfung schadhafter Diskettensektoren

CP/M-Sonderteil

Z80-Assembler-Programmierung

Applesoft

Print-Using
Rechtsbündige Zahlenformatierung

Pascal

Tips und Tricks in Pascal
Teil 4: Die Compiler-Optionen

Kompaktkurs für UCSD- und Turbo-Pascal

Testberichte

- Beagle-Brothers-Utilities
- Triple-Dump
- Diversi-DOS 4.1-C
- BASF-Flexydisk 1X und 1D
- Microsoft-Premium-Softcard
- Star-Drucker SG-15

FÜR WERBEPLANER:

NEU

Die Profis der Mini-Mikrocomputer-Werbung haben jetzt ein sicheres Medium:
mini Micro magazin, die Fachzeitschrift mit Kontaktkarten.

mini Micro magazin wendet sich an alle Entwicklungsingenieure, Techniker und Mitglieder des technischen Managements, die als Systemintegratoren, OEM und Anwender fundierte Informationen aus dem gesamten Gebiet der professionellen Mini- und Mikrocomputertechnik benötigen.
mini Micro magazin berichtet über Minis, Mikros, jeder Art von Peripherie, Software und komplette Computersysteme, einschließlich „lokale" Netzwerktechnik, Datenfernübertragung, CAD/CAM/CAE und CIM.
Den Kern der Berichterstattung bilden anwendungsorientierte Fachbeiträge, aufbereitete Marktinformationen, aktuelle Produktneuheiten, Trendanalysen, internationale Korrespondentenberichte und interdisziplinäre Schwerpunktthemen.

COUPON

☐ Ja, ich bin interessiert das neue **mini Micro magazin** kennenzulernen. Bitte schicken Sie mir kostenlos und unverbindlich Ihr Infopaket.

Name

Firma

Funktion

PLZ/Ort

Telefon

Bitte ausschneiden und adressieren an:
mini Micro magazin
Verlagsgruppe Hüthig
Landsberger Straße 439
8000 München 60

mini Micro magazin will den Leser bei seiner Wahl der Hard- und Software-Komponenten für komplette Computerlösungen effektiv unterstützten und dabei helfen, die notwendigen Entscheidungsprozesse zeitlich zu verkürzen.
Kontaktkarten dienen zur qualifizierten Kontaktaufnahme mit Anbietern auf dem Feld der gesamten professionellen Mini- und Microcomputertechnik.

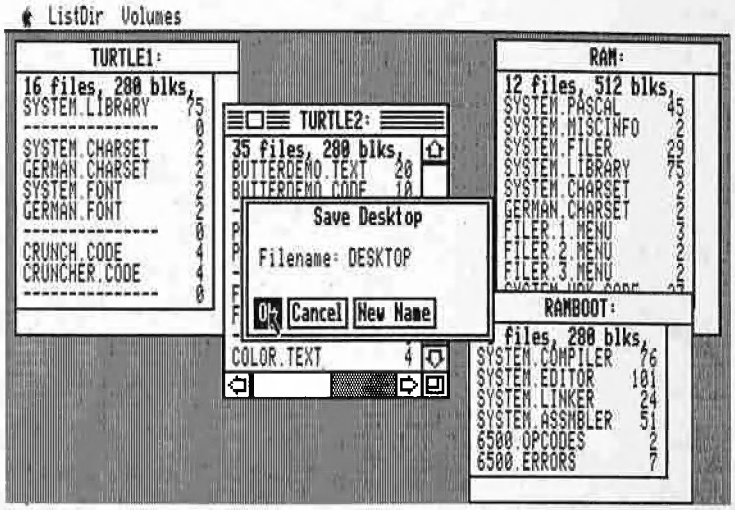
Hüthig
PUBLIKATION

In Vorbereitung

TurtleGraphics-Library-Paket

von Dieter Geiß

Turtle-Utilities für Fenstertechnik und Apple-Maus in einfacher und doppelter Hires-Grafik für Pascal 1.1/1.2 auf Apple II+/e/c mit Maus oder Joystick.
2 Disketten mit umfangreichem Manual, DM 98,–
Erscheinungstermin etwa Ende Oktober



Das Utility-Paket besteht aus vier Modulen, die von Programmierern benutzt werden können, um professionelle grafische Anwendungsprogramme in Pascal zu schreiben.

Benötigt wird ein Apple Pascal Betriebssystem, entweder die Version 1.1 oder die neue Version 1.2. Bestehende Programme laufen ohne Einschränkung mit der neuen „TurtleGraphics", wenn diese nicht zu viel Speicherplatz verbraucht haben, da die neue „TurtleGraphics" umfangreicher als die alte ist.

Im einzelnen bietet das Paket folgende Möglichkeiten:

- volle Kompatibilität mit der alten „TurtleGraphics"
- lauffähig auf Pascal 1.1 und 1.2
- funktionsfähig mit angeschlossener UltraTerm-Karte
- alle zeitkritischen Funktionen in reinem Assembler programmiert
- Benutzung der zweiten Hires-Seite ($4000–$5FFF) möglich
- für Apple IIc und Apple IIe mit erweiterter 80-Zeichen-Karte Benutzung der doppelten Hires-Grafik mit 560 × 192 Punkten bzw. 16 neuen Farben möglich
- schnelle Prozeduren zum Zeichnen eines Punktes oder einer Linie
- Linearisierung von Teilen des Hires-Schirms
- Benutzung mehrerer Zeichensätze gleichzeitig
- Laden und Speichern von Hires-Bildern mit Ausdruck über Pascal-SUPERDUMP
- Scrolling des Hires-Schirms oder eines Teils in vier Richtungen
- drei verschiedene Schriftarten: Fett-, Breit- und Proportionalschrift, beliebig mischbar (acht Möglichkeiten)
- spezielle schnelle Ausgabe von Text
- Cursor bei Eingabe frei programmierbar
- Ein-/Ausgabe von INTEGER-, CHAR-, STRING- und REAL-Werten im Grafikmodus
- Menüzeile wie beim Macintosh (Die nachfolgenden Module benötigen Maus/Joystick)
- Pull-down-Menüs
- Laden und Speichern von Fenstern (Windows)
- Öffnen von Fenstern
- Aktivieren und Deaktivieren von Fenstern
- Verschieben und Vergrößern/Verkleinern von Fenstern
- Scrolling von Fensterinhalten in allen vier Richtungen
- Umfangreiche Demos als Quelltexte.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg