Nr. 12/86 Dezember
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

peeker

MAGAZIN FÜR MIKROCOMPUTER

ProDOS-Dateikopierprogramm

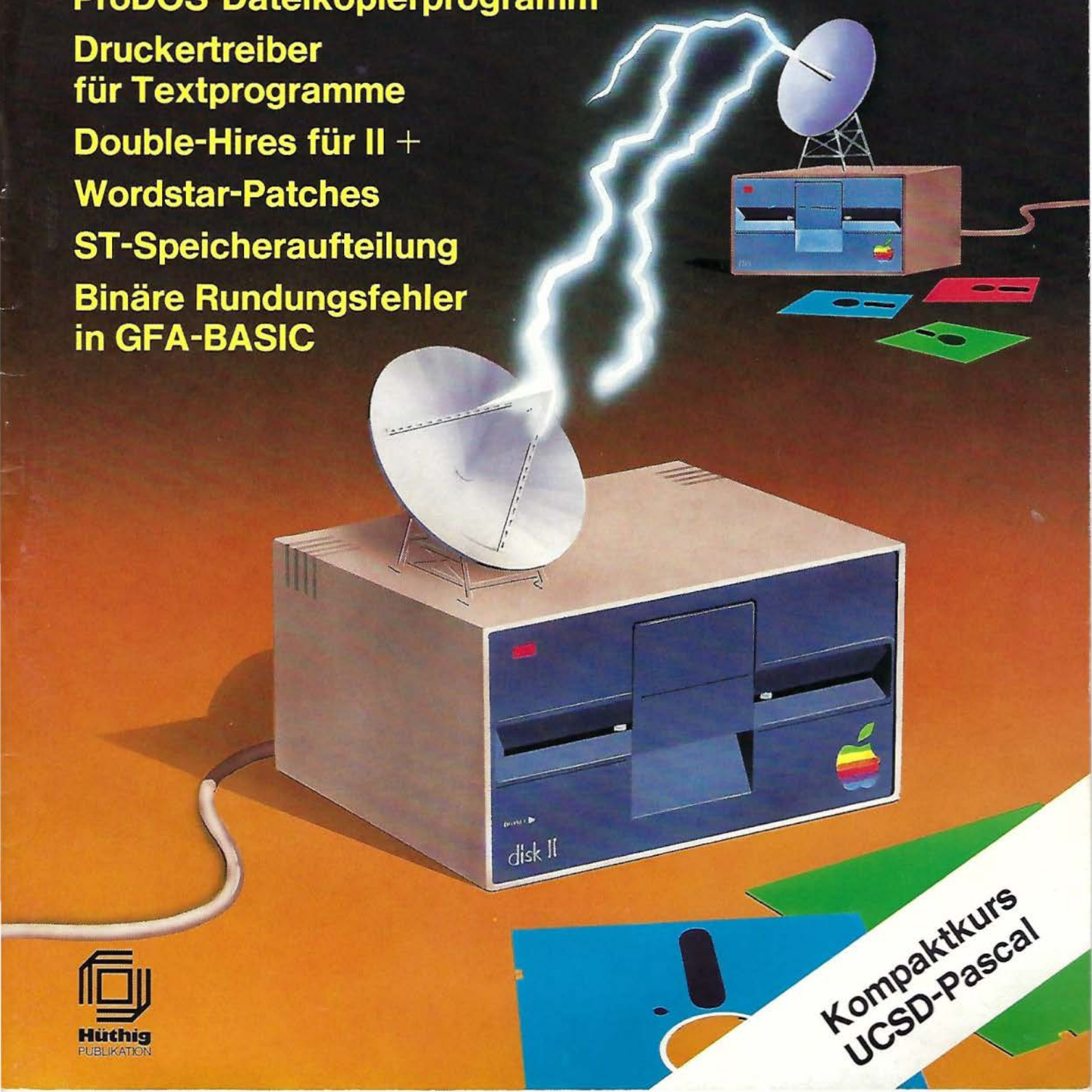
Druckertreiber
für Textprogramme

Double-Hires für II +

Wordstar-Patches

ST-Speicheraufteilung

Binäre Rundungsfehler
in GFA-BASIC



Kompaktkurs
UCSD-Pascal

Hüthig
PUBLIKATION

Preiswerte Apple-IIgs-Programme

Amerikanische Software-Häuser schreiben zur Zeit alte Apple-II-Software für den neuen IIgs um und entwikkeln darüber hinaus originäre IIgs-Produkte, die im Laufe der nächsten Wochen und Monate sukzessive erscheinen werden. Der Direktbezug aus den USA ist jedoch nicht nur umständlich, sondern auch kostspielig. Deshalb werden wir die interessanteren IIgs-Programme auf Lager nehmen und unseren inländischen Peeker-Abonnenten zu ungewöhnlich günstigen Sonderpreisen anbieten. Darüber hinaus können sich Abonnenten über uns amerikanische Spezialprogramme besorgen lassen.

Die nachfolgende Aufstellung vermittelt Ihnen einen kleinen Eindruck von einigen der in Arbeit befindlichen Programme, die demnächst erscheinen werden. Deshalb können im Moment auch noch keine Preise genannt werden, die wir erst nach Erhalt und Prüfung der Produkte festlegen und dann veröffentlichen werden. Wenn Sie bereits vor den monatlichen Peeker-Ausgaben über Produktneuheiten informiert werden wollen, so lassen Sie sich in unseren für Sie kostenlosen Verteiler von Vorabinformationen aufnehmen (Postkarte mit Abo-Nummer genügt). Übrigens liefern wir die Software in offener Rechnung und nicht in dem für Sie teuren und umständlichen Nachnahmeverfahren.

Multiscribe

Textverarbeitungsprogramm, das wahlweise mit Maus oder Cursortasten bedient werden kann und sich u.a. durch eine Fülle von Sonderzeichensätzen auszeichnet. Bereits früher auf dem Apple II unter ProDOS erzeugte ASCII-Textfiles können übernommen werden. Ausdruck über Image- oder Laserwriter. Matrixdrucker von Epson usw. bedingt einsetzbar. Produzent: Styleware.

Mousewrite

Textverarbeitungsprogramm, das wahlweise mit Maus oder Cursortasten bedient werden kann und sich u.a. durch ein mitgeliefertes Modem-Programm auszeichnet. Ausdruck über Imagewriter-, Epson- und andere Drucker. Produzent: Roger Wagner Publishing.

VIP Professional

Integriertes Programmpaket, bestehend aus Tabellenkalkulation, Datenbank und Geschäftsgrafik. Produzent: VIP Technologies

Topdraw

Komfortables Zeichenprogramm für geometrische Gebilde und Freihandzeichnungen. Ausdruck über Image- und Laserwriter. Produzent: Styleware

ASCII-Express

Kommunikations- bzw. Modemprogramm mit diversen Übertragungsprotokollen. Produzent: Roger Wagner Publishing

Kyan-Pascal-Compiler

65816-Pascal, das Assembler-Zwischenquelltext erzeugt. Produzent: Kyan-Software

TML-Pascal-Compiler

65816-Pascal. Produzent: TML Systems

Merlin-816-Assembler

65816-Version des beliebten Assemblers. Produzent: Roger Wagner Publishing

Pinpoint-Accessories

Sammlung von Accessories, d.h. speicherresidenten Hilfsprogrammen, z.B. Terminkalender, Minitextprogramm (Notizblock), Vierspeziestaschenrechner usw. Produzent: Pinpoint

Hüthig Software Service · Postfach 102869 · 6900 Heidelberg 1

EDITORIAL

Die behutsame Erweiterung des Peekers in Richtung auf andere Computer hat neben Zustimmung natürlich auch Kritik gefunden. Ich möchte Ihnen deshalb von einigen Schlüsselerlebnissen berichten, die selbst eingefleischten Apple-Anhängern zu denken geben dürften:

– Vom größten Apple-Club in den USA, nämlich Call A.P.P.L.E., würde man normalerweise annehmen, daß er die reine Apple-Linie vertritt, zumal der Apple in den USA viel stärker als beispielsweise in der Bundesrepublik verbreitet ist. Im Juli-Heft der gleichnamigen Club-Zeitschrift wurde dann jedoch quasi beiläufig mitgeteilt, daß man nunmehr offizieller Amiga-Händler sei. Die Begründung liest sich dann so (7/86, S. 10): „Although traditionalists may wonder how a group whose very initials (Anm.: A.P-.P.L.E. = Apple PugetSound Program Library Exchange) proclaim its roots can dare to dally with the competition, we remind you that our primary mission is to serve you, the computer user, rather than any single corporate giant. In order to thrive, our organization must respond to the changes in your needs, wherever that may lead..."

– Die AUGE als ehemals lupenreiner deutscher Apple-Club hat sich seit kurzem in Richtung auf Fremdcomputer geöffnet. In einer Mitteilung vom 15.9.86 liest sich dies dann so: „Mit der Einrichtung von Arbeitsgemeinschaften für Atari-ST- und MS-DOS-Rechner wird jetzt die Unterstützungsleistung des Clubs auf die derzeit wichtigsten Computertypen ausgedehnt." Weiter heißt es an anderer Stelle: „Ich muß als Vorsitzender einen dynamischen Kurs fahren, sonst treten unsere Mitglieder aus... Es ist nur noch eine Frage der Zeit, bis der Markenname aus der Vereinsbezeichnung gestrichen wird." Für das nächste Club-Heft ist bereits ein Interview mit Atari-Deutschland angekündigt.

Eine gerätespezifische Computerzeitschrift ist nur dann sinnvoll, wenn es genügend Gerätebesitzer gibt. Für den Macintosh, der sich an Spezialzielgruppen wendet, dürfte es deshalb in Deutschland nie eine Fachzeitschrift geben, wenn man von der kostenlosen Werbebroschüre Macup absieht. Entscheidend für die Erweiterung des Peekers war deshalb nicht der Mac, sondern der IIgs. Auch hier möchte ich Ihnen von zwei Schlüsselerlebnissen berichten:

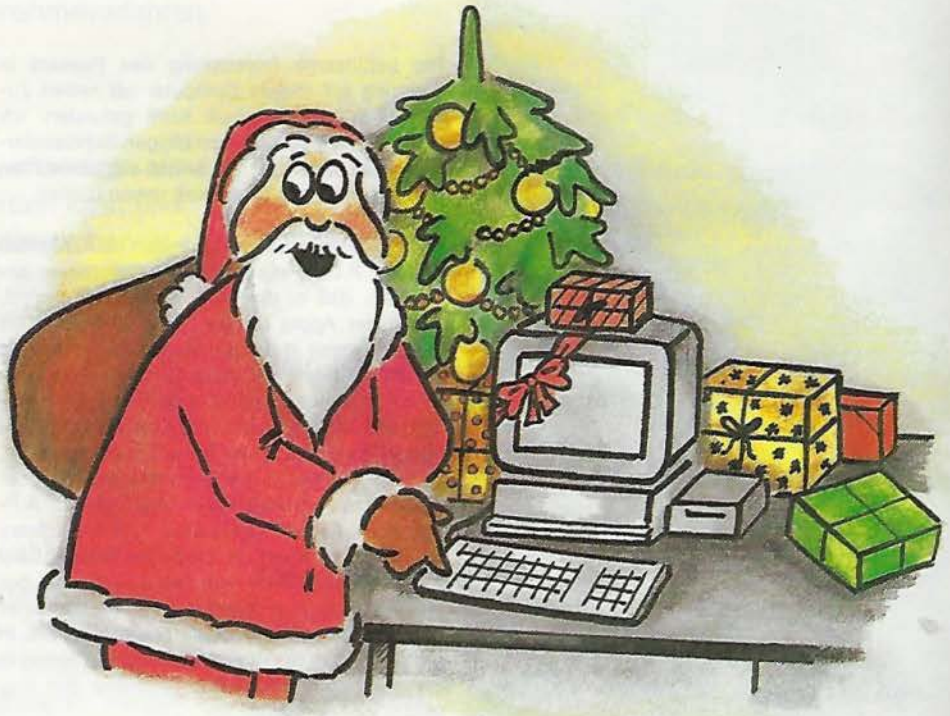
– Als wir Mitte August bei Apple-München den IIgs testeten, war niemand da, der sich mit dem IIgs auskannte. Selbst die Umstellung der amerikanischen Tastatur auf deutschen Zeichensatz mußten wir Mangels Hilfe selbst ausknobeln. Daraus schließen wir: Wenn Sie einen technischen Rat zum IIgs benötigen, müssen Sie sich selbst beraten.

– Als wir uns dann nach dem strategischen Stellenwert des IIgs erkundigten, wurde der IIgs wörtlich als „Übergangsgerät" bezeichnet, der „keine Priorität" genießt. Folgerichtig ist denn beispielsweise auch die Pressekonferenz entfallen. Daraus schließen wir: Wenn Sie einen IIgs kaufen wollen, müssen Sie sich selbst werben.

Der typische Peeker-Leser als der schon seit Jahren loyale Apple-Besitzer wird von Apple-München völlig ignoriert. Und auch die knapp elf Dutzend verbliebenen Apple-Händler werden dem IIgs nicht aus eigener Kraft zum Verkaufserfolg verhelfen können. Die weitere Entwicklung kann sich jeder selbst ausmalen. Trotzdem werden wir Altgeräte-Besitzer weiterhin unterstützen und auch über den IIgs mehr als jede andere Zeitschrift berichten. Wir werden sogar über den Hüthig Software Service IIgs-Programme vertreiben, weil diese in Deutschland kaum erhältlich sein werden. Wir werden jedoch keine reine Apple-Zeitschrift mehr sein.

Ulrich Stiehl

INHALT



Impressum

Peeker
3. Jahrgang 1986
ISSN 0176–9200

Verleger und Herausgeber:
Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften:
Heinz Melcher
Chefredakteur: Ulrich Stiehl (us)
Redaktion: Dagmar Berberich
Anzeigenleitung: Karl M. Dietzow
Anzeigendisposition: Diana Walter

Telefonnummern:
Zentrale: 06221/489-0
Redaktion: 06221/489-352
Anzeigen: 06221/489-206
Abonnement: 06221/489-283
Software: 06221/489-231
Bücher: 06221/489-353
(Bestellungen bitte nur schriftlich)

Abonnement:
Der Abonnent kann seine Bestellung innerhalb von 7 Tagen schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag GmbH, Postfach 102869, 6900 Heidelberg 1, widerrufen. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels). Das Abonnement verlängert sich zu den jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Jahresende schriftlich gekündigt wird. Die Abonnementsgelder werden jährlich im voraus in Rechnung gestellt, wobei bei Teilnahme am Lastschriftabbuchungsverfahren über die Postscheckämter und Bankinstitute eine vierteljährliche Abbuchung möglich ist.
Nichterscheinen infolge höherer Gewalt berechtigt nicht zu Ansprüchen gegen den Verlag.

Hüthig
PUBLIKATION

PEEKER

Heft 12/1986

Anschrift:
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.
Telefax (0 62 21) 489 279
BTX * 51851 #

Auslieferung für die Schweiz:
Delta-Verlag
Herr R. de Forest
Gugelmattstraße 31
8967 Widen
Telefon 057 / 33 86 86

Vertrieb:
Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement Inland DM 72,–, einschl. MwSt und Versandkosten.
Jahresabonnement Ausland DM 72,– plus DM 18,– Versandkosten.
Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089 / 31 90 06 13, Telex 0 522 656
Vertriebsleitung:
Walter Menzel, Tel. (0 62 21) 48 92 80

Bankverbindungen:
Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postgirokonten: Ludwigshafen 4799-673, BLZ 545 100 67; Österreich: Wien 7555 88; Schweiz: Basel 40-24417-4; Niederlande: Den Haag 1 457 28; Italien: Mailand 5 968 92 08; Belgien: Brüssel 07 230 26-85; Dänemark: Kopenhagen 603 4969; Norwegen: Oslo 199 4243; Schweden: Stockholm 5477 76-5
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 02 65 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

Herstellung:
Produktionsleitung: Gunter Sokollek
Gestaltung: Rainer Schmitt
Titelbild: Werner Hable
Satz und Druck:
Heidelberger Verlagsanstalt
Printed in Germany

FILECOPY

Eine Befehlserweiterung des BASIC.SYSTEMs

von Arne Schäpers

Dem BASIC.SYSTEM fehlen, hauptsächlich wegen Mangel an Speicherplatz, einige Kommandos, die bei ernsthafter Programmierarbeit fast ständig benötigt werden. Zwei der wichtigsten Erweiterungen, nämlich TYPE zur Ausgabe von Textdateien auf den Bildschirm und MON/NOMON zur Überprüfung des Dateiverkehrs, sind in [1] ausführlich besprochen worden.

Dieser Artikel beschäftigt sich mit einer dritten Befehlserweiterung, nämlich dem Kopieren von Dateien. In [3] ist zwar bereits ein Dateikopierprogramm namens PROFID veröffentlicht worden, das jedoch erstens nicht in das BASIC.SYSTEM eingebunden und zweitens nur für Besitzer von 2 Laufwerken gedacht ist. Das nachfolgende Programm FILECOPY kann hingegen auch von 1-Drive-Besitzern verwendet werden und läßt sich zudem in andere BASIC.SYSTEM-Erweiterungen, z.B. den PRODOS.EDITOR, einbinden (1. BRUN PRODOS.EDITOR, 2. BRUN FILECOPY).

Normalerweise sind unter dem BASIC.SYSTEM für das Kopieren von Dateien folgenden Schritte notwendig:

- Speichern des Applesoft-Programms, das sich momentan in Arbeit befindet;
- Beenden des BASIC.SYSTEMs und Start des FILERs;
- Durchführen des Kopierens;
- Beenden des FILERs und Start des BASIC.SYSTEMs;
- Laden des Applesoft-Programms und Weiterarbeiten.

Falls nur ein Diskettenlaufwerk verfügbar ist, macht einem der FILER außerdem die Kopierarbeit nicht gerade einfach, weil das Programm einen geradezu erstaunlichen Umfang hat und nur 16K als Pufferspeicher übrigbleiben. Sollte die zu kopierende Datei länger sein, ist eine entsprechende Anzahl von Diskettenwechseln erforderlich.
Die Alternative dazu sieht nach einem Start von FILECOPY wesentlich benutzerfreundlicher aus. Sie besteht aus der Eingabe von:

COPY Quell-Pfadname, Ziel-Pfadname

Ein eventuell geladenes Applesoft-Programm und seine Variablen bleiben dabei erhalten. Wenn LOMEM entsprechend gesetzt ist, bleiben zusätzlich die Hires-Grafikseiten unberührt.

1. Leistung des Programms

a) FILECOPY enthält keine „Tricks" und benutzt die Standardschnittstelle zu ProDOS. Das Programm arbeitet deshalb mit beliebigen ProDOS-Volumes einschließlich Festplatten und RAM-Karten (/RAM usw.).
b) Außer DIR kann im Prinzip jeder Dateityp kopiert werden (also auch SYS oder z.B. eine AppleWorks-Datei).
c) Die zu kopierende Datei kann eine beliebige Länge haben. Falls sie nicht auf einmal in den verfügbaren Speicher paßt, führt das Programm die Kopie in mehreren Schritten aus.
d) Folgende Arten von Kopien sind möglich:

– innerhalb desselben Subdirectory (z.B. /VOL.A/SUBDIR/Datei.A nach /VOL.A/SUBDIR/Datei.B);
– innerhalb eines Volume von einem Subdirectory in ein anderes (z.B. /VOL.A/SUBDIR.A/Datei.A nach /VOL.A/SUBDIR.B/Datei.B);
– von einem Volume zu einem anderen (z.B. /VOL.A/Datei.A nach /VOL.B/Datei.B). Falls nur ein Laufwerk zur Verfügung steht, gibt das Programm zu den entsprechenden Zeitpunkten eine Aufforderung zum Diskettenwechsel aus.

e) FILECOPY hängt sich beim Start an eventuell bereits geladene Befehlserweiterungen des BASIC.SYSTEMs an – im Gegensatz zu HELP und APA, die andere Befehlserweiterungen einfach überschreiben.
f) Als Puffer für die gelesenen Daten wird der Bereich von STREND (oberes Ende der Applesoft-Variablentabellen) bis FRETOP (unteres Ende der Stringvariablen) benutzt. Wenn zum Schutz einer Grafikseite der Befehl
LOMEM: Adresse
gegeben wurde (und Applesoft seine Variablentabellen damit oberhalb des geschützten Bereiches aufbaut), dann läßt FILECOPY die Grafikseite(n) damit ebenfalls unzerstört.
g) Das Programm ist eine „echte" Befehlserweiterung des BASIC.SYSTEMs und kann auch von einem laufenden Applesoft-Programm aus benutzt werden, z.B. mit
PRINT CHR$(4); "BRUN FILECOPY"
Auftretende Fehler lassen sich mit ONERR GOTO abfangen.

2. Bedienung des Programms

ProDOS-Diskette mit FILECOPY einlegen und den Befehl -FILECOPY oder BRUN FILECOPY geben. Das Programm wird in den Bereich der ersten Hires-Grafikseite geladen ($2000..$25xx) und gestartet. Innerhalb der Startphase werden drei Schritte ausgeführt:

– Herabsetzen von HIMEM um $500 und Verschiebung aller Stringvariablen von Applesoft um diesen Betrag nach unten. (Achtung: Wenn HIMEM vorher auf einen Wert < 11008 gesetzt wurde, zerstört sich das Programm durch diese Verschiebung selbst.)

– Relokalisierung in den durch die Verschiebung von HIMEM freigewordenen Bereich oberhalb der Dateipuffer des BASIC.SYSTEMs.

– Anhängen an den Kommando-Interpreter des BASIC.SYSTEMs bzw. an eine bereits geladene Befehlserweiterung.

Solange ein Applesoft-Programm den Speicherbereich $2000..$2500 nicht benutzt, bleibt es unverändert. Der Start von FILECOPY erweitert das BASIC.SYSTEM um zwei Befehle:

- COPY zum Kopieren von Dateien (s.u.);
- NOCOPY entfernt FILECOPY wieder aus dem Speicher und verschiebt HIMEM auf den alten Platz zurück. Ein Applesoft-Programm und seine Variablen bleiben unverändert.

FILECOPY erwartet entweder einen vollständigen Pfadnamen oder benutzt ein gesetztes PREFIX. Die folgenden Kombinationen sind möglich:

COPY/VOL.A/Datei.A,/VOL.B/Datei.B
Da hier beide Dateinamen vollständig angegeben wurden, wird ein eventuell gesetztes Präfix ignoriert. Je nachdem, ob ein oder zwei Laufwerke zur Verfügung stehen, ergibt sich der folgende Ablauf:

– 2 Laufwerke, in dem einem befindet sich /VOL.A, in dem anderen /VOL.B: Lesen von Datei.A und Kopie nach Datei.B ohne weitere Rückmeldung. Falls Datei.A nicht gefunden werden kann oder Datei.B bereits existiert, endet das Programm mit einer entsprechenden Fehlermeldung.

– 1 Laufwerk: In diesem Laufwerk muß sich zum Zeitpunkt des COPY-Befehls die Diskette mit der Quell-Datei (/VOL.A) befinden. FILECOPY liest Datei.A, gibt die Meldung „/VOL.B EINLEGEN.." aus und wartet auf eine Bestätigung. Danach wird Datei.B erzeugt und beschrieben. Falls die Kopie nicht in einem Durchgang erfolgen kann, folgt darauf die Meldung /VOL.A EINLEGEN usw.

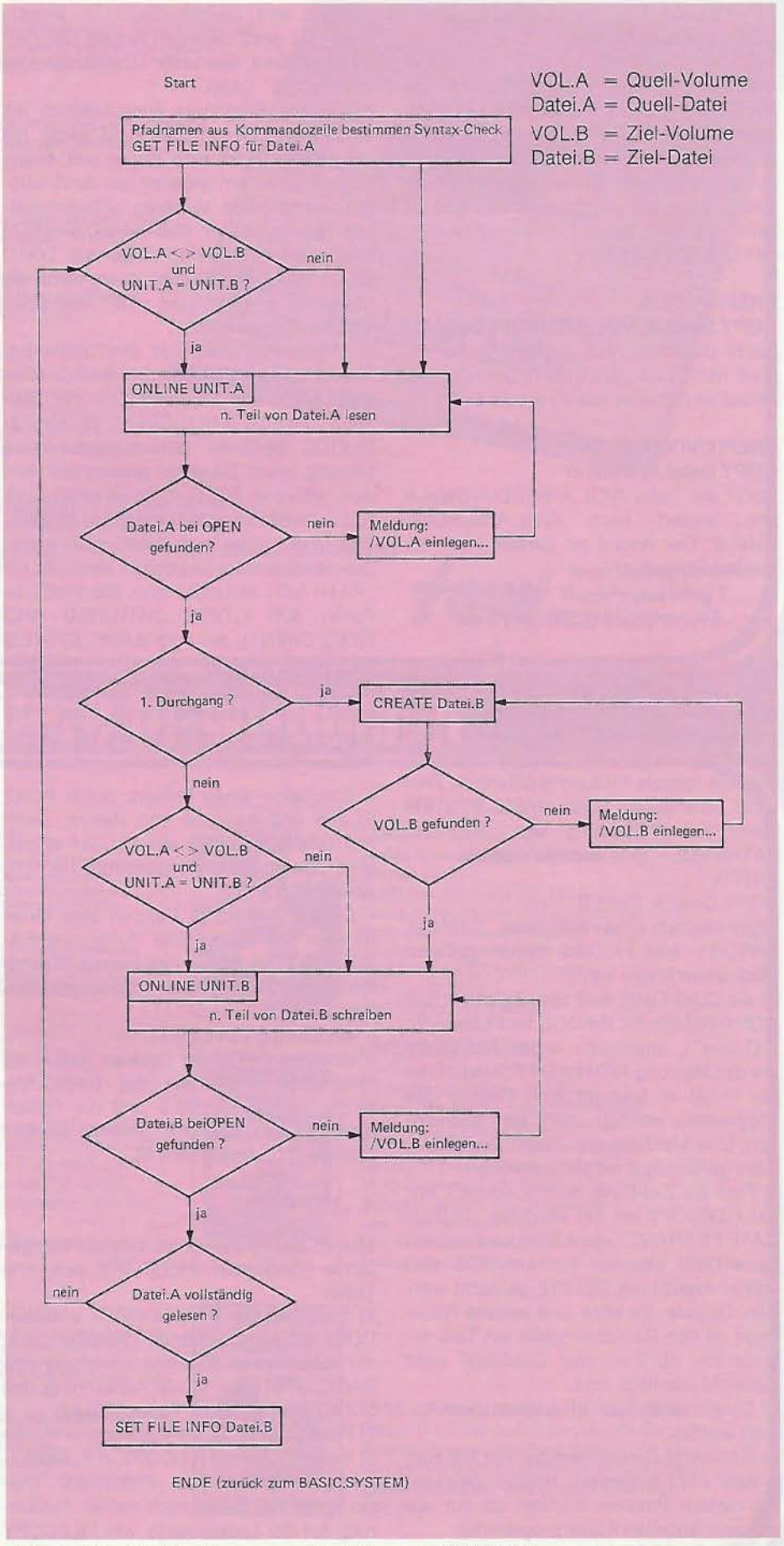


Abb. 1: Vereinfachter Programmablaufplan von FILECOPY.

PREFIX/VOL.A
COPY Datei.A, Datei.B
Nachdem das Präfix auf /VOL.A gesetzt und für die Quell- und Zieldatei keine vollständigen Pfadnamen angegeben wurden, sucht FILECOPY die Datei /VOL.A/Datei.A und kopiert ihren Inhalt in /VOL.A/Datei.B. Ein Unterschied im Betrieb mit einem oder zwei Laufwerken ergibt sich naturgemäß nicht, weil sich beide Dateien auf derselben Diskette befinden.

PREFIX/VOL.A
COPY Datei.A,/VOL.B/SUBDIR/Datei.B
Sucht die Datei /VOL.A/Datei.A und kopiert nach /VOL.B/SUBDIR/Datei.B. Der Ablauf ist derselbe wie im ersten Beispiel.

PREFIX/VOL.A/SUBDIR
COPY Datei.A, Datei.B
Sucht die Datei /VOL.A/SUBDIR/Datei.A und kopiert nach /VOL.A/SUBDIR/Datei.B. Der Ablauf ist derselbe wie im zweiten Beispiel.

3. Grenzen von FILECOPY

a) Um die Bearbeitung der Pfadnamen nicht noch aufwendiger ausfallen zu lassen als sowieso bereits geschehen (die entsprechenden Routinen machen rund ein Drittel des gesamten Programms aus), wird die formale Prüfung größtenteils ProDOS überlassen. Das BASIC.SYSTEM kennt keine Meldung wie „ILLEGAL PATHNAME". Eine Befehlsfolge wie
PREFIX /
COPY Datei.A, Datei.B
ergibt deshalb einen schlichten „SYNTAX ERROR", weil ProDOS keinen gültigen Pfadnamen bilden kann.
b) die Quell-Datei muß zum Zeitpunkt des COPY-Befehls für ProDOS verfügbar sein („Online"), ansonsten endet FILECOPY mit der Meldung PATH NOT FOUND. Dieser Punkt ist weniger eine Grenze des Programms, sondern mehr eine Konvention. Eine Meldung wie „/xxx EINLEGEN" wäre genausogut möglich gewesen.
c) Falls die Ziel-Datei bereits existiert, endet FILECOPY mit der Meldung „DUPLICATE FILENAME". Eine eventuell vorhandene Datei gleichen Namens muß also vorher explizit mit DELETE gelöscht werden. Gründe: Es wäre eine weitere Rückfrage an den Benutzer sowie ein Test erforderlich, ob Ziel- und Quelldatei nicht vielleicht identisch sind.
d) Dateien vom Typ DIR können nicht kopiert werden.
e) RANDOM-Dateien werden nur bis zum ersten nicht-existenten Record gelesen. An diesem Problem kranken so gut wie alle sequentiellen Kopierprogramme.
f) Wenn FILECOPY bei einer Aufforderung zum Diskettenwechsel mit Ctrl-C abgebrochen wird, bleiben die bis zu diesem Zeitpunkt geschriebenen Anteile der Ziel-Datei erhalten, also unter Umständen eine „halbfertige" Datei.
g) Da das Programm ausschließlich mit Volume-Namen arbeitet (und nicht mit „Slot/Drive"), ist eine Kopie von einem Volume zu einem anderen nur dann möglich, wenn beide Volumes unterschiedliche Namen haben. Das BASIC.SYSTEM ermöglicht eine Befehlsfolge wie LOAD XX,D1 SAVE XX,D2 auch dann, wenn die Disketten in „D1" und „D2" dieselben Volume-Namen haben.
h) Wenn zum Zeitpunkt des COPY-Befehls andere Dateien (außer EXEC) offen sind, bricht das Programm mit der Meldung „FILE(S) STILL OPEN" ab. Grund: ProDOS benimmt sich ausgesprochen bösartig, wenn Disketten gewechselt werden, während WRITE-Dateien offen sind. EXEC-Dateien werden immer nur gelesen. Das Schlimmste, was dabei nach einem Diskettenwechsel passieren kann, ist ein „PATH NOT FOUND" bzw. der ProDOS-Fehler $2E („DISK SWITCHED AND FILES OPEN"), der vom BASIC.SYSTEM unerfreulicherweise mit „I/O ERROR" übersetzt wird [2]. Hier wäre ein Ansatzpunkt für Verbesserungen des Programms, die aber alles andere als „legal" sind:

– Simulation eines Fehlers durch POKE 51,255 und Ausgabe von Return, damit das BASIC.SYSTEM ein FLUSH sämtlicher offener Dateien durchführt (s. [1], Abschnitt 6.1.3);
– CLOSE sämtlicher Dateien über einen direkten MLI-Aufruf. Die Puffer des BASIC.SYSTEMs sowie das interne Pfadnamenverzeichnis bleiben dabei unverändert;
– Ausführung von COPY;
– erneutes OPEN der Dateien, wobei die Pfadnamen direkt aus der BASIC.SYSTEM-Tabelle FNAMES und die Pufferadressen aus BUFTAB gelesen werden müssen ([1], Abschnitt 4.4).

4. „EXFRAME"

Wie im Listing zu sehen, besteht das gesamte Programm FILECOPY aus drei Teilen:
1) Erkennen der Befehle COPY und NOCOPY mit entsprechenden Routinen, d.h. der eigentlichen Befehlserweiterung des BASIC.SYSTEMs, sowie Ausführung des „NOCOPY"-Befehls (Routine QUIT);
2) Programm FILECOPY selbst;
3) Installation von FILECOPY, d.h. Relokalisieren, Kopieren und „Einklinken". Dieser dritte Teil bleibt nach seiner Ausführung auf der Ladeadresse von FILECOPY stehen und wird folglich nicht mitverschoben.

Die Teile 1 und 3 sind für alle Befehlserweiterungen des BASIC.SYSTEMs gleich und werden quasi als Rahmen („EXtension FRAME") grundsätzlich benötigt – es sei denn, man begnügt sich mit einem absoluten Speicherbereich und nimmt in Kauf, daß eventuell bereits geladene Erweiterungen einfach „übermangelt" werden.

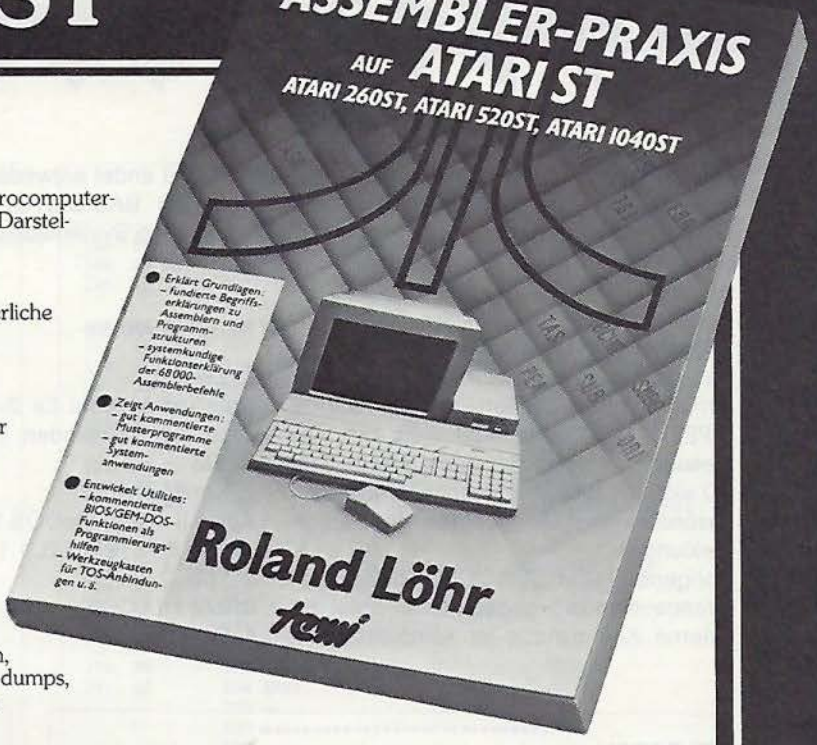
Um Assemblerprogrammierern das Leben etwas leichter zu machen, wurde EXFRAME so ausgelegt, daß zur „Verpakkung" einer eigenen Befehlserweiterung nur die folgenden Schritte notwendig sind:
- Einsetzen der Gesamtzahl der Kommandos im Listing T.FILECOPY in Zeile 33 (SCANCMD LDY #??);
- Einsetzen der Sprünge zur Kommandoausführung in die Tabelle CMDJMPS (Zeile 74ff), der erste Sprung ist für die QUIT-Routine reserviert;
- Einfügen des eigenen Programms ab Zeile 118;
- Einfügen der Parameter-Bits in die Tabelle CBITS (ab Zeile 561), die ersten beiden Bytes sind für die QUIT-Funktion reserviert;
- Einsetzen der Befehlsnamen in die Tabelle CMDS (ab Zeile 564), der erste Name steht für die QUIT-Funktion.

Alle anderen Schritte (Bestimmung des benötigten Speicherplatzes, Verschiebung von HIMEM, Relokalisierung, Behandlung von dabei auftretenden Fehlern, „Einklinken", Kommando-Interpretation und QUIT) werden durch EXFRAME bzw. den Assembler erledigt.

Das Thema „Befehlserweiterungen und EXFRAME" ist in [1] auf rund 40 Seiten mehr als ausführlich behandelt. Wir gehen deshalb an dieser Stelle nur auf die wichtigsten Punkte ein:
1) Das BASIC.SYSTEM liest die ersten acht Zeichen einer Eingabe, übersetzt auf Großbuchstaben und entfernt Leerzeichen. Das Ergebnis wird in TXBUF ($BCBD) gespeichert. Auf diesen Puffer greift die Erkennung des Kommandowortes („SCANCMD") zu. (TXBUF ist ein „offizielles" Label).
2) FILECOPY setzt den Wert $00 00 als PBITS. Das BASIC.SYSTEM nimmt damit (wie bei FRE, NOMON und BYE) überhaupt keine weitere Auswertung der Kommandozeile vor.
3) Die angegebenen Pfadnamen werden weder vervollständigt noch umkopiert. Die Routine GETPATHS bestimmt lediglich ihre Startadresse und setzt entsprechende Längenbytes davor.
4) Das BASIC.SYSTEM setzt vor der Ausführung eines Befehls die „tatsächlichen" Ein-/Ausgabevektoren auch dann, wenn es sich um ein externes Kommando han-

ATARI ST

ASSEMBLER-PRAXIS AUF ATARI ST



Roland Löhr
...ein Altmeister der Assembleranwendung, Herausgeber des Mikrocomputer-Magazins MICRO MAG, veröffentlicht bei te-wi seine souveräne Darstellung der Assemblerprogrammierung auf ATARI STs.

Erklärt Grundlagen:

Begriffe und Werkzeuge der Assemblerprogrammierung...erforderliche Systemkenntnisse...systembezogene Erläuterung der 68000er Befehlsfunktionen.

Zeigt Anwendungen:

Hantieren mit Assemblern: Aufruf von Assemblern; Steuern ihrer Optionen über Direktiven; Stellungnahme zu realen ATARI-ST-Assemblern.
Arbeiten in der ATARI-ST-Programmierumgebung: Textprogramme zur Programmentwicklung; ein Editor; ein Parser; das Betriebssystem; BIOS-Funktionen; BIOS-Toolbox; GEMDOS Toolkit; das erweiterte XBIOS.
Anwenden des Befehlssatzes in Musterprogrammen für: E/A-Routinen, Rekursionen, dez/bin Rechenarten, Stackverwaltung, Adressverwaltung, Entscheidungen, Schleifenkonstrukte, Unterprogramme, numerierte Traps, Bedienen von Interfacebausteinen, Texterkennung, Textverarbeitung, Tastaturdekodierung, memory dumps, Floppy-Tests/Funktionen, serielle RS232-Datenübertragung usw.

Entwickelt Hilfsprogramme:

BIOS-Toolbox; GEMDOS-Toolkits; ein Editor; ein Parser; Arbeiten mit Toolkits.
Die Programme des Buchs sind auf Diskette vom Autor erhältlich.

Ein Fachtext in klarer Sprache mit leserfreundlichem Druckbild, guter Bilddokumentation und umfangreichen Listings von Musterprogrammen (auf Diskette beim Autor erhältlich).

ca. 300 Seiten, Softcover, DM 59,–

tewi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weitere te-wi-Bücher

NEU

DAS „C"-BUCH
(Herold / Unger)
Ein „C"-Kurs der Industrie. Für sämtliche C-Konstrukte. Über 100 Beispiele. Anspruchsvoll in Text/Bildmaterial, ca. 500 Seiten, Softcover, DM 79,–

UNIX
(Yates/Thomas) US-Standardwerk der UNIX-Promoterin Yates. Eine sachkundige Übersicht und Einführung in die Anwendung, 550 Seiten, Softcover, DM 79,–

M68000 FAMILIE, 2 Bd.
Hilf/Nausch, ges. 968 Seiten
Einzige Motorola-authentische Darstellung von CPU-68000-Architektur, Programmierung, Systemaufbauten. Behandelt alle 68000-Bausteine sowie 68020, 68881.
Bd 1, Grundlagen + Architektur, 568 Seiten, DM 79,–
Bd 2, Anwendung und Bausteine, 400 Seiten, DM 69,–

Mein ATARI Computer
Best-rezensiertes Standardwerk deutscher ATARI-User-Groups. Kompakte ATARI 400-/800-System/Peripheriebeschreibung.
Von Poole/McNiff/Cook, 500 Seiten, Softcover, DM 59,–

Sprühende Ideen mit ATARI-GRAPHIK
Fröhlicher Lehrstoff in Geometrie und Farbenlehre eines amerikanischen Lehrers mit ATARI Graphikmöglichkeiten.
Von Tom Rowley, 224 Seiten, Softcover, DM 49,–

LOGO –
Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den USA.
Best-rezensiert von Pädagogen und deutschen Kultusministerien. Ein bildreicher Führer durch u. a. ATARI's LOGO. Von Papert's Schüler D. Watt.
384 Seiten, A4, DM 59,–

UMWELTDYNAMIK
30 Programme für kybernetische Umwelterfahrungen auf allen BASIC-Rechnern. Das Buch enthält beides: Ein Programmsystem zur Simulation eigener Problemformulierungen und 29 kommentierte Modellbeispiele wie Baumsterben, Heizungsbedarf, Nahrungsketten usw. Prospekt anfordern.
Von Hartmut Bossel, 480 Seiten, Softcover, DM 59,–

6502 – Programmieren in Assembler
Dieses Buch behandelt ausführlich die Assemblersprachen-Programmierung für den weitverbreiteten Mikroprozessor 6502.
Lance Leventhal, 704 Seiten, Softcover, DM 59,–

tm 4645

Noch im Programm: Einführung in die Mikrocomputer-Technik, DM 66,–
Computer für Kinder, ATARI, DM 29,80

delt. Auf diese Weise kommen sich interaktive Befehle und die Überwachungsroutinen des BASIC.SYSTEMs nicht ins Gehege.
5) Der Rücksprung am Ende der Ausführung von COPY führt in das BASIC.SYSTEM. Wenn das Carry-Flag gesetzt ist, wird der Inhalt des Akkumulators als Fehlernummer interpretiert und in $00DE („PEEK (222)") gesetzt. Falls ein Applesoft-Programm läuft und ONERR GOTO aktiv ist, wird entsprechend verzweigt; ansonsten folgt die Ausgabe der Fehlermeldung.
Übrigens: Das im „ProDOS Technical Reference Manual" gegebene Beispiel für externe Kommandos ist kompletter Unsinn und endet entweder mit dem Abhängen des BASIC.SYSTEMs oder einem kompletten Systemabsturz.

Kurzhinweise

1. Zweck:
Dateikopierbefehl für 2- und 1-Drive-Besitzer zum Einbinden in das BASIC.SYSTEM
2. Konfiguration:
Apple II+/e/c; ProDOS 1.0.1, 1.0.2, 1.1.1, BASIC.SYSTEM 1.0, 1.1
3. Test:
BRUN FILECOPY
4. Sammeldisk:
T.FILECOPY (Big-Mac-Quelltext)
FILECOPY (Objektcode)
5. Sonstiges:
Die Dateien müssen zunächst mit CONVERT oder DOSTOPRO von der DOS-3.3-Sammeldisk auf Ihre ProDOS-Arbeitsdiskette konvertiert werden.

Literatur

[1] A. Schäpers, Das BASIC.SYSTEM. Lesebuch, Analyse, Tips und Tricks, Heidelberg 1986
[2] A. Schäpers, ProDOS-Analyse, Heidelberg 1985
[3] U. Stiehl, ProDOS für Aufsteiger, Band 2, Heidelberg 1985

FILECOPY

```
BSAVE FILECOPY, A$2000,L1312

                    1  * FILECOPY
                    2  COUT     EQU  $FDED
                    3  *
                    4  EXTRNCMD EQU  $BE06      ;Sprung zu SCANCMD
                    5  ERROUT   EQU  $BE09      ;Meldung und Warmstart
                    6  XTRNADDR EQU  $BE50      ;Sprung zur Ausführung
                    7  *
                    8  XLEN     EQU  $BE52      ;Kommandowortlänge-1
                    9  XCNUM    EQU  $BE53      ;Kommandonummer
                   10  PBITS    EQU  $BE54      ;erlaubte Parameter
                   11  FBITS    EQU  $BE56      ;gefundene Parameter
                   12  *
                   13  TXBUF    EQU  $BCBD      ;Puffer des B.S.
                   14  GOSYSTEM EQU  $BE70      ;MLI-Aufrufe
                   15  M_ROOM   EQU  $BEF5      ;Aufruf von MAKE ROOM
                   16  D_ROOM   EQU  $BEF8      ; "" von DISCARD ROOM
 Zur Adresse       17  M_TOP    EQU  $BEFB      ;Konstante für D_ROOM
 jeweis $2000      18  *
 hinzuzählen!      19          ORG  $2000
0000: 20 29 24 20  20  INSTL    JSR  INSTALL    ;RELOCATE+COPY
                   21  *
0003: AD 08 BE     22          LDA  EXTRNCMD+2 ;alte Adresse von
0006: AC 07 BE     23          LDY  EXTRNCMD+1 ;EXTRNCMD retten
0009: 8D 4F 20     24          STA  NOTMINE+3  ;diese Befehle sind
000C: 8C 4E 20     25          STY  NOTMINE+2  ;bereits relokalisiert!
000F: AD 1E 20     26          LDA  JMPSCAN+2  ;relokalisierter Sprung
0012: AC 1D 20     27          LDY  JMPSCAN+1  ;zu SCANCMD
0015: 8D 08 BE     28          STA  EXTRNCMD+2 ;wird in EXTRNCMD
0018: 8C 07 BE     29          STY  EXTRNCMD+1 ;gesetzt
001B: 60           30          RTS
001C: 4C 1F 20     31  JMPSCAN JMP  SCANCMD    ;nur für RELOC
                   32  *
001F: A0 01        33  SCANCMD LDY  #01        ;Anzahl der Kommandos-1
0021: 8C AF 23     34          STY  INTCNUM
0024: A0 0C        35          LDY  #LCMD-CMDS ;Ende Kommandotabelle
0026: A9 00        36  NXTCMD  LDA  #0
0028: 8D B0 23     37          STA  CMDOK      ;Flag: Kommando erkannt
002B: 88           38          DEY
002C: B9 D3 23     39          LDA  CMDS,Y     ;Länge momentanes CMD-1
002F: 8D 52 BE     40          STA  XLEN
0032: AA           41          TAX             ;Zähler für Vergleich
0033: 88           42  DOSCAN  DEY
0034: BD BD BC     43          LDA  TXBUF,X    ;Vergleich TXBUF
0037: D9 D3 23     44          CMP  CMDS,Y     ;mit CMDS
003A: F0 03        45          BEQ  CHAROK
003C: EE B0 23     46          INC  CMDOK      ;ungleich: Flag setzen
003F: CA           47  CHAROK  DEX
0040: 10 F1        48          BPL  DOSCAN
0042: AD B0 23     49          LDA  CMDOK      ;Vergleich ok?
0045: F0 09        50          BEQ  GOTCMD     ;-> ja
0047: CE AF 23     51          DEC  INTCNUM    ;CMDS zuende?
004A: 10 DA        52          BPL  NXTCMD     ;-> nein, nächstes CMD
                   53  *
004C: 38           54  NOTMINE SEC
004D: 4C 00 00     55          JMP  0000       ;Exit bzw. nächste Erw.
                   56  *
0050: EE 53 BE     57  GOTCMD  INC  XCNUM      ;auf $00
0053: AD AF 23     58          LDA  INTCNUM    ;"interne" Kommando-Nr
0056: 0E AF 23     59          ASL  INTCNUM
0059: AC AF 23     60          LDY  INTCNUM    ;* 2 für PBITS
005C: 6D AF 23     61          ADC  INTCNUM    ;mal 3
005F: AA           62          TAX             ;=> Index für
0060: BD 7B 20     63          LDA  CMDJMPS+1,X ;Startadresse
0063: 8D 50 BE     64          STA  XTRNADDR   ;und Einsetzen in
0066: BD 7C 20     65          LDA  CMDJMPS+2,X
0069: 8D 51 BE     66          STA  XTRNADDR+1 ;XTRNADDR
006C: B9 CF 23     67          LDA  CBITS,Y    ;Y = INTCMD * 2
006F: 8D 54 BE     68          STA  PBITS      ;Einsetzen in PBITS
0072: B9 D0 23     69          LDA  CBITS+1,Y
0075: 8D 55 BE     70          STA  PBITS+1    ;CBITS in DATA-Sektion!
0078: 18           71          CLC
0079: 60           72          RTS             ;zum BASIC.SYSTEM
                   73  *
                   74  CMDJMPS EQU  *
007A: 4C 80 20     75          JMP  QUIT       ;ist relokalisiert!
007D: 4C C9 20     76          JMP  FCOPY
                   77  *
                   78  *.. weitere Kommandos
                   79  *
                   80  QUIT    EQU  *
0080: AD 08 BE     81          LDA  EXTRNCMD+2 ;zeigt XTRNCMD direkt
0083: CD 1E 20     82          CMP  JMPSCAN+2  ;auf SCANCMD?
0086: F0 11        83          BEQ  DOQUIT     ;-> ja
0088: A2 00        84          LDX  #$00       ;dieses Kommando ist
008A: BD B1 23     85  CANTQT  LDA  NMSG,X     ;nicht das "unterste"
008D: F0 08        86          BEQ  CNZ        ;im Speicher und kann
008F: 09 80        87          ORA  #$80       ;nicht entfernt werden
0091: 20 ED FD     88          JSR  COUT       ;'REMOVE OTHER
0094: E8           89          INX             ;"XTRNS" FIRST'
0095: D0 F3        90          BNE  CANTQT     ;"always"
0097: 18           91  CNZ     CLC
0098: 60           92          RTS             ;zum BASIC.SYSTEM
                   93  *
0099: AD 4F 20     94  DOQUIT  LDA  NOTMINE+3  ;EXTRNCMD wieder
009C: AC 4E 20     95          LDY  NOTMINE+2  ;auf den alten Stand
009F: 8D 08 BE     96          STA  EXTRNCMD+2
00A2: 8C 07 BE     97          STY  EXTRNCMD+1
00A5: A2 15        98          LDX  #QTEND-QUITZ ;Anzahl Bytes
00A7: BD B3 20     99  MVQUIT  LDA  QUITZ,X    ;dieser Teil muß
00AA: 9D 80 02    100          STA  $0280,X    ;kopiert werden, weil
00AD: CA          101          DEX             ;er sonst von DISCARD
00AE: 10 F7       102          BPL  MVQUIT     ;übermangelt wird!
00B0: 4C 80 02    103          JMP  $0280
00B3: AD FB BE    104  QUITZ   LDA  M_TOP      ;Konstante f. D_ROOM
00B6: 48          105          PHA             ;zwischenspeichern
00B7: AD AE 23    106          LDA  PGSTART    ;Programmstart
00BA: 18          107          CLC             ;plus Prog-Länge
00BB: 69 01       108          ADC  #>PRGLEN-$400 ;ergibt
00BD: 8D FB BE    109          STA  M_TOP      ;Obergrenze für D_ROOM
00C0: 20 F8 BE    110          JSR  D_ROOM     ;DISCARD ROOM
00C3: 68          111          PLA
00C4: 8D FB BE    112          STA  M_TOP      ;Restore M_TOP
00C7: 18          113          CLC             ;Kommando o.k.
00C8: 60          114  QTEND   RTS             ;zum BASIC.SYSTEM
                  115  *
                  117  ****************************
                  118  FCOPY    EQU  *
                  119  ****************************
                  121  *
                  122  *Applesoft-Speicherstellen
                  123  STREND   EQU  $6D        ;Ende VAR-Tabellen
                  124  FRETOP   EQU  $6F        ;unteres Ende Strings
```

```
                        125 HIMEM    EQU  $73        ;= ProDOS-Pufferstart
                        126 *
                        127 INBUF    EQU  $0200      ;Kommandozeile
                        128 *
                        129 *Global Page des BASIC.SYSTEM
                        130 OPENCNT  EQU  $BE4D      ;Zahl offener Dateien
                        131 CRACCESS EQU  $BEA3      ;ACCESS f. CREATE
                        132 CRFILID  EQU  $BEA4      ;Dateityp
                        133 CRFKIND  EQU  $BEA7      ;Storage Type Kopie
                        134 *
                        135 SSGINFO  EQU  $BEB4      ;Parm-Zahl F'INFO
                        136 FIFILID  EQU  $BEB8      ;Dateityp Original
                        137 FIFKIND  EQU  $BEBB      ;Storage Type Org.
                        138 *
                        139 SUNITNUM EQU  $BEC7      ;Unitno f. ONLINE
                        140 SREFNUM  EQU  $BEC7      ;Refno MARK/EOF
                        141 SMARK    EQU  $BEC8      ;Daten GET/SET MARK
                        142 SEOF     EQU  $BEC8      ;Daten GET EOF
                        143 *
                        144 OSYSBUF  EQU  $BECE      ;ProDOS-Puffer
                        145 OREFNUM  EQU  $BED0      ;Refno offene Datei
                        146 RWREFNUM EQU  $BED6      ;Refno READ/WRITE
                        147 RWDATA   EQU  $BED7      ;Datenadresse RD/WR
                        148 RWCOUNT  EQU  $BED9      ;Anzahl Bytes R/W
                        149 RWTRANS  EQU  $BEDB      ;tatsächliche Anzahl
                        150 CFREFNUM EQU  $BEDE      ;Refno CLOSE
                        151 *
                        152 *ProDOS
                        153 LASTDEV  EQU  $BF30      ;aktuelle Unitnummer
                        154 *
                        155 *Monitor
                        156 RDKEY    EQU  $FD0C      ;Tastatur lesen
                        157 *
0C9: 20 72 21           158          JSR  GETPATHS   ;Pfadnamen bestimmen
0CC: B0 4B              159          BCS  ERRJMP     ;-> SYNTAX
0CE: A9 15              160          LDA  #21
0D0: AE 4D BE           161          LDX  OPENCNT    ;Dateien offen?
0D3: D0 44              162          BNE  ERRJMP     ;-> ja: FILES OPEN
0D5: 20 33 22           163          JSR  SETVARS    ;Puf-Größe, Positionen
0D8: B0 3F              164          BCS  ERRJMP     ;-> NO BUFFERS
0DA: 20 F0 21           165          JSR  CMPVOL     ;VOL.A = VOL.B?
                        166 *
0DD: 20 14 22           167          JSR  SPATH_A    ;Pfadname A setzen
0E0: A9 0A              168          LDA  #$0A
0E2: 8D B4 BE           169          STA  SSGINFO    ;Parm-Zahl GET F'INFO
0E5: A9 C4              170          LDA  #$C4
0E7: 20 70 BE           171          JSR  GOSYSTEM   ;GET F'INFO für
0EA: B0 2D              172          BCS  ERRJMP     ;die Quell-Datei
0EC: A9 0D              173          LDA  #13
0EE: AE BB BE           174          LDX  FIFKIND    ;Storage Type:
0F1: E0 04              175          CPX  #$04       ;ist Directory?
0F3: B0 24              176          BCS  ERRJMP     ;ja: FILE TYPE MISMATCH
0F5: AD 30 BF           177          LDA  LASTDEV    ;Unitnummer des
0F8: 8D F1 23           178          STA  UNIT_A     ;Quell-Laufwerks
0FB: D0 10              179          BNE  READ_A     ;ok. 1.Teil lesen
                        180 *
0FD: AD F1 23           181 LOOP     LDA  UNIT_A     ;ONLINE, falls UNITs
100: 20 6C 22           182          JSR  ONLINE     ;gleich, VOLs versch.
103: 90 08              183          BCC  READ_A     ;-> Volume gefunden
105: 20 4B 23           184 NO_A     JSR  MSG1       ;'/xxx/ EINLEGEN'...
108: B0 F3              185          BCS  LOOP       ;-> nochmal versuchen
10A: 4C 71 21           186          JMP  EXIT       ;Abbruch
                        187 *
10D: 20 14 22           188 READ_A   JSR  SPATH_A    ;Pfadname A setzen
110: 20 A6 22           189          JSR  READ       ;OPEN/READ/CLOSE Quelle
113: 90 07              190          BCC  GET_B
115: C9 06              191          CMP  #$06       ;FILE NOT FOUND?
117: F0 EC              192          BEQ  NO_A       ;-> ja, Meldung
119: 4C 70 21           193 ERRJMP   JMP  ERROR
                        194 *
11C: AD ED 23           195 GET_B    LDA  PASSONE    ;erster Durchlauf?
11F: D0 1D              196          BNE  TEST_B     ;-> nein
121: 20 19 22           197          JSR  SPATH_B    ;Durchlauf 1:
124: 20 57 22           198 CR_B     JSR  CREATE     ;CREATE von Datei B
127: AE 30 BF           199          LDX  LASTDEV    ;Unitnummer von
12A: 8E F2 23           200          STX  UNIT_B     ;Datei B
12D: 8E ED 23           201          STX  PASSONE    ;immer <> 0
130: 90 1C              202          BCC  WRITE_B
132: C9 06              203          CMP  #$06       ;FILE (DIR) NOT FOUND?
134: D0 3A              204          BNE  ERROR
136: 20 50 23           205          JSR  MSG2       ;'/yyy/ EINLEGEN'...
139: B0 E9              206          BCS  CR_B       ;-> nochmal
13B: 4C 71 21           207          JMP  EXIT
                        208 *
13E: AD F2 23           209 TEST_B   LDA  UNIT_B     ;ONLINE, wenn 1 Drive
141: 20 6C 22           210          JSR  ONLINE     ;und VOL_A <> VOL_B
144: 90 08              211          BCC  WRITE_B
```

```
146: 20 50 23           212 NO_B     JSR  MSG2       ;'/yyy/ EINLEGEN'...
149: B0 F3              213          BCS  TEST_B
14B: 4C 71 21           214          JMP  EXIT
                        215 *
14E: 20 19 22           216 WRITE_B  JSR  SPATH_B    ;Pfadname B setzen
151: 20 05 23           217          JSR  WRITE      ;Pufferinhalt schreiben
154: 90 06              218          BCC  TSTEND     ;-> ok. Fertig?
156: C9 06              219          CMP  #$06       ;FILE NOT FOUND?
158: D0 16              220          BNE  ERROR
15A: F0 EA              221          BEQ  NO_B       ;ja, Meldung
                        222 *
15C: AD EC 23           223 TSTEND   LDA  DONE       ;A komplett gelesen?
15F: F0 03              224          BEQ  SET_B      ;-> ja
161: 4C FD 20           225          JMP  LOOP       ;nächster Durchgang
                        226 *
164: A9 07              227 SET_B    LDA  #$07       ;nach Ende der
166: 8D B4 BE           228          STA  SSGINFO    ;Kopie wird das
169: A9 C3              229          LDA  #$C3       ;FILE INFO für die
16B: 20 70 BE           230          JSR  GOSYSTEM   ;neue Datei gesetzt
16E: 90 01              231          BCC  EXIT
                        232 *
170: 38                 233 ERROR    SEC
171: 60                 234 EXIT     RTS
                        235 *
                        236 ********************************
                        237 *
                        238 GETPATHS EQU  *
                        239 *
172: A2 00              240          LDX  #0         ;Suche nach dem
174: 20 DC 21           241 SKIPY    JSR  NXTCHR     ;1.Zeichen
177: C9 59              242          CMP  #'Y'       ;nach "COPY"
179: D0 F9              243          BNE  SKIPY
17B: 20 B7 21           244 G_PATHA  JSR  SETPATH    ;Pfadname A:
17E: F0 33              245          BEQ  SYNTAX     ;-> Ende mit <CR>
180: 48                 246          PHA             ;"Stop-Zeichen"
181: 98                 247          TYA             ;Länge Pfad A
182: AC F3 23           248          LDY  TEMP       ;Start-Index A
185: 8C EF 23           249          STY  INDEXA
188: 99 FF 01           250          STA  INBUF-1,Y  ;Längenbyte davor
18B: 68                 251          PLA
18C: C9 2C              252          CMP  #','       ;Komma?
18E: F0 07              253          BEQ  G_PATHB    ;-> ja
190: 20 D1 21           254          JSR  SKIPSP     ;nein, Leerzeichen
193: C9 2C              255          CMP  #','       ;davor überspringen
195: D0 1C              256          BNE  SYNTAX
197: E8                 257 G_PATHB  INX             ;Komma überspringen
198: 20 B7 21           258          JSR  SETPATH    ;Pfad B:
19B: 48                 259          PHA             ;darf mit <CR> enden
19C: 98                 260          TYA             ;Länge Pfad B
19D: AC F3 23           261          LDY  TEMP       ;Start-Index
1A0: 8C F0 23           262          STY  INDEXB
1A3: 99 FF 01           263          STA  INBUF-1,Y  ;Längenbyte davor
1A6: 68                 264          PLA
1A7: C9 0D              265          CMP  #$0D       ;Ende mit <CR>?
1A9: F0 06              266          BEQ  GOTPATHS   ;-> ja. Ansonsten
1AB: CA                 267          DEX             ;neuer Test und Leer-
1AC: 20 D1 21           268          JSR  SKIPSP     ;zeichen davor
1AF: D0 02              269          BNE  SYNTAX     ;überspringen
1B1: 18                 270 GOTPATHS CLC
1B2: 60                 271          RTS
                        272 *
1B3: A9 10              273 SYNTAX   LDA  #16        ;SYNTAX ERROR
1B5: 38                 274          SEC
1B6: 60                 275          RTS
                        276 *
1B7: 20 D1 21           278 SETPATH  JSR  SKIPSP     ;X auf erstes
1BA: F0 14              279          BEQ  PATHEND    ;nicht-Leerzeichen
1BC: 8E F3 23           280          STX  TEMP       ;<- Start-Index
1BF: A0 FF              281          LDY  #$FF
1C1: C8                 282 COUNT    INY             ;Zähler: Pfadlänge
1C2: 20 DC 21           283          JSR  NXTCHR
1C5: F0 09              284          BEQ  PATHEND    ;-> <CR>
1C7: C9 2C              285          CMP  #','       ;Komma?
1C9: F0 04              286          BEQ  PATHSET    ;Pfad-Ende
1CB: C9 20              287          CMP  #' '       ;Leerzeichen?
1CD: D0 F2              288          BNE  COUNT
1CF: CA                 289 PATHSET  DEX             ;auf "Stop-Zeichen"
1D0: 60                 290 PATHEND  RTS
                        291 *
1D1: 20 DC 21           292 SKIPSP   JSR  NXTCHR     ;liefert erstes
1D4: F0 05              293          BEQ  SPSKIP     ;nicht-Leerzeichen
1D6: C9 20              294          CMP  #' '       ;und "EQ", wenn <CR>
1D8: F0 F7              295          BEQ  SKIPSP
1DA: CA                 296          DEX
1DB: 60                 297 SPSKIP   RTS
                        298 *
1DC: BD 00 02           299 NXTCHR   LDA  INBUF,X    ;X: Zeiger auf
```

```
1DF: 29 7F    300          AND  #$7F       ;das momentane
1E1: C9 61    301          CMP  #'a'       ;Zeichen. Wird nur
1E3: 90 02    302          BCC  ISUPPER    ;erhöht, wenn
1E5: 29 DF    303          AND  #$DF       ;Zeichen <> <CR>
1E7: 9D 00 02 304 ISUPPER  STA  INBUF,X    ;GROßBUCHSTABEN
1EA: C9 0D    305          CMP  #$0D       ;ist <CR>?
1EC: F0 01    306          BEQ  NXTRET     ;ja, "EQ" zurück
1EE: E8       307          INX             ;"NE"
1EF: 60       308 NXTRET   RTS
              309 *
1F0: AE EF 23 310 CMPVOL   LDX  INDEXA     ;Start Pfad A
1F3: AC F0 23 311          LDY  INDEXB     ;Start Pfad B
1F6: BD 00 02 312          LDA  INBUF,X    ;beginnen beide
1F9: D9 00 02 313          CMP  INBUF,Y    ;Pfade mit "/"?
1FC: D0 15    314          BNE  NOTSVL
1FE: C9 2F    315          CMP  #'/'       ;via PREFIX?
200: D0 0E    316          BNE  ISSVL      ;ja, -> VOLs gleich
202: E8       317 CMPV1    INX             ;Vergleich der
203: C8       318          INY             ;beiden VOL-Namen
204: BD 00 02 319          LDA  INBUF,X    ;wenn gleich, dann
207: D9 00 02 320          CMP  INBUF,Y    ;SAMEVL auf $FF,
20A: D0 07    321          BNE  NOTSVL     ;sonst bleibts 0
20C: C9 2F    322          CMP  #'/'       ;zweites "/"?
20E: D0 F2    323          BNE  CMPV1
210: CE EE 23 324 ISSVL    DEC  SAMEVL     ;ok. Setzen
213: 60       325 NOTSVL   RTS
              326 *
214: AE EF 23 328 SPATH_A  LDX  INDEXA     ;Pfadname A setzen
217: D0 03    329          BNE  SPATH
219: AE F0 23 330 SPATH_B  LDX  INDEXB     ;Pfadname B setzen
              331 *
21C: CA       332 SPATH    DEX             ;-> Längebyte
21D: BD 00 02 333          LDA  INBUF,X    ;Pfad-Länge
220: 8D F3 23 334          STA  TEMP
223: A0 00    335          LDY  #0
225: BD 00 02 336 SPH1     LDA  INBUF,X    ;Kopie des Pfades
228: 99 BC BC 337          STA  TXBUF-1,Y  ;in den Puffer
22B: E8       338          INX             ;des BASIC.SYSTEM
22C: C8       339          INY
22D: CE F3 23 340          DEC  TEMP
230: 10 F3    341          BPL  SPH1
232: 60       342          RTS
              343 *
233: A2 0C    345 SETVARS  LDX  #$0C       ;POS_A, POS_B
235: A9 00    346          LDA  #0         ;Pufferlänge etc.
237: 9D E2 23 347 SV1      STA  POS_A,X    ;auf 00
23A: CA       348          DEX
23B: 10 FA    349          BPL  SV1
              350 *
23D: A6 6E    351          LDX  STREND+1   ;Ende der VAR-
23F: E8       352          INX             ;Tabelle von Applesoft
240: 8E E9 23 353          STX  BUFSTART+1
243: A5 70    354          LDA  FRETOP+1
245: 18       355          CLC
246: E5 6E    356          SBC  STREND+1   ;Pufferlänge
248: 8D EB 23 357          STA  BUFLEN+1
24B: 90 06    358          BCC  NOSPACE
24D: C9 02    359          CMP  #$02       ;mindestens $200 Byte?
24F: 90 02    360          BCC  NOSPACE
251: 18       361          CLC
252: 60       362          RTS
253: A9 0C    363 NOSPACE  LDA  #12         ;NO BUFFERS AVALABLE
255: 38       364          SEC
256: 60       365          RTS
              366 *
              367 ************************
              368 *
257: A9 C3    369 CREATE   LDA  #$C3       ;ACCESS für die
259: 8D A3 BE 370          STA  CRACCESS   ;neue Datei: RW erlaubt
25C: AD B8 BE 371          LDA  FIFILID    ;Dateityp von A
25F: 8D A4 BE 372          STA  CRFILID    ;-> Typ neue Datei
262: A9 01    373          LDA  #$01       ;Storage Type:
264: 8D A7 BE 374          STA  CRFKIND    ;= $01
267: A9 C0    375          LDA  #$C0
269: 4C 70 BE 376          JMP  GOSYSTEM   ;CREATE
              377 *
26C: 8D C7 BE 378 ONLINE   STA  SUNITNUM   ;Unitnummer
26F: AD EE 23 379          LDA  SAMEVL     ;VOL_A = VOL_B?
272: 30 19    380          BMI  ONQUIT     ;-> ja, kein ONLINE
274: AD F1 23 381          LDA  UNIT_A     ;verschiedene UNITs
277: CD F2 23 382          CMP  UNIT_B     ;für A und B?
27A: D0 11    383          BNE  ONQUIT     ;-> ja, kein ONLINE
27C: A9 02    384          LDA  #$02       ;Puffer für
27E: 8D C9 BE 385          STA  SMARK+1    ;ONLINE: $2A0
281: A9 A0    386          LDA  #$A0
283: 8D C8 BE 387          STA  SMARK
286: A9 C5    388          LDA  #$C5       ;verschiedene VOLs in
```

```
288: 20 70 BE 389          JSR  GOSYSTEM   ;einem Drive
28B: B0 01    390          BCS  BADONL
28D: 18       391 ONQUIT   CLC
28E: 60       392 BADONL   RTS
              393 *
28F: A5 74    395 OPEN     LDA  HIMEM+1    ;ProDOS-Puffer ist der
291: 8D CF BE 396          STA  OSYSBUF+1  ;"interne" des B-S
294: A9 C8    397          LDA  #$C8       ;OPEN der Datei
296: 20 70 BE 398          JSR  GOSYSTEM
299: AE D0 BE 399          LDX  OREFNUM    ;zurückgelieferte
29C: 8E D6 BE 400          STX  RWREFNUM   ;Refno -> READ/WRITE
29F: 8E DE BE 401          STX  CFREFNUM   ;-> CLOSE
2A2: 8E C7 BE 402          STX  SREFNUM    ;SET/GET MARK,EOF
2A5: 60       403          RTS
              404 *
              406 READ     EQU  *
2A6: A2 03    407          LDX  #$03       ;Setzen von RWDATA
2A8: BD E8 23 408 SBUF     LDA  BUFSTART,X ;und RWCOUNT
2AB: 9D D7 BE 409          STA  RWDATA,X   ;mit Pufferstart
2AE: CA       410          DEX             ;und Pufferlänge
2AF: 10 F7    411          BPL  SBUF
              412 *
2B1: 20 8F 22 413          JSR  OPEN       ;OPEN von Datei A
2B4: B0 4E    414          BCS  BADRD
2B6: AD ED 23 415          LDA  PASSONE    ;erster Durchgang?
2B9: D0 10    416          BNE  GETDTA     ;-> nein
2BB: A9 D1    417          LDA  #$D1       ;Lesen des EOF von A
2BD: 20 70 BE 418          JSR  GOSYSTEM   ;für Vergleich
2C0: A2 02    419          LDX  #$02       ;auf "alles gelesen"
2C2: BD C8 BE 420 GETEOF   LDA  SEOF,X
2C5: 9D DF 23 421          STA  SIZE_A,X
2C8: CA       422          DEX
2C9: 10 F7    423          BPL  GETEOF
              424 *
2CB: A2 02    425 GETDTA   LDX  #$02       ;POS_A -> SMARK
2CD: 20 3E 23 426          JSR  GETPOS
2D0: A9 CE    427          LDA  #$CE       ;SET MARK in
2D2: 20 70 BE 428          JSR  GOSYSTEM   ;Datei A
2D5: B0 27    429          BCS  ERRCLOSE
2D7: A9 CA    430          LDA  #$CA       ;READ: wird vom
2D9: 20 70 BE 431          JSR  GOSYSTEM   ;MLI auf den EOF
2DC: B0 20    432          BCS  ERRCLOSE   ;begrenzt
2DE: A9 CF    433          LDA  #$CF       ;GET MARK: wo sind
2E0: 20 70 BE 434          JSR  GOSYSTEM   ;wir jetzt?
              435 *
2E3: A0 00    436          LDY  #$00       ;Speicherung der
2E5: A2 02    437          LDX  #$02       ;jetzigen Position
2E7: BD C8 BE 438 SPOS_A   LDA  SMARK,X    ;und dabei
2EA: 9D E2 23 439          STA  POS_A,X    ;Vergleich mit
2ED: DD DF 23 440          CMP  SIZE_A,X   ;dem EOF
2F0: F0 01    441          BEQ  SP1
2F2: C8       442          INY             ;<>0: POS<>EOF
2F3: CA       443 SP1      DEX
2F4: 10 F1    444          BPL  SPOS_A
2F6: 8C EC 23 445          STY  DONE       ;00: letzter READ
              446 *
2F9: A9 CC    447 RDCLOSE  LDA  #$CC       ;CLOSE der Datei
2FB: 4C 70 BE 448          JMP  GOSYSTEM
              449 *
2FE: 48       451 ERRCLOSE PHA             ;Fehlernummer retten
2FF: 20 F9 22 452          JSR  RDCLOSE    ;CLOSE
302: 68       453          PLA             ;Fehlernummer
303: 38       454          SEC
304: 60       455 BADRD    RTS
              456 *
              458 WRITE    EQU  *
305: AD DB BE 459          LDA  RWTRANS    ;von READ gelesene
308: 8D D9 BE 460          STA  RWCOUNT    ;Byte-Anzahl ->
30B: AD DC BE 461          LDA  RWTRANS+1  ;zu schreibende
30E: 8D DA BE 462          STA  RWCOUNT+1  ;Bytes
311: 20 8F 22 463          JSR  OPEN       ;OPEN von Datei B
314: B0 EE    464          BCS  BADRD
316: A2 05    465          LDX  #$05       ;POS_B -> SMARK
318: 20 3E 23 466          JSR  GETPOS
31B: A9 CE    467          LDA  #$CE       ;und Position setzen
31D: 20 70 BE 468          JSR  GOSYSTEM
320: B0 DC    469          BCS  ERRCLOSE
322: A9 CB    470          LDA  #$CB       ;WRITE ab dieser
324: 20 70 BE 471          JSR  GOSYSTEM   ;Position
327: B0 D5    472          BCS  ERRCLOSE
329: A9 CF    473          LDA  #$CF       ;GET MARK: wo
32B: 20 70 BE 474          JSR  GOSYSTEM   ;sind wir?
32E: A2 02    475          LDX  #$02
330: BD C8 BE 476 SPOS_B   LDA  SMARK,X    ;Speicherung
333: 9D E5 23 477          STA  POS_B,X    ;der Position
336: CA       478          DEX             ;(entspricht dem EOF)
337: 10 F7    479          BPL  SPOS_B     ;von Datei B
```

```
                        480 *
339: A9 CC              481         LDA  #$CC       ;und CLOSE von B
33B: 4C 7Ø BE           482         JMP  GOSYSTEM
                        483 *
33E: AØ Ø2              485 GETPOS  LDY  #$Ø2       ;Kopie von
34Ø: BD E2 23           486 GPS1    LDA  POS_A,X    ;POS_A/POS_B
343: 99 C8 BE           487         STA  SMARK,Y    ;für SET MARK
346: CA                 488         DEX
347: 88                 489         DEY
348: 1Ø F6              49Ø         BPL  GPS1
34A: 6Ø                 491         RTS
                        492 *
                        493 *************************
                        494 *
34B: AE EF 23           495 MSG1    LDX  INDEXA     ;Pfadname A
34E: DØ Ø3              496         BNE  TSTVOL     ;"always"
35Ø: AE FØ 23           497 MSG2    LDX  INDEXB     ;Pfadname B
                        498 *
353: BD ØØ Ø2           499 TSTVOL  LDA  INBUF,X    ;Start mit "/",
356: C9 2F              5ØØ         CMP  #'/'       ;d.h. voller Pfad?
358: FØ 15              5Ø1         BEQ  PRTVOL     ;-> ja
35A: A9 C7              5Ø2         LDA  #$C7       ;ansonsten wird
35C: 2Ø 7Ø BE           5Ø3         JSR  GOSYSTEM   ;das PREFIX
35F: A2 ØØ              5Ø4         LDX  #Ø         ;geholt und
361: BD BD BC           5Ø5 PRTPFX  LDA  TXBUF,X    ;ausgegeben. Kann
364: 2Ø A9 23           5Ø6         JSR  PRINT      ;an diesem Punkt
367: E8                 5Ø7         INX             ;nie leer sein!
368: CE BC BC           5Ø8         DEC  TXBUF-1    ;Längenbyte
36B: DØ F4              5Ø9         BNE  PRTPFX
36D: FØ ØB              51Ø         BEQ  PRTMSG
                        511 *
36F: 2Ø A9 23           512 PRTVOL  JSR  PRINT      ;zuerst 1. "/"
372: E8                 513         INX             ;danach der VOL-Name
373: BD ØØ Ø2           514         LDA  INBUF,X
376: C9 2F              515         CMP  #'/'       ;2. "/"?
378: DØ F5              516         BNE  PRTVOL
                        517 *
37A: A2 ØØ              518 PRTMSG  LDX  #Ø
37C: BD F4 23           519 PRTM1   LDA  TEXT,X     ;' EINLEGEN UND:'
37F: FØ Ø6              52Ø         BEQ  MSGEND
381: 2Ø A9 23           521         JSR  PRINT      ;<RETURN>=WEITER
384: E8                 522         INX             ;<CONTROL-C>=ENDE
385: DØ F5              523         BNE  PRTM1
                        524 *
387: A2 15              525 MSGEND  LDX  #21        ;Cursor zurück
389: A9 Ø8              526 PRTM2   LDA  #$Ø8
38B: 2Ø A9 23           527         JSR  PRINT      ;21 * <BS>
38E: CA                 528         DEX
38F: DØ F8              529         BNE  PRTM2
                        53Ø *
391: 2Ø ØC FD           531 GETANS  JSR  RDKEY
394: C9 8D              532         CMP  #$8D       ;<RETURN>?
396: FØ Ø4              533         BEQ  GOTANS
398: C9 83              534         CMP  #$83       ;<CONTROL-C>?
39A: DØ F5              535         BNE  GETANS
39C: 48                 536 GOTANS  PHA             ;gelesenes Zeichen
39D: A9 ØD              537         LDA  #$ØD
39F: 2Ø A9 23           538         JSR  PRINT      ;zwei * <CR>
3A2: 2Ø A9 23           539         JSR  PRINT
3A5: 68                 54Ø         PLA
3A6: C9 8D              541         CMP  #$8D       ;RTS mit C gesetzt,
3A8: 6Ø                 542         RTS             ;wenn <CR>
                        543 *
3A9: Ø9 8Ø              544 PRINT   ORA  #$8Ø
3AB: 4C ED FD           545         JMP  COUT
                        546 *
                        548 *****************************
                        549 DATA    EQU  *
                        55Ø *****************************
                        551 *
                        552 PGSTART DS   1          ;Programmstart
                        553 INTCNUM DS   1          ;"interne" Kommando-Nr.
                        554 CMDOK   DS   1          ;Flag für SCANCMD
                        555 OFFSET  EQU  CMDOK      ;Temp für RELOCATE
                        556 *
3B1: 8D 87              557 NMSG    DFB  $8D,$87    ;<CR>, <BELL>
3B3: 52 45 4D           558         ASC  'REMOVE OTHER "XTRNS" FIRST'
3B6: 4F 56 45 2Ø 4F 54 48 45
3BE: 52 2Ø 22 58 54 52 4E 53
3C6: 22 2Ø 46 49 52 53 54
3CD: 8D ØØ              559         DFB  $8D,ØØ     ;<ØØ> = Ende
                        56Ø *
3CF: ØØ ØØ              561 CBITS   DFB  Ø,Ø         ;keine Parms für NOCOPY
3D1: ØØ ØØ              562         DFB  Ø,Ø         ;nichts für COPY
                        563 *
3D3: 4E 4F 43           564 CMDS    ASC  'NOCOPY'   ;MSB gelöscht...
3D6: 4F 5Ø 59
```

```
3D9: Ø5                 565         DFB  $Ø5        ;Länge von "NOCOPY" - 1
3DA: 43 4F 5Ø           566         ASC  'COPY'
3DD: 59
3DE: Ø3                 567         DFB  $Ø3        ;Länge "COPY"-1
                        568 LCMD    EQU  *
                        569 *
3DF: ØØ ØØ ØØ           57Ø SIZE_A  DFB  Ø,Ø,Ø      ;Dateigröße Original
3E2: ØØ ØØ ØØ           571 POS_A   DFB  Ø,Ø,Ø      ;Position in A
3E5: ØØ ØØ ØØ           572 POS_B   DFB  Ø,Ø,Ø      ;dito B
3E8: ØØ ØØ              573 BUFSTART DA  Ø           ;Pufferstart
3EA: ØØ ØØ              574 BUFLEN  DA   Ø           ;Pufferlänge
                        575 *
3EC: ØØ                 576 DONE    DFB  Ø          ;Flag: letztes READ
3ED: ØØ                 577 PASSONE DFB  Ø          ;Durchgangszähler
3EE: ØØ                 578 SAMEVL  DFB  Ø          ;$FF, wenn VOL_A=VOL_B
                        579 *
3EF: ØØ                 58Ø INDEXA  DFB  Ø          ;Pfadname A: Start
3FØ: ØØ                 581 INDEXB  DFB  Ø          ;dto. Pfadname B
3F1: ØØ                 582 UNIT_A  DFB  Ø          ;Slot/Drive für A
3F2: ØØ                 583 UNIT_B  DFB  Ø          ;Slot/Drive für B
3F3: ØØ                 584 TEMP    DFB  Ø          ;"Scratch"
                        585 *
3F4: 2Ø 45 49           586 TEXT    ASC  ' EINLEGEN UND'
3F7: 4E 4C 45 47 45 4E 2Ø 55
3FF: 4E 44
4Ø1: ØD ØD              587         DFB  $ØD,$ØD
4Ø3: 3C 52 45           588         ASC  '<RETURN>=WEITER'
4Ø6: 54 55 52 4E 3E 3D 57 45
4ØE: 49 54 45 52
412: 2Ø 2Ø 2Ø           589         ASC  '   '
415: 3C 43 4F           59Ø         ASC  '<CONTROL-C>=ABBRUCH'
418: 4E 54 52 4F 4C 2D 43 3E
42Ø: 3D 41 42 42 52 55 43 48
428: ØØ                 591         DFB  Ø          ;Textende
                        592 *
                        593 ************************************
                        594 PRGEND  EQU  *
                        595 *
                        596 PRGLEN  EQU  PRGEND-INSTL+$1ØØ ;Programmlänge
                        597 ************************************
                        598 *
429: A9 Ø5              599 INSTALL LDA  #>PRGLEN   ;Anzahl Prog-Seiten
42B: 2Ø F5 BE           6ØØ         JSR  M_ROOM
42E: 9Ø Ø3              6Ø1         BCC  GOTROOM    ;-> ok. Platz vorhanden
43Ø: 4C Ø9 BE           6Ø2         JMP  ERROUT     ;NO BUFFERS AVAILABLE
433: 8D AE 23           6Ø3 GOTROOM STA  PGSTART    ;Start freier Bereich
436: 38                 6Ø4         SEC
437: E9 2Ø              6Ø5         SBC  #>INSTL    ;minus Programmstart
439: 8D BØ 23           6Ø6         STA  OFFSET     ;= Verschiebungsfaktor
43C: 2Ø 42 24           6Ø7         JSR  RELOCATE
43F: 4C 78 24           6Ø8         JMP  COPY       ;RTS zum Original!
                        6Ø9 *
                        61Ø * Die folgenden Labels müssen definiert sein:
                        611 * INSTL = Programmstart
                        612 * DATA = Anfang Daten, Obergrenze für RELOC
                        613 * PRGEND = Ende Programm, Obergrenze für COPY
                        614 * PRGLEN = Programmlänge in Speicherseiten
                        615 * OFFSET = Speicherstelle mit Verschiebungsfak.
                        616 *
                        617 PTR     EQU  $3A      ;für RELOCATE
                        618 PTR2    EQU  $3C      ;für COPY
                        619 *
442: 2Ø 96 24           62Ø RELOCATE JSR SETPTR     ;(PTR) auf INSTL
445: AØ ØØ              621 RELNEXT LDY  #Ø
447: B1 3A              622         LDA  (PTR),Y    ;Opcode
449: 2Ø 9F 24           623         JSR  DISASM     ;liefert die Länge
44C: C9 Ø2              624         CMP  #$Ø2       ;des Opcode zurück
44E: DØ 13              625         BNE  NOTABS     ;-> nicht absolut
45Ø: AØ Ø2              626         LDY  #$Ø2       ;auf 3.Byte
452: B1 3A              627         LDA  (PTR),Y
454: C9 2Ø              628         CMP  #>INSTL    ;unterhalb der Grenze?
456: 9Ø Ø9              629         BCC  NJET       ;-> ja
458: C9 25              63Ø         CMP  #>PRGEND+$1ØØ ;oberhalb?
45A: BØ Ø5              631         BCS  NJET
45C: 6D BØ 23           632         ADC  OFFSET     ;ok. Offset dazu
45F: 91 3A              633         STA  (PTR),Y
461: A9 Ø2              634 NJET    LDA  #$Ø2       ;Reload der Länge
463: 38                 635 NOTABS  SEC             ;(!)
464: 65 3A              636         ADC  PTR
466: 85 3A              637         STA  PTR        ;PTR auf nächsten Opc
468: A8                 638         TAY             ;für Test auf Prog-Ende
469: A5 3B              639         LDA  PTR+1
46B: 69 ØØ              64Ø         ADC  #$ØØ
46D: 85 3B              641         STA  PTR+1
46F: C9 23              642         CMP  #>DATA     ;DATA-Sektion
471: 9Ø D2              643         BCC  RELNEXT    ;erreicht?
473: CØ AE              644         CPY  #<DATA
```

```
475: 90 CE    645          BCC  RELNEXT
477: 60       646          RTS
              647 *
478: 20 96 24 649 COPY     JSR  SETPTR     ;(PTR) auf INSTL
47B: 84 3C    650          STY  PTR2       ;Ziel-Pointer
47D: 18       651          CLC
47E: 6D B0 23 652          ADC  OFFSET
481: 85 3D    653          STA  PTR2+1
483: A0 00    654          LDY  #0
485: A2 05    655          LDX  #>PRGLEN   ;Seitenzähler
487: B1 3A    656 MOV1     LDA  (PTR),Y
489: 91 3C    657          STA  (PTR2),Y   ;Kopie des
48B: C8       658          INY             ;gesamten Programms
48C: D0 F9    659          BNE  MOV1       ;in die neue
48E: E6 3B    660          INC  PTR+1      ;Quelle + $100
490: E6 3D    661          INC  PTR2+1     ;Ziel + $100
492: CA       662          DEX
493: D0 F2    663          BNE  MOV1
495: 60       664          RTS
              665 *
496: A9 20    667 SETPTR   LDA  #>INSTL    ;höherwertiger Teil
498: A0 00    668          LDY  #<INSTL    ;PTR auf den
49A: 85 3B    669          STA  PTR+1      ;Start des zu
49C: 84 3A    670          STY  PTR        ;relokalisierenden
49E: 60       671          RTS             ;Programmcodes
              672 *
49F: AA       673 DISASM   TAX             ;Opcode
4A0: A0 07    674          LDY  #$07       ;7 ungültige Opcdes,
4A2: D9 D8 24 675 TSTOP    CMP  BADOPS-1,Y ;die nicht von den
4A5: F0 16    676          BEQ  BADINS     ;anderen Tests erkannt
4A7: 88       677          DEY             ;werden
4A8: D0 F8    678          BNE  TSTOP
4AA: D9 E0 24 679 TSTSPEC  CMP  SPECS,Y    ;sowie 16 Opcodes
4AD: F0 08    680          BEQ  SPECOP     ;die nicht logisch
4AF: C8       681          INY             ;dezidierbar sind
4B0: C8       682          INY
4B1: C0 20    683          CPY  #AMODES-SPECS ;Tab-Ende?
4B3: 90 F5    684          BCC  TSTSPEC
4B5: B0 0A    685          BCS  NORMOP
4B7: B9 E1 24 686 SPECOP   LDA  SPECS+1,Y  ;Adressierungsart
4BA: 4C CE 24 687          JMP  GETLEN
4BD: A9 00    688 BADINS   LDA  #$00       ;"implizit"
4BF: F0 0D    689          BEQ  GETLEN
4C1: 4A       690 NORMOP   LSR  A          ;Opcode durch 16
4C2: 4A       691          LSR  A
4C3: 4A       692          LSR  A
4C4: 4A       693          LSR  A          ;$Xx => $0X
4C5: 4A       694          LSR  A          ;setzt C bei $0X odd
4C6: 8A       695          TXA             ;Opcode
4C7: 29 0F    696          AND  #$0F       ;$xX => $0X
4C9: 2A       697          ROL  A          ;=> $0X * 2 + Carry
4CA: A8       698          TAY
4CB: B9 00 25 699          LDA  AMODES,Y   ;Adressierungsart
              700 *
4CE: 4A       701 GETLEN   LSR  A          ;Bits 2/3 stehen für
4CF: 4A       702          LSR  A          ;00 = Implizit, 01 = Imm
4D0: 29 03    703          AND  #$03       ;10 = Relativ, 11 = ABS
4D2: C9 02    704          CMP  #02
4D4: 90 02    705          BCC  GOTLEN
4D6: E9 01    706          SBC  #$01       ;Imm, ZP u. Rel: 2 Byte
4D8: 60       707 GOTLEN   RTS
              708 *
4D9: 44 54 D4 709 BADOPS   DFB  $44,$54,$D4,$F4 ;ungültige Ops
4DC: F4
4DD: 5C DC FC 710          DFB  $5C,$DC,$FC
4E0: 00 00 20 711 SPECS    DFB  $00,$00,$20,$0C,$40,$00 ;spez. Ops
4E3: 0C 40 00
4E6: 60 00 80 712          DFB  $60,$00,$80,$08,$A2,$06
4E9: 08 A2 06
4EC: 14 04 96 713          DFB  $14,$04,$96,$64,$89,$06
4EF: 64 89 06
4F2: 1C 0C 6C 714          DFB  $1C,$0C,$6C,$1D,$7C,$AD
4F5: 1D 7C AD
4F8: 9C 0C BE 715          DFB  $9C,$0C,$BE,$6C,$9E,$2C
4FB: 6C 9E 2C
4FE: B6 64    716          DFB  $B6,$64
500: 06 08 A5 717 AMODES   DFB  $06,$08,$A5,$75,$00,$15 ;Adr-
503: 75 00 15
506: 00 00 04 718          DFB  $00,$00,$04,$24,$04,$24 ;arten
509: 24 04 24
50C: 04 24 00 719          DFB  $04,$24,$00,$00,$00,$00
50F: 00 00 00
512: 06 6C 00 720          DFB  $06,$6C,$00,$00,$00,$00
515: 00 00 00
518: 0C 2C 0C 721          DFB  $0C,$2C,$0C,$2C,$0C,$2C
51B: 2C 0C 2C
51E: 00 00    722          DFB  $00,$00
```

ProDOS-Startup mit Komfort

von Marcus Gröber

Das Programm „STARTUP mit Komfort" ist ein ProDOS-Hilfsprogramm, das beim Booten einer Diskette ein Menü mit allen Programmen dieser Diskette auflistet, von denen jedes mit wenigen Tastendrücken gestartet werden kann.
Das Programm sollte auf einer Diskette unter dem Namen STARTUP gespeichert werden, weil es dann nach dem Laden von ProDOS selbständig aufgerufen wird.

Arbeitsweise des Programms

Nach dem Bootvorgang liest STARTUP den Namen der Diskette ein und zeigt ihn an. Danach wird das Inhaltsverzeichnis von der Diskette gelesen. Dazu wird eine abgewandelte CAT.ARRAY-Routine aus Peeker 4/85 benutzt, um aus dem Catalog ein Datenfeld zu erzeugen. Die in START-UP benutzte Variante von CAT.ARRAY übernimmt jedoch nur startfähige Dateien (mit den Dateierweiterungen BAS, TXT, BIN, SYS) und speichert getrennt davon evtl. vorhandene Subdirectories (Dateierweiterung DIR).
Anschließend zeigt STARTUP die Liste der Programme auf dem Bildschirm an. Sollten es mehr als 15 Einträge sein, so werden sie auf mehrere Seiten verteilt. Am rechten Bildschirmrand werden die Typbezeichnungen der Programme angezeigt. In der untersten Zeile erscheinen Informationen über den Umfang der Programmliste und die Zahl der freien Blocks auf der Diskette.
Wenn die Eingabeforderung erscheint, gibt es verschiedene Möglichkeiten: mit der Taste „Pfeil nach rechts" kann man eine neue Bildschirmseite mit 15 weiteren Programmen abrufen (sofern diese vorhanden sind), mit „Pfeil nach links" wird eine Seite zurückgeblättert.

Programmablauf

Soll ein Programm gestartet werden, so ist zunächst dessen Nummer (wird in der Liste angezeigt) und danach die Taste „S" einzugeben. Zum Laden eines Programmes wird die Nummer und der Buchstabe „L" gedrückt.
Auf die gleiche Weise lassen sich auch Text- und Sys-Files laden und starten. Sys-Files stehen im Speicher ab $2000.

Am Ende der Programmliste werden die Subdirectories geführt. Um ihren Inhalt zu lesen, muß man zuerst ihre Nummer und dann „P" eingeben. Danach startet man die Programme der Subdirectories wie aus dem Hauptinhaltsverzeichnis. Um dorthin zurückzukehren und ein Leerpräfix zu setzen, reicht es, „P" ohne Nummer oder „O P" zu tippen. Damit kann auch der Inhalt einer anderen Diskette gelesen werden.

Durch Drücken der Esc-Taste kann das Programm abgebrochen werden.
Bei der Übernahme des Programmes STARTUP können alle REM-Zeilen weggelassen werden. Das Programm belegt dann auf der Diskette einen Block weniger.

Kurzhinweise

1. Zweck:
ProDOS-Hilfsprogramm; erstellt ein Menü zum Auflisten, Laden und Starten aller Dateien einer Diskette.
2. Konfiguration:
II+/e/c; 40 Z/Z ProDOS;
3. Test:
Programm auf ProDOS-Diskette unter „STARTUP" speichern, Diskette booten.
4. Sammeldisk:
STARTUP

```
STARTUP

100 PRINT CHR$(21): TEXT : HOME : DIM FI$(99),SD$(10)
110 PRINT CHR$ (4)"CLOSE"
120 FI = 0:SD = 0: PRINT CHR$ (4)"PREFIX": INPUT P$
130 REM —— Kopfzeile lesen und ausgeben ——
140 PRINT CHR$ (4)"OPEN"P$",TDIR"
150 PRINT CHR$ (4)"READ"P$
160 INPUT D1$: INPUT D2$: INPUT D3$
170 DI$ = "Diskname:" + MID$ (D1$,2): IF MID$ (D1$,1,1)
    < > "/" THEN DI$ = "Subdirectory:" + D1$
180 VTAB 1: PRINT DI$ TAB( 35)"ProDOS":: FOR A = 1 TO 40:
    PRINT "-";: NEXT A: POKE 34,2: POKE 35,23
190 REM —— Inhaltsverzeichnis lesen ——
200 FI = FI + 1: IF FI > 89 THEN 240
210 INPUT FI$(FI):F$ = MID$ (FI$(FI),18,3): IF F$ = "DIR" AND
    SD < 10 THEN SD = SD + 1:SD$(SD) = FI$(FI)
220 IF F$ < > "SYS" AND F$ < > "BIN" AND F$ < > "BAS" AND
    F$ < > "TXT" AND F$ < > "" THEN FI = FI - 1
230 IF F$ > "" THEN 200
240 FI = FI - 1: INPUT D3$: PRINT CHR$ (4)"CLOSE "P$
250 FOR A = FI + 1 TO FI + SD:FI$(A) = SD$(A - FI): NEXT A:
    F2 = FI + SD
260 FR = VAL ( MID$ (D3$,15,3)):PA = 1
270 REM —— Files als Menü ausgeben ——
280 HTAB 1: VTAB 24: PRINT "Frei:"FR; TAB( 14);"Files:"F2;
    TAB( 26);"Seite:"PA" von " INT ((F2 - 1) / 15) + 1;: HTAB 1
290 VTAB 4: FOR A = (PA - 1) * 15 + 1 TO PA * 15: IF A < = F2
    THEN PRINT " "; RIGHT$ (" " + STR$ (A),2);". ";
    MID$ (FI$(A),2,16) TAB( 34); MID$ (FI$(A),18,3)
300 IF A > F2 THEN PRINT SPC( 40);
310 NEXT A: VTAB 20:NU$ = ""
320 PRINT "Nummer+P = Präfix";TAB( 20);"P     = Nullpräfix":
    PRINT "Nummer+S = Start"; TAB( 20);"<-,-> = vor/zurück";
    PRINT "Nummer+L = Laden"; TAB( 20);"ESC   = Ende  ->   ";:
    HTAB 36
330 GET A$: IF A$ = CHR$ (8) AND PA > 1 THEN PA = PA - 1: GOTO 280
340 REM —— Eingabe annehmen und auswerten ——
350 IF A$ > = "A" AND A$ < = "Z" THEN PRINT A$;
360 IF A$ = CHR$ (21) AND PA * 15 < F2 THEN PA = PA + 1: GOTO 280
370 IF A$ = CHR$ (27) THEN TEXT : HOME : END
380 IF A$ > = "0" AND A$ < = "9" THEN NU$ = NU$ + A$:
    IF LEN (NU$) < 3 THEN PRINT A$;: GOTO 330
390 N = VAL (NU$): IF N > F2 THEN 440
400 IF A$ = "P" AND (N > FI OR N = 0) THEN 570
410 IF N > FI OR N = 0 THEN 440
420 IF A$ = "S" THEN 460
430 IF A$ = "L" THEN 490
440 NU$ = "": HTAB 36: PRINT "   ";: HTAB 36: GOTO 330
450 REM —— Programm starten ——
460 TEXT : HOME : PRINT CHR$ (4)"-" MID$ (FI$(N),2,16)
470 END
480 REM —— Programm lesen (auch SYS!) ——
490 A$ = "":F$ = MID$ (FI$(N),19,1): IF F$ = "A" THEN F$ = "LOAD"
500 IF F$ = "X" THEN F$ = "EXEC"
510 IF F$ = "I" THEN F$ = "BLOAD"
520 IF F$ = "Y" THEN F$ = "BLOAD":A$ = ",A$2000,TSYS"
530 TEXT : HOME
540 PRINT CHR$ (4)F$; MID$ (FI$(N),2,16);A$
550 END
560 REM —— Präfix neu setzen / Leerpräfix setzen ——
570 IF N = 0 THEN PRINT CHR$ (4)"PREFIX /": GOTO 120
580 PRINT CHR$ (4)"PREFIX " MID$ (FI$(N),2,16): GOTO 120
```

Druckertreiber für Fast-Writer

und andere Textverarbeitungsprogramme

von Ulrich Stiehl

Das Textprogramm Fast-Writer gestattet die direkte Eingabe aller ASCII-Zeichen von $01 bis $7F (1-127). Damit können auch alle Ctrl-Zeichen mit Ausnahme von Ctrl-0 ($00 = 0) eingegeben werden. Wenn man jedoch die Fähigkeiten eines Matrixdruckers voll ausnutzen will – und das wollen viele! –, so müssen oft schier endlose Esc-Sequenzen zur Druckersteuerung eingetippt werden. Hatte der Epson MX-80 aus dem Jahre 1981 noch eine 50seitige Anleitung mit nur 29 Steuerzeichen einschließlich Formfeed, Linefeed, Return usw., so beschreibt die 200seitige Anleitung zum Epson LQ-800 aus dem Jahre 1985 eine fast unübersehbare Fülle von Esc-Sequenzen. Will man beispielsweise ein schwarzes Viereck (= volles Kästchen als Blockade) ausgeben, so muß man eine 13 Zeichen umfassende Esc-Sequenz eintippen. Leichter wäre es, wenn man statt dessen eine 2-Zeichen-Kombination, z.B. „§b", eingeben könnte, die dann während des Ausdrucks in die gewünschte 13stellige Esc-Sequenz umgewandelt werden würde. Unter einer **Makro-Definition** verstehen wir die Festlegung eines Ausgangscodes (= Ausgangszeichenfolge) für einen bestimmten Eingangscode (= Eingangszeichenfolge), z.B.

```
Eingangscode -> Ausgangscode
§b           -> Esc  t Ctrl-A Esc 7...
§b           -> 1B  74 Ø1     1B  37...
```

Der Fast-Writer gestattet zwar die Definition von Makros sowie das Anlegen ganzer Makro-Dateien, die automatisch abgearbeitet werden können, doch sind diese Fast-Writer-Makros erstens für „normale" Makros, z.B. „BA" für „Sehr geehrte Herren" (Briefanrede usw.), sowie zweitens für die Automatisierung von Befehlsfolgen (Laden-Drucken-Laden usw.) gedacht. Man könnte zwar auch beim Fast-Writer vor dem Ausdruck eines Textes alle darin enthaltenen Eingangscodes durch die entsprechenden Ausgangscodes ersetzen, doch hat dies zwei Nachteile:

– Erstens kann es passieren, daß ein extrem langer Text mit vielen Makros derart expandiert, daß er gar nicht mehr in den Speicher paßt.

– Zweitens werden die in Esc-Sequenzen umgewandelten Eingangscodes bei der Berechnung des Zeilenumbruchs (Blocksatz usw.) mitgezählt, was zu unschönen „Rinnsalen", wie der Setzer sagt, führen kann.

Der nachfolgend beschriebene Druckertreiber, den Sie in Verbindung mit dem Fast-Writer bei *jedem* Drucker einsetzen können, soll Ihnen den Umgang mit Esc-Sequenzen durch die Definition spezieller Drucker-Makros erleichtern. Theoretisch können Sie den Druckertreiber auch bei anderen Textverarbeitungsprogrammen einsetzen, falls diese die Einbindung von Treibern mit entsprechenden Makro-Puffern zulassen, was jedoch aus verschiedenen Gründen (Kopierschutz, Speicherorganisation usw.) oft nicht möglich ist, etwa bei Appleworks oder beim Applewriter.

1. Ein- und Ausgangscode

1. Der **Eingangscode** darf aus einem einzigen oder aus maximal zwei Zeichen bestehen. Dies ist keine Beschränkung, denn wenn man sich unnötige Tipparbeit ersparen will, muß der Eingangscode so kurz wie möglich sein. Legt man beispielsweise als *Vorcode*, d.h. als erstes Zeichen eines 2-Zeichen-Makros, das Paragraph-Zeichen (§) fest, so können damit 127 *Nachcodes* kombiniert werden. Würde dies nicht ausreichen, so könnte man noch einen weiteren Vorcode einführen, z.B. das Nummernzeichen (#), womit weitere 127 Makros entstehen würden (#A, #B, #C ... #a, #b, #c usw.).
Die Tastatur des Apple IIe/c kann 128 verschiedene Zeichen erzeugen, nämlich 32 Ctrl-Zeichen ($00-$1F bzw. 0-31) sowie 96 sichtbare Zeichen (Buchstaben, Ziffern und Sonderzeichen). In bestimmten Texten kommt das eine oder andere sichtbare Sonderzeichen, z.B. „↑" u.a., niemals vor. Diese brachliegenden Sonderzeichen können dann als 1-Zeichen-Eingangscodes verwendet werden, womit sich die Tipparbeit für bestimmte Esc-Sequenzen auf jeweils einen einzigen Tastendruck reduziert. Darüber hinaus kann man verschiedene Ctrl-Zeichen als 1-Zeichen-Makros verwenden. Beim Fast-Writer lassen sich übrigens die vier Ctrl-Zeichen
Ctrl-_
Ctrl-↑
Ctrl-Ö
Ctrl-Ü
direkt, also nicht über Ctrl-V, eintippen, so daß sie sich als weitere 1-Zeichen-Makros anbieten.

2. Der **Ausgangscode** kann 0 bis 115 Zeichen umfassen, so daß praktisch beliebig lange Esc-Sequenzen definiert werden können. Wird kein Ausgangscode angegeben („0 Zeichen"), dann wird der Eingangscode eliminiert. Auf diese Weise lassen sich 1- und 2-Zeichen-Kombinationen entfernen, die aus anderen Gründen in den Text eingestreut sind, z.B. bei Dateien, die zusätzlich für die Belichtung

in Setzereien codiert sind. (Die Beschränkung auf 115 Zeichen liegt an dem Applesoft-Programm MACRO.BIN.MAKER und nicht an dem Druckertreiber DRIVER, der Makros mit einer Länge von ca. 250 Zeichen verarbeiten könnte.)

2. Makro-Tabelle

Die Makro-Tabelle wird als normale Fast-Writer-Datei erstellt und besteht aus der Summe der Makro-Definitionszeilen, die am Schluß durch einen sog. Endmarker („00") abgeschlossen werden. Da die direkte Eingabe von Ctrl-Zeichen sehr unübersichtlich wäre, muß die Makro-Tabelle in hexadezimaler Notation angelegt werden. Wir wollen dies anhand der **Abb. 1** erläutern. Betrachten wir hierzu die 1. Zeile

```
4Ø61 Ø7 1B52Ø04Ø1B52Ø2  §a
E.   L. A.              K.
```

Jede Makro-Definition umfaßt bis zu 4 Teile:

E. = Eingangscode: Der Eingangscode besteht aus zwei 2stelligen hexadezimalen Zahlen (hier „4061"), der somit vier hexadezimale Ziffern umfaßt und ohne Leertaste eingegeben werden muß. Die hexadezimalen Zahlen müssen im Bereich $01 bis $7F liegen (also nicht $00 und nicht $80-$FF), weil sie sonst nicht erkannt werden würden, denn der Fast-Writer kennt nur die Zeichen $01-$7F bzw. $81-$FF, die vom Druckertreiber auf $01-$7F normiert werden. Zum besseren Verständnis nehmen Sie eine ASCII-Tabelle zur Hand (s. **Abb. 4**). In dem obigen Beispiel steht „4061" für „§a". Wenn der Eingangscode nur aus *einem* Zeichen besteht, so muß als zweite hexadezimale Zahl „00" angegeben werden. Man beachte hierzu die letzten sieben Zeilen in Abb. 1.

L. = Länge: Nach dem Eingangscode wird die Länge des Ausgangscodes als 2stellige Hex-Zahl angegeben, die links und rechts von einer Leertaste begrenzt wird (in der obigen Zeile „ 07 "). Beachten Sie, daß ein Ausgangscode, der 10 Hex-Zahlen umfaßt, eine Länge von „0A" und nicht von „10" hat, denn „10" wäre eine Dezimalzahl.

A. = Ausgangscode: Der Ausgangscode besteht aus einer Folge von 2stelligen Hex-Zahlen, die der Steuerzeichenfolge (meist Esc-Sequenz) des jeweiligen Drukkers entspricht. Insgesamt können etwa 100 2stellige Hex-Zahlen eingegeben werden. Die hexadezimalen Zahlen des Ausgangscodes können im Bereich $00-$FF liegen, vorausgesetzt Ihr Drucker-Interface ist in der Lage, alle 256 verschiedenen ASCII-Codes zu senden. Dies ist jedoch meist nicht der Fall, weil das Interface intern fast immer Bit 7 löscht und somit ASCII-Codes im Bereich $80-$FF in ASCII-Codes im Bereich $00-$7F umwandelt. Allerdings besteht bei manchen Druckern die Möglichkeit, über Esc-Sequenzen den Bit-7-Status zu kontrollieren. Davon wurde bei den Makro-Definitionen in der Mitte der Abb. 1 (§b, §- usw.) Gebrauch gemacht. Dies ist allerdings nicht trivial, sondern setzt vielmehr genaue Kenntnisse des jeweiligen Druckers voraus.
Wie oben bereits erwähnt wurde, kann der Ausgangscode auf Null gesetzt werden (s. Abb 1: §T, §L), d.h. das Ausgangscode-Feld bleibt dann leer, und als Länge wird „00" eingetragen.

K. = Kommentar: Der Kommentar zur Makro-Definition ist optional. Wenn man von der Kommentarspalte Gebrauch macht, so beachte man, daß man keine Kommas und keine Doppelpunkte verwendet, weil das Konvertierungsprogramm MACRO.BIN.MAKER aus speichertechnischen Gründen sehr kurz gehalten werden mußte und die Makro-Datei in Applesoft-BASIC eingelesen wird.

Jede Makro-Definitionszeile (Eingangscode, Länge, Ausgangscode, Kommentar) muß mit Return abgeschlossen werden. Am Schluß setzen Sie dann „00" als Endmarkierung auf eine eigene Zeile, gefolgt von Return. Alles, was nach „00" folgt, wird ignoriert. Sie können deshalb auch „00" benutzen, um Teile der Makro-Tabelle lahmzulegen. „00" in der ersten Zeile der Makro-Tabelle würde bedeuten, daß überhaupt keine Makros definiert sind.

Abb. 2 zeigt einen Testtext, wie man ihn am Bildschirm sieht oder wie er ohne Druckertreiber ausgedruckt werden würde. **Abb. 3** zeigt denselben Testtext, der mit dem Druckertreiber unter Verwendung der Makro-Tabelle (s. Abb. 1) auf dem Epson-LQ-800 ausgedruckt worden ist.

```
MACRO.TXT

4061 07 1B5200401B5202              §a At-Zeichen
403C 07 1B52005B1B5202              §< Eckige Klammer auf
402F 07 1B52005C1B5202              §/ Schrägstrich nach links
403E 07 1B52005D1B5202              §> Eckige Klammer zu
4028 07 1B52007B1B5202              §( Geschweifte Klammer auf
4021 07 1B52007C1B5202              §! Senkrechter Strich
4029 07 1B52007D1B5202              §) Geschweifte Klammer zu
4073 07 1B5205241B5202              §s Lichtes Kästchen (schwedisch)
4062 0D 1B74011B371B3E7E1B74001B23  §b Volles Kästchen (ESC-P)
402D 0D 1B74011B371B3EC41B74001B23  §- Gedankenstrich (ESC-P)
4022 0D 1B74011B371B3EAE1B74001B23  §" Französische Anführung
4060 0D 1B74011B371B3EAF1B74001B23  §' Französische Abführung
4068 04 1B451B47                    §h halbfett als Doppeldruck
406B 04 1B341B47                    §k kursiv als Doppeldruck
4031 03 0E1B50                      §1 Breitschrift Pitch 10
4032 03 0E1B4D                      §2 Breitschrift Pitch 12
4033 03 0E1B67                      §3 Breitschrift Pitch 15
4030 08 1B461B481B351B50            §0 Grundschrift Pitch 10
403F 01 40                          §? Paragraph-Zeichen
4040 01 00                          §§ Ctrl-0
4054 00                             §T entfernen
404C 00                             §L entfernen
6000 01 22                          "  Anführungszeichen
5F00 03 1B2D01                      _  Unterstreichen ein
1F00 03 1B2D00                      C_ Unterstreichen aus (Ctrl-_)
5E00 03 1B5300                      ^  Hochstellen ein
1E00 02 1B54                        C^ Hochstellen aus
1C00 03 1B5301                      CÖ Tiefstellen ein
1D00 02 1B54                        CÜ Tiefstellen aus
00
Im Kommentarfeld keine Kommas und keine Doppelpunkte!
```

Abb. 1. Makro-Tabelle

3. Installation

Auf der Peeker-Sammeldiskette befinden sich die Dateien T.DRIVER, DRIVER, MACRO.BIN.MAKER, START.FW. MACRO, MACRO.TXT, MACRO.BIN und TESTTEXT. Die weitere Vorgehensweise ist wie folgt:

Schritt 1: Kopieren Sie sich die obigen Dateien auf die Kopie Ihrer Fast-Writer-Originaldiskette.
Schritt 2: Starten Sie das Konfigurationsprogramm CONFIG.FW, wählen Sie „D = Speicheraufteilung", dann „A = Pufferstart" und geben Sie hier
$1000
ein. Speichern Sie dann im Hauptmenü den Fast-Writer auf Diskette zurück, verlassen Sie das Konfigurationsprogramm und ändern Sie den neuen Fast-Writer mit

RENAME FAST.WRITER, FAST.WRITER.MAC
in die speicherverteilungsmäßig für den Druckertreiber angepaßte Fast-Writer-Version ab.

Schritt 3: Starten Sie nunmehr den Fast-Writer mit
RUN START.FW.MACRO
und erstellen Sie sodann eine Makro-Tabelle für Ihren speziellen Matrixdrucker aufgrund der Esc-Sequenzen, die in Ihrem Druckerhandbuch beschrieben sind. Diese Makro-Tabelle speichern Sie dann unter dem Namen MACRO.TXT auf der Diskette ab. Falls Sie zufällig einen Epson-LQ-800-Drucker besitzen sollten, so können Sie die in Abb. 1 abgedruckte und auf der Sammeldiskette gespeicherte Makro-Tabelle unverändert übernehmen und bei Bedarf erweitern. Sie gilt zum Teil auch für andere Epson-Drucker.

Schritt 4: Verlassen Sie nach dem Abspeichern von MACRO.TXT den Fast-Writer und starten Sie mit
RUN MACRO.BIN.MAKER
das Konvertierungsprogramm, das automatisch den Textfile MACRO.TXT in den Binärfile MACRO.BIN konvertiert und abspeichert. Wenn MACRO.TXT nicht korrekt angelegt worden ist, erhalten Sie eine entsprechende Fehlermeldung. In diesem Falle tippen Sie
CLOSE
und starten mit
CALL 768
den Fast-Writer erneut. Lesen Sie nun MACRO.TXT erneut ein und prüfen Sie, was Sie falsch getippt haben (zusätzliche Returns, keine Hex-Ziffern, sondern andere Buchstaben in den Eingangs- und Ausgangscodes usw.).
Sobald die Datei MACRO.BIN korrekt angelegt worden ist, können Sie mit
RUN START.FW.MACRO
den Fast-Writer einschließlich Druckertreiber (DRIVER) und Makro-Tabelle (MACRO.BIN) direkt starten.

4. Technisches

Slot: Der Druckertreiber ist für Slot 1 eingestellt. In Speicherstelle $0AE2 kann man bei Bedarf eine andere Slotnummer eintragen. Für Slot 2 würde gelten:
BLOAD DRIVER
CALL -151
AE2:2
BSAVE DRIVER, A$A00,L$100
Bit 7: Wenn Bit 7 nicht gesetzt werden soll, so füge man mit A 62: EA EA zwei NOPs ein.

C100: Der Druckertreiber geht über die normale Schnittstelle ($C100 für Slot 1 usw.). Dies ist das einzige Verfahren, das bei allen Interface-Karten und somit auch

```
§T
Driver-Test
At-Zeichen §a und Paragraph-Zeichen §?
§<Eckige Klammern§>
§(Geschweifte Klammern§)
Schrägstrich nach rechts /
Schrägstrich nach links §/
Senkrechter Strich §!
Gedanken §- Strich und Binde-Strich
"Deutsche Anführungszeichen'
§"Französische Anführungszeichen§'
§hDies ist halbfett gedruckt§0
§kDies ist kursiv gedruckt§0
_Dies ist unterstrichen
hochgestellt ^123
tiefgestellt 123
lichtes Kästchen §s
volles Kästchen §b
§1Breitschrift Pitch 10§0
§2Breitschrift Pitch 12§0
§3Breitschrift Pitch 15§0
Ende des Tests
§L
```

Abb. 2. Testtext (ohne Druckertreiber ausgedruckt)

```
Driver-Test
At-Zeichen @ und Paragraph-Zeichen §
[Eckige Klammern]
{Geschweifte Klammern}
Schrägstrich nach rechts /
Schrägstrich nach links \
Senkrechter Strich ¦
Gedanken – Strich und Binde-Strich
"Deutsche Anführungszeichen"
«Französische Anführungszeichen»
Dies ist halbfett gedruckt
Dies ist kursiv gedruckt
Dies ist unterstrichen
hochgestellt 123
tiefgestellt 123
lichtes Kästchen ¤
volles Kästchen ■
Breitschrift Pitch 10
Breitschrift Pitch 12
Breitschrift Pitch 15
Ende des Tests
```

Abb. 3. Testtext (mit Druckertreiber ausgedruckt)

	@§					@§					@§					@§			
NUL	@§	00	00000000	000		@§	40	01000000	064		@§	80	10000000	128		@§	C0	11000000	192
	A	01	00000001	001		A	41	01000001	065		A	81	10000001	129		A	C1	11000001	193
	B	02	00000010	002		B	42	01000010	066		B	82	10000010	130		B	C2	11000010	194
	C	03	00000011	003		C	43	01000011	067		C	83	10000011	131		C	C3	11000011	195
EOT	D	04	00000100	004		D	44	01000100	068		D	84	10000100	132		D	C4	11000100	196
	E	05	00000101	005		E	45	01000101	069		E	85	10000101	133		E	C5	11000101	197
	F	06	00000110	006		F	46	01000110	070		F	86	10000110	134		F	C6	11000110	198
BELL	G	07	00000111	007		G	47	01000111	071		G	87	10000111	135		G	C7	11000111	199
←	H	08	00001000	008		H	48	01001000	072		H	88	10001000	136		H	C8	11001000	200
TAB	I	09	00001001	009		I	49	01001001	073		I	89	10001001	137		I	C9	11001001	201
↓	J	0A	00001010	010		J	4A	01001010	074		J	8A	10001010	138		J	CA	11001010	202
↑	K	0B	00001011	011		K	4B	01001011	075		K	8B	10001011	139		K	CB	11001011	203
FF	L	0C	00001100	012		L	4C	01001100	076		L	8C	10001100	140		L	CC	11001100	204
RTN	M	0D	00001101	013		M	4D	01001101	077		M	8D	10001101	141		M	CD	11001101	205
SO	N	0E	00001110	014		N	4E	01001110	078		N	8E	10001110	142		N	CE	11001110	206
SI	O	0F	00001111	015		O	4F	01001111	079		O	8F	10001111	143		O	CF	11001111	207
	P	10	00010000	016		P	50	01010000	080		P	90	10010000	144		P	D0	11010000	208
XON	Q	11	00010001	017		Q	51	01010001	081		Q	91	10010001	145		Q	D1	11010001	209
	R	12	00010010	018		R	52	01010010	082		R	92	10010010	146		R	D2	11010010	210
XOFF	S	13	00010011	019		S	53	01010011	083		S	93	10010011	147		S	D3	11010011	211
	T	14	00010100	020		T	54	01010100	084		T	94	10010100	148		T	D4	11010100	212
	U	15	00010101	021		U	55	01010101	085		U	95	10010101	149		U	D5	11010101	213
	V	16	00010110	022		V	56	01010110	086		V	96	10010110	150		V	D6	11010110	214
	W	17	00010111	023		W	57	01010111	087		W	97	10010111	151		W	D7	11010111	215
	X	18	00011000	024		X	58	01011000	088		X	98	10011000	152		X	D8	11011000	216
	Y	19	00011001	025		Y	59	01011001	089		Y	99	10011001	153		Y	D9	11011001	217
	Z	1A	00011010	026		Z	5A	01011010	090		Z	9A	10011010	154		Z	DA	11011010	218
ESC	[Ä	1B	00011011	027		[Ä	5B	01011011	091		[Ä	9B	10011011	155		[Ä	DB	11011011	219
	\ Ö	1C	00011100	028		\ Ö	5C	01011100	092		\ Ö	9C	10011100	156		\ Ö	DC	11011100	220
	] Ü	1D	00011101	029		] Ü	5D	01011101	093		] Ü	9D	10011101	157		] Ü	DD	11011101	221
	↑	1E	00011110	030		↑	5E	01011110	094		↑	9E	10011110	158		↑	DE	11011110	222
	_	1F	00011111	031		_	5F	01011111	095		_	9F	10011111	159		_	DF	11011111	223
		20	00100000	032		'	60	01100000	096			A0	10100000	160		'	E0	11100000	224
	!	21	00100001	033		a	61	01100001	097		!	A1	10100001	161		a	E1	11100001	225
	"	22	00100010	034		b	62	01100010	098		"	A2	10100010	162		b	E2	11100010	226
	#	23	00100011	035		c	63	01100011	099		#	A3	10100011	163		c	E3	11100011	227
	$	24	00100100	036		d	64	01100100	100		$	A4	10100100	164		d	E4	11100100	228
	%	25	00100101	037		e	65	01100101	101		%	A5	10100101	165		e	E5	11100101	229
	&	26	00100110	038		f	66	01100110	102		&	A6	10100110	166		f	E6	11100110	230
	'	27	00100111	039		g	67	01100111	103		'	A7	10100111	167		g	E7	11100111	231
	(	28	00101000	040		h	68	01101000	104		(	A8	10101000	168		h	E8	11101000	232
	)	29	00101001	041		i	69	01101001	105		)	A9	10101001	169		i	E9	11101001	233
	*	2A	00101010	042		j	6A	01101010	106		*	AA	10101010	170		j	EA	11101010	234
	+	2B	00101011	043		k	6B	01101011	107		+	AB	10101011	171		k	EB	11101011	235
	,	2C	00101100	044		l	6C	01101100	108		,	AC	10101100	172		l	EC	11101100	236
	–	2D	00101101	045		m	6D	01101101	109		–	AD	10101101	173		m	ED	11101101	237
	.	2E	00101110	046		n	6E	01101110	110		.	AE	10101110	174		n	EE	11101110	238
	/	2F	00101111	047		o	6F	01101111	111		/	AF	10101111	175		o	EF	11101111	239
	0	30	00110000	048		p	70	01110000	112		0	B0	10110000	176		p	F0	11110000	240
	1	31	00110001	049		q	71	01110001	113		1	B1	10110001	177		q	F1	11110001	241
	2	32	00110010	050		r	72	01110010	114		2	B2	10110010	178		r	F2	11110010	242
	3	33	00110011	051		s	73	01110011	115		3	B3	10110011	179		s	F3	11110011	243
	4	34	00110100	052		t	74	01110100	116		4	B4	10110100	180		t	F4	11110100	244
	5	35	00110101	053		u	75	01110101	117		5	B5	10110101	181		u	F5	11110101	245
	6	36	00110110	054		v	76	01110110	118		6	B6	10110110	182		v	F6	11110110	246
	7	37	00110111	055		w	77	01110111	119		7	B7	10110111	183		w	F7	11110111	247
	8	38	00111000	056		x	78	01111000	120		8	B8	10111000	184		x	F8	11111000	248
	9	39	00111001	057		y	79	01111001	121		9	B9	10111001	185		y	F9	11111001	249
	:	3A	00111010	058		z	7A	01111010	122		:	BA	10111010	186		z	FA	11111010	250
	;	3B	00111011	059		{ ä	7B	01111011	123		;	BB	10111011	187		{ ä	FB	11111011	251
	<	3C	00111100	060		| ö	7C	01111100	124		<	BC	10111100	188		| ö	FC	11111100	252
	=	3D	00111101	061		} ü	7D	01111101	125		=	BD	10111101	189		} ü	FD	11111101	253
	>	3E	00111110	062		~ ß	7E	01111110	126		>	BE	10111110	190		~ ß	FE	11111110	254
	?	3F	00111111	063		DEL	7F	01111111	127		?	BF	10111111	191		DEL	FF	11111111	255

Abb. 4. ASCII-Tabelle

bei allen Druckern funktioniert. Es hat jedoch den Nachteil, daß in der Regel nur die ASCII-Codes im Bereich $00 bis $7F gesendet werden können. Assembler-Programmierer, die mit ihrer Schnittstelle vertraut sind, können im DRIVER-Listing ab Zeile 61 eine Senderoutine einbauen, die die Data- und Ready-Adressen der Schnittstelle direkt anspricht. Dies ist leider bei jedem Interface anders, so daß hier keine globale Lösung angeboten werden kann.

Speicherverteilung: Der Druckertreiber DRIVER belegt den Bereich $0A00 bis $0AFF. Für die Makro-Tabelle steht dann der Bereich $0B00 bis $0FFF (1280 Bytes) zur Verfügung. Größere Makro-Tabellen sind möglich, wenn man über CONFIG.FW den Pufferstart heraufsetzt und im DRIVER-Listing, Zeile 89, die Obergrenze erhöht.

1-Zeichen-Makro: Wenn man neben 2-Zeichen-Makros, die beispielsweise mit „§" beginnen, z.B. „§a", „§b" usw., auch den Vorcode „§" *allein* umdefinieren möchte (z.B. „§" bedeute „xxx", sofern nicht „a", „b" usw. folgen), dann muß dieses 1-Zeichen-Makro in der Makro-Tabelle MACRO.TXT *nach* den 2-Zeichen-Makros definiert werden, weil sonst die Makro-Suche bereits bei „§" abgebrochen werden würde. Im übrigen sollte man diesen zweideutigen Sonderfall vermeiden.

Letztes Zeichen: Da der Eingangscode bei unserem Druckertreiber bis zu 2 Zeichen umfassen kann, wird ein Minipuffer angelegt, der jeweils die zwei letzten Zeichen enthält. Da der Druckertreiber nicht weiß, wann der auszudruckende Fast-Writer-Text zu Ende ist, geht auf diese Weise das jeweils letzte Zeichen des Textes verloren, es sei denn, die letzten beiden Zeichen des Textes wären ein Makro. Bevor man also mit dem Fast-Writer einen Text ausdruckt, springe man mit Ctrl-E zum Textende und tippe eine Leertaste oder ein Return.

ProDOS: Beim ProDOS-Fast-Writer beginnt der Pufferstart nicht bei $0A00, sondern bei $0E00. Folglich müssen DRIVER und MACRO.BIN nach oben verlagert werden. Entsprechende ProDOS-Versionen aller soeben beschriebenen Programme befinden sich auf der Original-Diskette des ProDOS-Fast-Writers, während die DOS-Programme auf die Peeker-Sammeldiskette aufgenommen worden sind, da der DOS-Fast-Writer bereits ausgeliefert worden ist.

Kurzhinweise

1. Zweck:
Druckertreiber zur Konvertierung von Makros in Esc-Sequenzen für den Fast-Writer oder andere Textverarbeitungsprogramme, die die Einbindung von Druckertreibern zulassen.
2. Konfiguration:
IIe/c für Fast-Writer; für andere Textprogramme auch II+; nur DOS 3.3 (zu ProDOS s. Aufsatz!)
3. Test:
siehe Aufsatz; TESTTEXT kann auf Epson-LQ-800 direkt ausprobiert werden.
4. Sammeldisk:
T.DRIVER
DRIVER
MACRO.BIN.MAKER
START.FW.MACRO
MACRO.TXT
MACRO.BIN
TESTTEXT

START.FW.MACRO

```
10 PRINT CHR$ (4);"BLOAD DRIVER":
   PRINT CHR$ (4);"BLOAD MACRO.BIN":
   PRINT CHR$ (4);"BRUN FAST.WRITER.MAC"
```

MACRO.BIN.MAKER

```
1 F$ = "MACRO.TXT":A = 2816:B = A
2 HOME : PRINT "MAKRO.BIN ERZEUGEN...":
  PRINT CHR$ (4);"OPEN";F$: PRINT CHR$ (4);"READ";F$
3 INPUT X$: IF LEFT$ (X$,2) = "00" THEN POKE B,0:
  PRINT CHR$ (4);"CLOSE":
  PRINT CHR$ (4);"BSAVE MACRO.BIN,A";A;",L";B + 1 - A: END
4 H$ = LEFT$ (X$,2): GOSUB 6:H$ = MID$ (X$,3,2): GOSUB 6:
  H$ = MID$ (X$,6,2): GOSUB 6:Y = H - 1: IF Y < 0 GOTO 3
5 FOR X = 0 TO Y:H$ = MID$ (X$,9 + X * 2,2): GOSUB 6: NEXT X:
  GOTO 3
6 LN = ASC ( LEFT$ (H$,1)):RN = ASC ( RIGHT$ (H$,1)):
  N = LN: GOSUB 7:H = N * 16:N = RN: GOSUB 7:H = H + N:
  POKE B,H:B = B + 1: RETURN
7 IF N > = 65 AND N < = 70 THEN N = N - 55: RETURN
8 IF N > = 48 AND N < = 57 THEN N = N - 48: RETURN
9 PRINT CHR$ (4);"CLOSE": PRINT CHR$ (7);
  "MAKRO.TXT FEHLERHAFT!": END
```

DRIVER

```
BSAVE DRIVER, A$0A00,L$0100
```

```
                1               ORG  $0A00
                2   *
                3   * DRIVER für FAST.WRITER/us
                4   * ========================
                5   *
                6   INDL        EQU  $0000
                7   INDH        EQU  $0001
                8   CSWL        EQU  $0036
                9   CSWH        EQU  $0037
                10  TAB         EQU  $0B00
                11  *
                12  * Hier Einsprung vor 1. Zeichen
                13  *
0A00: 20 55 0A  14  ENTRY1      JSR  REGSAVE     ;$0A00
0A03: A9 00     15              LDA  #$00        ;SC100
0A05: 85 36     16              STA  CSWL
0A07: AD E2 0A  17              LDA  SLOT
0A0A: 09 C0     18              ORA  #$C0
0A0C: 85 37     19              STA  CSWH
                20  *
                21  * Vorab Return ausgeben, um
                22  * Ausgabevektor umzulenken
                23  *
0A0E: A9 8D     24              LDA  #$8D        ;Return
0A10: 20 5F 0A  25              JSR  OUTPUT1
0A13: A5 36     26              LDA  CSWL
0A15: 8D 65 0A  27              STA  OUTPUT2+1
0A18: A5 37     28              LDA  CSWH
0A1A: 8D 66 0A  29              STA  OUTPUT2+2
0A1D: A9 2D     30              LDA  #<ENTRY2
0A1F: 85 36     31              STA  CSWL
0A21: A9 0A     32              LDA  #>ENTRY2
0A23: 85 37     33              STA  CSWH
0A25: A9 00     34              LDA  #0
0A27: 8D E6 0A  35              STA  CNT
0A2A: 20 4B 0A  36              JSR  REGLOAD
                37  *
                38  * Hier Einsprung nach 1. Zeichen
                39  *
0A2D: 20 55 0A  40  ENTRY2      JSR  REGSAVE
0A30: AD E5 0A  41              LDA  CHAR2
0A33: 8D E4 0A  42              STA  CHAR1
0A36: AD E3 0A  43              LDA  CHAR0
0A39: 29 7F     44              AND  #$7F
0A3B: 8D E5 0A  45              STA  CHAR2
0A3E: EE E6 0A  46              INC  CNT
0A41: AD E6 0A  47              LDA  CNT
0A44: C9 02     48              CMP  #2
0A46: D0 03     49              BNE  REGLOAD
0A48: 20 67 0A  50              JSR  MACRO1
                51  *
0A4B: AD E3 0A  52  REGLOAD     LDA  CHAR0
0A4E: AE E7 0A  53              LDX  XSAVE
0A51: AC E8 0A  54              LDY  YSAVE
0A54: 60        55              RTS
0A55: 8D E3 0A  56  REGSAVE     STA  CHAR0
0A58: 8E E7 0A  57              STX  XSAVE
0A5B: 8C E8 0A  58              STY  YSAVE
0A5E: 60        59              RTS
0A5F: 6C 36 00  60  OUTPUT1     JMP  (CSWL)
0A62: 09 80     61  OUT         ORA  #$80
```

```
ØA64: 4C FF FF  62   OUTPUT2  JMP  $FFFF
                63   *
ØA67: 2Ø D7 ØA  64   MACRO1   JSR  INDSAVE
ØA6A: A9 ØØ     65            LDA  #<TAB
ØA6C: 85 ØØ     66            STA  INDL
ØA6E: A9 ØB     67            LDA  #>TAB
ØA7Ø: 85 Ø1     68            STA  INDH
ØA72: AØ ØØ     69   MACRO2   LDY  #Ø
ØA74: B1 ØØ     7Ø            LDA  (INDL),Y   ;Code1
ØA76: DØ ØB     71            BNE  MACRO3     ;Ctrl-Ø?
ØA78: AD E4 ØA  72            LDA  CHAR1      ;kein
ØA7B: 2Ø 62 ØA  73            JSR  OUT        ;Makro
ØA7E: CE E6 ØA  74            DEC  CNT        ;CNT=1
ØA81: DØ 49     75            BNE  INDLOAD
                76   *
ØA83: CD E4 ØA  77   MACRO3   CMP  CHAR1      ;1.Char?
ØA86: FØ 17     78            BEQ  MACRO5
                79   *
ØA88: AØ Ø2     8Ø   MACRO4   LDY  #2         ;weiter-
ØA8A: B1 ØØ     81            LDA  (INDL),Y   ;suchen
ØA8C: 18        82            CLC
ØA8D: 69 Ø3     83            ADC  #3         ;3+Länge
ØA8F: 65 ØØ     84            ADC  INDL
ØA91: 85 ØØ     85            STA  INDL
ØA93: 9Ø DD     86            BCC  MACRO2
ØA95: E6 Ø1     87            INC  INDH
ØA97: A5 Ø1     88            LDA  INDH
ØA99: C9 1Ø     89            CMP  #>$1ØØØ    ;<$ØFFF?
ØA9B: 9Ø D5     9Ø            BCC  MACRO2
ØA9D: BØ Ø8     91            BCS  OVERFL     ;Überlauf
                92   *
ØA9F: C8        93   MACRO5   INY
ØAAØ: B1 ØØ     94            LDA  (INDL),Y   ;Code2
ØAA2: DØ Ø8     95            BNE  MACRO6     ;Ctrl-Ø?
ØAA4: 2Ø BB ØA  96            JSR  MACRO7
ØAA7: CE E6 ØA  97   OVERFL   DEC  CNT        ;CNT=1
ØAAA: DØ 2Ø     98            BNE  INDLOAD
                99   *
ØAAC: CD E5 ØA  1ØØ  MACRO6   CMP  CHAR2      ;2.Char
ØAAF: DØ D7     1Ø1           BNE  MACRO4
ØAB1: 2Ø BB ØA  1Ø2           JSR  MACRO7
```

```
ØAB4: A9 ØØ     1Ø3           LDA  #Ø
ØAB6: 8D E6 ØA  1Ø4           STA  CNT        ;CNT=Ø
ØAB9: FØ 11     1Ø5           BEQ  INDLOAD
                1Ø6  *
ØABB: C8        1Ø7  MACRO7   INY             ;Makro-
ØABC: B1 ØØ     1Ø8           LDA  (INDL),Y   ;ausgabe
ØABE: AA        1Ø9           TAX
ØABF: DØ Ø1     11Ø           BNE  MACRO8
ØAC1: 6Ø        111           RTS             ;leer
ØAC2: C8        112  MACRO8   INY
ØAC3: B1 ØØ     113           LDA  (INDL),Y
ØAC5: 2Ø 62 ØA  114           JSR  OUT
ØAC8: CA        115           DEX
ØAC9: DØ F7     116           BNE  MACRO8
ØACB: 6Ø        117           RTS
                118  *
ØACC: AD E9 ØA  119  INDLOAD  LDA  INDLSAV
ØACF: 85 ØØ     12Ø           STA  INDL
ØAD1: AD EA ØA  121           LDA  INDHSAV
ØAD4: 85 Ø1     122           STA  INDH
ØAD6: 6Ø        123           RTS
ØAD7: A5 ØØ     124  INDSAVE  LDA  INDL
ØAD9: 8D E9 ØA  125           STA  INDLSAV
ØADC: A5 Ø1     126           LDA  INDH
ØADE: 8D EA ØA  127           STA  INDHSAV
ØAE1: 6Ø        128           RTS
ØAE2: Ø1        129  SLOT     HEX  Ø1         ;Slot
ØAE3: ØØ        13Ø  CHARØ    HEX  ØØ         ;Char
ØAE4: ØØ        131  CHAR1    HEX  ØØ         ;vorher
ØAE5: ØØ        132  CHAR2    HEX  ØØ         ;jetzt
ØAE6: ØØ        133  CNT      HEX  ØØ         ;Zähler
ØAE7: ØØ        134  XSAVE    HEX  ØØ
ØAE8: ØØ        135  YSAVE    HEX  ØØ
ØAE9: ØØ        136  INDLSAV  HEX  ØØ
ØAEA: ØØ        137  INDHSAV  HEX  ØØ
                138  * Bis auf $ØAFF auffüllen
                139           DS   2Ø
ØAFF: ØØ        14Ø  ADR_ØAFF HEX  ØØ         ;$ØAFF?

256 Bytes
```

Die BASIC-Maske

Ein raffiniertes BASIC-Programm macht Eingaben leicht

von Christiane und Jürgen Kehrel

Einsatzgebiet

Diese Eingaberoutine dient zum bequemen Bearbeiten von Eingabefeldern in Bildschirmmasken. Dazu werden meist längere Maschinenprogramme benötigt. Hier wird gezeigt, daß auch in BASIC eine durchaus professionelle Lösung möglich ist, die fast alle Eigenschaften ihrer großen „Schwestern" besitzt.

Die BASIC-Maske kann als Unterroutine in jedes größere BASIC-Programm eingebunden und leicht nach eigenen Vorstellungen verändert werden. Unsere Routine verlangt folgende Parameter, die vom Hauptprogramm übergeben werden müssen:
– der vertikale Tabulator (VT) und der horizontale Tabulator (HT), die zusammen die Position des ersten Zeichens auf dem 40-Z/Z-Bildschirm bestimmen
– die maximale Länge (LE) der einzugebenden Zeichenkette
– den String-Inhalt (CC$). Normalerweise muß CC$ „leer" sein (CC$ = ""). Um sich wiederholende Eingaben zu vermeiden, ist es aber auch möglich, einen String vorab festzulegen (Default-String), der mit einem einfachen Return übernommen wird.
Mit den Cursortasten kann man nach links, nach rechts, mit Ctrl-B an den Beginn und Ctrl-C an das Ende einer noch nicht mit <Return> abgeschlossenen Eingabe gehen. Bei mehrzeiligen Feldern funktionieren auch die Hoch- und Tiefpfeile. Besitzer eines alten II+ betätigen ersatzweise Ctrl-J und Ctrl-K. Es ist weiterhin möglich, nach Belieben bis zur maximalen Länge anzufügen, einzufügen, zu löschen oder zu überschreiben.

Programmablauf

Um zu zeigen, wie die Routine eingebunden wird, wählten wir ein kurzes Demonstrationsprogramm, das Adresseneingaben mit Notizen bis zu 245 Zeichen erlaubt.
Im Hauptprogramm legen wir die Form der Maske fest, die dann mit dem Aufruf der BASIC-Routine (GOSUB 60000) gefüllt werden kann.
Mit dem GET-Befehl in Zeile 60340 werden alle Tastendrücke eingesammelt, bevor durch eine Folge von „IF... THEN"-Bedingungen die einzelnen Befehlszeichen ausgesondert werden. Ihre Abarbeitung erfolgt in eigenen kleinen Unterprogrammen. Bei allen druckbaren Zeichen springt das Programm in die Anfüge-Routine und gibt über den Rücksprung nach Zeile 60330 den Buchstaben auf dem Bildschirm aus. Gleichzeitig wird der Cursorzähler M intern eine Position weitergesetzt. Wollen Sie Verbesserungen an Ihrem Text vornehmen, so verzweigt das Programm entsprechend Ihrer Ctrl-Eingabe. Nach allen Cursor-Bewegungen springt das Programm wieder nach Zeile 60340 zurück, um auf einen neuen Tastendruck zu warten.
Nach einem Return oder bei Überschreitung der maximalen Länge kehren wir direkt in das Hauptprogramm zurück.
Am Beispiel der Löschroutine mit Ctrl-D sei beschrieben, welche verschiedenen Fälle zu berücksichtigen sind. Sie kennt vier verschiedene Möglichkeiten: Wenn M (Cursorposition) und bisherige Eingabelänge (LEN(CC$)) gleich 1 sind, dann sind B$ und E$, der linke und der rechte Teilstring, leer. Steht der Cursor auf der ersten Position eines Wortes, muß lediglich E$ um eine Position nach links verschoben werden. Beim Löschen des letzten Zeichens eines Wortes muß dagegen nur B$ gerettet werden; die Cursorposition bleibt unverändert.
Treffen diese drei Fälle nicht zu, dann befindet sich der Cursor irgendwo mitten im Wort, und wir müssen den String links und rechts vom Cursor trennen, retten und E$ um eine Position nach links verschieben.
Bei der Behandlung der Del-Taste ist es erforderlich, den Cursor auf dem Bildschirm um eine Stelle nach links zu bewegen. Wir drucken dazu einfach ein „Backspace" = CHR$(8) aus.

Falls Sie sich jetzt Sorgen über die Programmlänge machen: Wenn Sie die Eingaberoutine in einen Textfile verwandeln (wie das geht, steht im BASIC-Handbuch), dann können Sie mit einem einfachen „EXEC *Filename*" das komplette Programm mit wenig Tipparbeit immer wieder an andere Programme anfügen. Einzige Voraussetzung: Im Hauptprogramm dürfen die Zeilennummern von 59000 bis 62000 nicht vorkommen.

Kurzhinweise

1. Zweck:
Bearbeiten von Eingabefeldern in Bildschirmmasken
2. Konfiguration:
II+/e/c (40 Z/Z); DOS 3.3 oder ProDOS
3. Test:
RUN BASIC.MASKE
4. Sammeldisk:
BASIC.MASKE

BASIC.MASKE

```
 20 TEXT : PRINT CHR$ (21): HOME
 40 VT = 1:HT = 10:LE = 28: INVERSE : PRINT "NAME:    ";:
    HTAB 35: PRINT ":";: NORMAL :CC$ = "": GOSUB 60000
 60 VT = 4:HT = 10:LE = 18: VTAB 4: HTAB 1: INVERSE : PRINT
    "VORNAME:";: HTAB 28: PRINT ":";: NORMAL :CC$ = "":
    GOSUB 60000
 80 VT = 7:HT = 10:LE = 25: VTAB 7: HTAB 1: INVERSE : PRINT
    "STRASSE:";: HTAB 34: PRINT ":";: NORMAL :CC$ = "":
    GOSUB 60000
100 VT = 10:HT = 10:LE = 28: VTAB 10: HTAB 1: INVERSE :
    PRINT "PLZ/ORT:";:CC$ = "6900 HEIDELBERG": HTAB 38:
    PRINT ":";: NORMAL : GOSUB 60000
120 VT = 13:HT = 10:LE = 13: VTAB 13: HTAB 1: INVERSE :
    PRINT "TELEFON:";: HTAB 23: PRINT ":";: NORMAL :CC$
    = "": GOSUB 60000
140 VT = 17:HT = 10:LE = 245: VTAB 17: HTAB 1: INVERSE :
    PRINT "NOTIZEN:";: NORMAL :CC$ = "": GOSUB 60000
160 END

59000 REM  ************************************
59010 REM  *                                  *
59020 REM  *    ALLGEMEINE EINGABEROUTINE     *
59030 REM  *  CHRISTIANE UND JUERGEN KEHREL    *
59040 REM  *                                  *
59050 REM  *  AUFRUF:                         *
59060 REM  *     VT = VERTIKALER TABULATOR    *
59070 REM  *     HT = HORIZONTALER TABULATOR  *
59080 REM  *     LE = MAXIMALE LAENGE         *
59090 REM  *     (HT + LE < 255 !!!)          *
59100 REM  *     CC$ = DEFAULT-STRING         *
59110 REM  *    (CC$ IST ENTWEDER ALS LEER-   *
59120 REM  *    STRING "" ZU UEBERGEBEN ODER  *
59130 REM  *    MIT DEM GEWUENSCHTEN INHALT)  *
59140 REM  *     GOSUB 60000                  *
59150 REM  *                                  *
59160 REM  *  AUSGABE:                        *
59170 REM  *     CC$ = EINGABE-STRING         *
59180 REM  *                                  *
59190 REM  *  KOMMANDOS:                      *
59200 REM  *     ↑D,LOESCHT ZEICHEN UNTER     *
59210 REM  *        DEM CURSOR                *
59220 REM  *     DEL LOESCHT ZEICHEN VOR      *
59230 REM  *        DEM CURSOR                *
59240 REM  *     ↑I EINFUEGEMODUS EIN         *
59250 REM  *        BELIEBIGES CONTROL-       *
59260 REM  *        ZEICHEN SCHALTET ZURUECK  *
59270 REM  *        ZUM UEBERSCHREIBMODUS     *
59280 REM  *                                  *
59290 REM  *     ALLE 4 CURSORTASTEN AKTIV    *
59300 REM  *     BEIM II+ ERSATZWEISE ↑K      *
59310 REM  *     UND ↑J FUER HOCH/TIEFPFEIL   *
59320 REM  *                                  *
59330 REM  *     ↑B CURSOR ZUM BEGINN         *
59340 REM  *     ↑E CURSOR ANS ENDE           *
59350 REM  *                                  *
59360 REM  *     <RETURN> EINGABE-ENDE        *
59370 REM  *     ↑Q EINGABE-ABBRUCH           *
59380 REM  *                                  *
59390 REM  *  BENUTZTE VARIABLEN:             *
59400 REM  *  B$, E$, C$, C, M, N, CH, CV     *
59410 REM  *                                  *
59420 REM  ************************************
60000 REM
60010 REM *** INITIALISIERUNG ***
60020 REM
60030 M = 1:B$ = "":E$ = "": VTAB VT: HTAB HT: PRINT CC$;
60040 IF LEN (CC$) > 0 THEN M = LEN (CC$) + 1
60050 GOTO 60340
60300 REM
60310 REM *** TASTATURABFRAGE ***
60320 REM
60330 PRINT C$;:M = M + 1
60340 GET C$:C = ASC (C$)
60400 REM
60410 REM *** ZEICHENAUSWERTUNG ***
60420 REM
60430 IF C = 17 THEN GOTO 61700: REM CTRL-Q
60440 IF C = 13 THEN RETURN :    REM <RETURN>
60450 IF C = 8  THEN GOTO 60910: REM CTRL-H
60460 IF C = 21 THEN GOTO 61010: REM CTRL-U
60470 IF C = 11 THEN GOTO 61110: REM CTRL-K
60480 IF C = 10 THEN GOTO 61210: REM CTRL-J
60490 IF C = 2  THEN GOTO 61810: REM CTRL-B
60500 IF C = 5  THEN GOTO 61910: REM CTRL-E
60510 IF C = 9  THEN GOTO 61310: REM CTRL-I
60520 IF C = 4  THEN GOTO 61510: REM CTRL-D
60530 IF C = 127 THEN GOTO 61610:REM DEL
60540 IF C < 32 OR C > 126 THEN CALL - 198: GOTO 60340
60600 REM
60610 REM *** TEST AUF MAX. LAENGE ***
60620 REM
60630 IF M > LEN (CC$) AND LEN (CC$) > LE - 1 THEN RETURN
60700 REM
60710 REM *** ANFUEGE-ROUTINE ***
60720 REM
60730 IF M > LEN (CC$) THEN CC$ = CC$ + C$: GOTO 60330
60800 REM
60810 REM *** UEBERSCHREIB-ROUTINE ***
60820 REM
60830 IF M > 1 THEN GOTO 60860
60840 IF LEN (CC$) = 1 THEN CC$ = C$: GOTO 60330
60850 B$ = "":E$ = RIGHT$ (CC$, LEN (CC$) - 1): GOTO 60880
60860 IF M = LEN (CC$) THEN B$ = LEFT$ (CC$, LEN (CC$) -
      1):E$ = "": GOTO 60880
60870 B$ = LEFT$ (CC$,M - 1):E$ = RIGHT$ (CC$, LEN (CC$) -
      M)
60880 CC$ = B$ + C$ + E$: GOTO 60330
60900 REM
60910 REM *** CURSOR ZURUECK ***
60920 REM
60930 IF M = 1 THEN CALL - 198: GOTO 60340
60940 CH = PEEK (36):CV = PEEK (37)
60950 IF CH > 0 THEN CH = CH - 1:M = M - 1: POKE 36,CH:
      VTAB CV + 1: GOTO 60340
60960 CH = 39:M = M - 1: POKE 36,CH: VTAB CV: GOTO 60340
61000 REM
61010 REM *** CURSOR VORWAERTS ***
61020 REM
61030 IF M > LEN (CC$) THEN CALL - 198: GOTO 60340
61040 CH = PEEK (36):CV = PEEK (37)
61050 IF CH < 39 THEN CH = CH + 1:M = M + 1: POKE 36,CH:
      VTAB CV + 1: GOTO 60340
61060 CH = 0:M = M + 1: POKE 36,CH: VTAB CV + 2: GOTO
      60340
61100 REM
61110 REM *** CURSOR HOCH ***
61120 REM
61130 IF M < 41 THEN CALL - 198: GOTO 60340
61140 CV = PEEK (37):M = M - 40: VTAB CV: GOTO 60340
61200 REM
61210 REM *** CURSOR RUNTER ***
61220 REM
61230 IF M > LEN (CC$) - 39 THEN CALL - 198: GOTO 60340
61240 CV = PEEK (37):M = M + 40: VTAB CV + 2: GOTO 60340
61300 REM
61310 REM *** EINFUEGE-ROUTINE ***
61320 REM
61330 IF M > LEN (CC$) THEN CALL - 198: GOTO 60340
61340 IF M = 1 THEN B$ = "":E$ = CC$: GOTO 61370
61350 B$ = LEFT$ (CC$,M - 1):E$ = RIGHT$ (CC$, LEN (CC$) -
      M + 1)
61360 REM *
61370 GET C$:C = ASC (C$)
61380 IF C < 32 OR C > 126 THEN GOTO 60430: REM EINFUEGEN
      ENDE
61390 REM *
61400 CC$ = B$ + C$ + E$: IF LEN (CC$) < = LE GOTO 61430
61410 CC$ = LEFT$ (CC$,LE): IF LEN (E$) = 1 THEN E$ = "":
      GOTO 61430
61420 E$ = LEFT$ (E$, LEN (E$) - 1)
61430 PRINT C$;:CH = PEEK (36):CV = PEEK (37): PRINT E$;
61440 POKE 36,CH: VTAB CV + 1:M = M + 1
61450 ON (M > LE) GOTO 60340: GOTO 61350
61500 REM
61510 REM *** LOESCH-ROUTINE ***
61520 REM
61530 IF M > LEN (CC$) THEN CALL - 198: GOTO 60340
61540 CH = PEEK (36):CV = PEEK (37)
61550 IF M = 1 AND LEN (CC$) = 1 THEN B$ = "":E$ = "":
      GOTO 61590
```

```
61560 IF M = 1 THEN B$ = "":E$ = RIGHT$ (CC$, LEN (CC$) -
      1): GOTO 61590
61570 IF M = LEN (CC$) THEN B$ = LEFT$ (CC$, LEN (CC$) -
      1):E$ = "": GOTO 61590
61580 B$ = LEFT$ (CC$,M - 1):E$ = RIGHT$ (CC$, LEN (CC$) -
      M)
61590 CC$ = B$ + E$: PRINT E$;" ";: POKE 36,CH: VTAB CV +
      1: GOTO 60340
61600 REM
61610 REM *** DELETE ***
61620 REM
61630 IF M = 1 THEN CALL - 198: GOTO 60340
61640 IF LEN (CC$) = 1 THEN B$ = "":E$ = "": GOTO 61680
61650 IF M = 2 THEN B$ = "":E$ = RIGHT$ (CC$, LEN (CC$) -
      1): GOTO 61680
61660 IF M > LEN (CC$) THEN B$ = LEFT$ (CC$, LEN (CC$) -
      1):E$ = "": GOTO 61680
61670 B$ = LEFT$ (CC$,M - 2):E$ = RIGHT$ (CC$, LEN (CC$) -
      M + 1)
61680 CC$ = B$ + E$: PRINT CHR$ (8);:CH = PEEK (36):CV =
      PEEK (37)
61690 PRINT E$;" ";: POKE 36,CH: VTAB CV + 1:M = M - 1:
      GOTO 60340
61700 REM
61710 REM *** EINGABE BIS CURSOR ***
61720 REM
61730 IF M < = LEN (CC$) THEN FOR N = 0 TO ( LEN (CC$) -
      M): PRINT " ";: NEXT
61740 IF M = 1 THEN CC$ = "": RETURN
61750 CC$ = LEFT$ (CC$,M - 1): RETURN
61800 REM
61810 REM *** CURSOR AN BEGINN ***
61820 REM
61830 HTAB HT: VTAB VT:M = 1: GOTO 60340
61900 REM
61910 REM *** CURSOR ANS ENDE ***
61920 REM
61930 VTAB VT: HTAB (HT + LEN (CC$)):M = LEN (CC$) + 1:
      GOTO 60340
```

Aktuelle Computerbücher

1985, 305 S., 6 Abb., kart., DM 58, – , ISBN 3-7785-1150-5
Begleitdiskette DM 48, – doppelseitig beschrieben
ISBN 3-7785-1290-9

Dieses Buch wendet sich als lehrbuchhafter Kurs an alle, die professionelle hochaufgelöste Grafiken auf dem Apple erzeugen wollen. Der erste Teil beginnt mit einem Abriß des Aufbaus der HGR-Seiten aus der Sicht des Programmierers. Danach wird das Programm HRCG (HI-RES Character Generator, Apple, Inc.) eingehend analysiert, und es werden sinnvolle Ergänzungen vorgestellt. Schrittweise wird die Nutzung des HRCG erarbeitet bis hin zur beliebigen Bewegung eines statistischen Objekts auf einer der HGR-Seiten.
Der zweite Teil baut auf dem ersten auf und führt über die Definition mehrerer Objekte und simultaner Bewegung hin zu einem Arcade-Spiel, das für die meisten kauflichen Action-Spiele in der meisterhaften Grafik als Vorbild dienen kann. Grundkenntnisse in 6502-Assembler sollten vorhanden sein.

1985, 470 S., kart., DM 68, –
ISBN 3-7785-1134-3

„Die ProDOS Analyse" ist die umfangreichste und detaillierteste Darstellung, die jemals ein Apple-Betriebssystem erfahren hat.
Wer die „Innereien" von ProDOS bis zum letzten Byte, z.T. bis ins letzte Bit kennenlernen möchte, braucht dieses Buch. Das komplette Betriebssystem (Urlader, MLI, Disk-Driver, RAM-Disk-Driver und Uhr-Routine) mit Ausnahme des BASIC-SYSTEM wird mit umfangreichen Kommentaren und Übersichtstabellen disassembliert.
Auch die nicht im „Technical Reference Manual" aufgeführten Eigenschaften von ProDOS werden analysiert und beschrieben, z.B. die vertrackten eingebauten Testroutinen zur Identifikation der verschiedenen Apple II Modelle und *eventueller Nachbaugeräte*.
Programmierer, die ProDOS versionsabhängig „patchen" möchten, erhalten hier den genauen Überblick, wo was geändert werden muß, damit dies keine negativen Konsequenzen hat.

Arne Schäpers, Das BASIC SYSTEM, 1984, ca. 300 S., kart., DM 54, – ISBN 3-7785-1407-5
Sie wollen die erweiterten Möglichkeiten von ProDOS und dem BASIC-SYSTEM voll ausnutzen? Nur hier finden Sie

- *das Lesebuch:* eine schrittweise Erklärung der Zusammenarbeit zwischen Applesoft und dem Monitor, des Aufbaus der Stringverwaltung, mögliche Umleitungen der Ein-/Ausgabe und der Eingriffsmöglichkeiten in diese Abläufe. Die gezeigten Mechanismen werden durch zahlreiche (und sehr kurze) Beispielprogramme untermauert;
- *die Analyse:* eine minutiöse Sezierung des BASIC-SYSTEM zusammen mit der Kommandoschnittstelle zu ProDOS. Die Hauptfunktionen (Vektorbehandlung, TRACE-Kontrolle, FRE-Kommando, Global Page und benutzte Speicherstellen) sind jeweils in eigenen Abschnitten erläutert, ein kommentiertes Listing schließt diesen Teil ab;
- *Tips & Tricks:* die verborgenen Möglichkeiten des BASIC-SYSTEM („Input Anything", andere Dateinamen als STARTUP, RESET ohne CLEAR), der Aufbau „externer Kommandos". Dieser Teil enthält ein komplettes Rahmenprogramm für die Erstellung eigener Kommandos sowie die Erweiterung MONITOR, mit der (analog zu DOS 3.3) die Ausgabe von Kommandos und der Dateiverkehr auf dem Bildschirm sichtbar gemacht werden können.

☐ Schäpers, **Bewegte Apple Grafik**, ISBN 3-7785-1150-5, DM 58, –

☐ Begleitdiskette, DM 48, – ISBN 3-7785-1290-9

☐ Schäpers, **Pro-DOS-Analyse** ISBN 3-7785-1134-3, DM 68, –

☐ Schäpers, **Das BASIC-SYSTEM,** ISBN 3-7785-1407-5, DM 54, –

BESTELLCOUPON

Gewünschte Bücher bitte ankreuzen und an Dr. Alfred Hüthig Verlag, Postfach 102869, 6900 Heidelberg, schicken.

Name _____

Straße _____

Ort _____

Datum _____ Unterschrift _____

Hüthig

PEEKER Börse

Biete Hardware

Apple IIe Speichererweiterung der 1 MB Karte von 256 K auf 1 MB DM 369,– Apple IIc 10er Block Adapter DM 99,–, Macintosh Speichererweiterung von 128 auf 512 KB DM 299,–
Macintosh Power Sound DM 149,–
XL-Enhancement-Kit DM 499,–
Fa. Reinhard Schlösser, Ismaninger Str. 108, 8000 München,
Tel. ab 18 Uhr 089 / 98 58 89

APPLE IIc + Monitor + 2. LW + Imagewriter + Maus + Joystick + Appleworks + Pascal + Applewriter + Spiele + Literatur. VB DM 3750,–. Tel. 0 23 32 / 8 25 02.

AP22 8MHz Z-80 Karte DM 450,–. AP 35 Grafikkarte 512*512 Punkte incl. AP22/35 Adapter DM 450,–. Tel. 02 08 / 75 14 66 ab 18.00 Uhr.

APPLE IIC + Z80-Karte mit CP/M 4.0, 2. LW, Monitor, viel Softw. u. Lit. VB DM 2000,–; Kyan-Pasc. + kix + Mous. Text DM 200,–; Peeker + Sammeldisk. kpl. DM 200,–. Turtle Graph. LIB (GEISS) DM 50,–. T. 02573/3212.

RAM-KARTEN
Rüsten Sie Ihren Apple II+ auf:
512KB ..... DM 383,–
768KB ..... DM 463,–
Die Karten rüsten direkt den Hauptspeicher auf und sind auch unter Basic voll nutzbar. Erhöhung der Anzahl der HGR-Seiten bis auf 32 Seiten.
Die Karten werden komplett aufgebaut und mit Software geliefert. Versand nur gegen VR-Scheck. Andreas Müller, Lessingstr. 17, 4000 Düsseldorf, T. 0211/776503

Orig. 128K IIe 2 LW + Contr. Mon. Joy. Lit. par. Druckerint. + Kabel div. Progr. + ca. 150 Disk. VB DM 3440,–.
Th. Niedermeier, T. 0 29 38 / 38 59.

Apple IIe 128K, Monitor II, 2 LW, 80Z, Z80, Par. Interf., Kyan 2.0, VB: DM 3000,–.
Tel. 0 64 09 / 564

ITOH 8510 FF Matrixdrucker Schnittst. par./ser. werksgepr. mit autom. EB-Einzug-Aufsatz NP 2300,– für DM 1100,–.
K. Freimann, Andelsbachstr. 2a, 7887 Laufenburg, 0 77 63 / 16 58 8-12 h.

IEE-488/Interface 300,–/80-Zeichenkarte 50,–/Num. Tastatur 150,–. Tel. 0 81 58 / 14 40.

Professionelle CAD-Anlage auf Apple IIe-Basis. NP über DM 60.000,– VB DM 26.000,–.
Tel. 0 71 43 / 9 18 36.

Apple IIe (Original), wie neu, mit orig. Monitor, 1 Disk. Laufwerk, 80Z-Karte, Joystick, Quickfile, Flugsim., Monitorständer DM 1800,–. beam-Verlag, Postf. 11 48, T. 0 64 21 / 6 36 02, 3550 Marburg. ©

Gewerbliche Anzeigen sind mit © gekennzeichnet.

Lightpen CT-100 f. Apple mit Software DM 169,–/Logikana-Lyzer ALA 0810 f. Apple od. IBM, 10 MHz, 8 Kanäle DM 2879,–.
Fa. ACT, Kaiserstr. 115, 6795 Kindsbach. Tel. 0 63 71 / 1 51 17 ©

Wir bieten Applied engineering: Ramworks 512K IIe 800,–, Z-RAM 999,–, Ramfactor 512K IIe/II+ 850,–, Transwarp 3,6 × schneller 825,–. Bitter Labortechnik, Große Bleiche 27, 3050 Steinhude, Tel. 0 50 33 / 15 22 ©

Orig. Apple IIe 128K 80 Zeichen DuoDisk Monitor IIe Z80A Z80B Joystick Mouse viel Software + Literatur Preis DM 3500,– VB. Tel. 0 26 33 / 9 61 47 nach 18 Uhr.

Orig. Apple II Graphics Tablet wie neu gegen Gebot.
H. Bartsch, Teichelweg 8,
7520 Bruchsal 7

Apple IIe m. vielen Extras, Peripherie, Literatur u. Softw. äußerst günstig zu haben (VB 3300,–). Keine Zeit verlieren!
Tel. 0 72 21 / 68 15 84. Ab 16 h. Es lohnt sich!

Verkaufe paral. Druckerinterf. o. Graf. Cntr. u. Telex-Interf. am. GP Preis VHB. Tel. 0 61 62 / 8 43 13 + 80 ZK.

Biete Software

Komfort. 6502-Debugger für Apple II. Testen von Masch. prog. mit allen erdenkl. Funkt. Auch für Einsteiger. DM 50,–.
Info: T. Buchali, Hanselmannstr. 2, 7100 Heilbronn.

Software Uhr für Apple II+, e, c, Zeitschaltmöglichkeit Diskette + Anleitung DM 25,– Oecking
Tel: Do. 02 31 / 39 19 20

PRINT, PLOT & SHOW IT! 225,–
Business Grafik mit allen Grafiken
REGRESSION XVII 225,–
Clusteranalyse 125,–
Plakat-Druck 225,–
Books & Software,
Tel. 0 95 71 / 31 82 ©

*** DISKETTEN ***
* 5 1/4", 48 tpi, DM 0,99, 2 D *
* 3 1/2", 135 tpi, DM 3,19, 1 DD *
* auch andere, bes. Garantie *
* Allg. Austro-AG, Ringstr. 10 *
* D-8057 Eching, *
* Tel. 0 81 33 / 61 16 *
©

!!! MAGNAT Das beste Apple Pascal-Disk-Bearbeitungsprogramm !!! Disk+Manual: 34,90 (ja!) Info gegen 50-Pf.-Rückumschlag. Jetzt zugreifen!!
Röhl, Horner Str. 9, 2800 Bremen

IIe+c: CAD paint: exakte Zeichnungen m. Beschriftung (2 Stärken), 40 Funktionen, 80 fertige Symbole für Wohnungspläne + elektr. Schalt. Info DM 2,–. Briefm., Disk: DM 99,– Scheck. Lohmann, M.-Müller-Ring 7, 6500 Mainz

Hallo 'TIME II' Besitzer: Werfen Sie Ihre Uhrenkarte nicht weg! IRQ-Software für DOS und ProDOS. Disk + Anleitung DM 25,–. Best. + Info b. Postfach 13 23, 2400 Lübeck.

*** Ausschneiden! Aufbewahren! ***
Applesoft – MSDOS-Transfer 1. Prg. DM 80,–, ab 2. DM 50,– macht: K. Freimann, Andelsbachstr. 2a, 7887 Laufenburg.

Apple II: DFÜ-Kermit, Pascal satt, Public Domain in DOS u. CP/M. Je Volume DM 15,–. Bahnhofssimulation, Sprachen (S.A.L.), Schulprogr. Gratisinfo: Fa. Waltraud Muhle, Waldwinkel 3, 2105 Seevetal 3.

Suche Software

ProDOS-Treiber für SATURN 128K
RAM-DISK J. Flacher, Kornweg 2, CH-8405 Winterthur, Tel. 0 14 35 / 43 16 od. 052 / 28 23 92

Suche Privatliquidation Arztp. für Apple IIe, 128 K 2. Laufw. Kopie oder Original.
Tel. 0 56 01 / 8 67 47

Verschiedenes

Verkaufe Peeker 1/84-12/86 + Sammeldisk Nr. 1-19 + Turtle Graphics-Library + Softbreaker VB DM 250,–. Tel. 0 26 33 / 9 61 47 nach 18 Uhr.

Wer hilft? Wie können Textfiles schneller bearbeitet werden? – (Apple II+, 2 × 128K-Saturn als Pseudo-Disk, Erphi-Contr. mit 2 × (2 × 80 Track), Div-DOS oder evtl. gepachtes DOS?).
Steffen Neugebauer, Schönwaldstr. 19, 8721 Hesselbach.

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

Aus Vorführbestand haben wir folgende sehr gut erhaltene Komponenten anzubieten:

	Preis je Stück
1 APPLE-EUROPLUS mit – 8"-Controller – 16 KB RAM-Karte – Drucker-Interface – zweite 16 KB Erweiterung	DM 456,—
1 APPLE-EUROPLUS mit – 8" Controller – 16 KB RAM-Karte	DM 342,—
2 Grafiktableaus für APPLE-EUROPLUS mit Interface und Touchpen	DM 570,—
1 PHILIPS-Monitor mit Kabel für APPLE-EUROPLUS	DM 114,—
1 WATANABE 6 Farben-Plotter 1000	DM 912,—
1 PROFILE-Interface	DM 57,—
1 5-MB-Profile mit Verbindungskabel	DM 570,—
1 10-MB-Profile mit Verbindungskabel	DM 1710,—
1 BINDER-Drucker 8510 A	DM 228,—
2 APPLE-Imagewriter 12"	DM 399,—
3 8" Laufwerke Serie 1553 mit Verbindungskabel	DM 570,—
1 Handbuch APPLE IIe	DM 11,40
1 Handbuch APPLE II	DM 11,40
1 Handbuch GRAPHIC TABLET	DM 5,70
1 Handbuch APPLE PASCAL	DM 5,70

Alle Preise verstehen sich incl. Mehrwertsteuer. Zwischenverkauf vorbehalten, Lieferung erfolgt auf Nachnahme.
INNOVATION GmbH
Scheffelstraße 16, 7265 Neubulach
Telefon 0 70 53 / 62 68 ©

Peeker 1/84-5/86, Apple IIe Reference Manual, Beneath ProDOS, CP/M u. a. abzugeben!
Tel. 030 / 8 25 43 95

Suche Install. Anweisung für Apple Scribe Printer.
Tel. 0 71 44 / 3 66 57

APPLE II Motherboard-Reparaturen. Pauschal DM 100,–.
Tel. 05 11 / 40 66 04.

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 GmbH, Postfach 10 28 69, 6900 Heidelberg 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

PEEKER

Abo-Karte

Ja, ich möchte **Peeker** abonnieren.
Liefern Sie mir **Peeker** ab Ausgabe ..... zum Jahresbezugspreis von z. Zt. DM 75,– (inkl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt z. Zt. DM 75,– plus DM 20,– Versandspesen.

X _____
Datum 1. Unterschrift

Bitte lesen!

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

X _____
Datum 2. Unterschrift

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

Wir können nur Bestellungen mit zwei Unterschriften bearbeiten.

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____
_____
_____
_____
_____

am: _____

bei:

Peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER

Buch-Karte

Bitte senden Sie mir gegen Rechnung folgende Bücher:

- ☐ Bühler, Applesoft-BASIC, 3-7785-1094-0, DM 38,–
- ☐ Eggerich, dBase II, Bd. 1, 3-7785-1147-5, DM 39,80
- ☐ Eggerich, dBase II, Bd. 2, 3-7785-0987-X, DM 39,80
- ☐ Eggerich, dBase II, Bd. 3, 3-7785-0988-8, DM 39,80
- ☐ Gabriel, Applewriter, 3-7785-1234-X, DM 35,–
- ☐ Hagenmüller, Microsoft-BASIC, Bd. 1, 3-7785-1038-X, DM 38,–
- ☐ Juhnke/Redlin, Apple Pascal, Bd. 1, 3-7785-1246-3, ca. DM 40,–
- ☐ Kehrel, Apple Assembler lernen, Bd. 1, 3-7785-1151-3, DM 38,–
- ☐ Kehrel, Apple Assembler lernen, Bd. 2, 3-7785-1170-X, DM 38,–
- ☐ Schäpers, ProDOS Analyse, 3-7785-1134-3, DM 68,–
- ☐ Schäpers, Bewegte Apple-Graphik, 3-7785-1150-5, DM 58,–
- ☐ Stiehl, Apple DOS 3.3, 3-7785-1297-8, DM 28,–
- ☐ Stiehl, Apple ProDOS, Bd. 1, 3-7785-1098-3, DM 28,–
- ☐ Stiehl, Apple ProDOS, Bd. 2, 3-7785-1036-3, DM 30,–
- ☐ Stiehl, Apple Assembler, 3-7785-1047-9, DM 34,–
- ☐ Wassermann, Apple IIc Handbuch, 3-7785-1157-2, DM 35,–

_____
Datum Unterschrift

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____
_____
_____
_____
_____

am: _____

bei:

Peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER

Software-Karte

Bitte senden Sie mir gegen Rechnung folgende Disketten:

- ☐ 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, Begleitdisk., DM 28,–
- ☐ ProDOS, Band 1, Begleitdisk., DM 28,–
- ☐ ProDOS, Band 2, Begleitdisk., DM 28,–
- ☐ Apple Assembler, Begleitdisk., DM 28,–
- ☐ ProDOS-Editor 1.0, Programm, DM 98,–
- ☐ MMU 2.0, Programm, DM 98,–
- ☐ INPUT 2.0, Programm, DM 98,–
- ☐ DB-Meister, Programm, DM 290,–
- ☐ Superquick, Programm, DM 48,–
- ☐ Turtle Graphics, Programm, DM 98,–
- ☐ Disk 40, Programm, DM 48,–
- ☐ Kyan-Pascal 2.0, Programm, DM 170,–
- ☐ Fast-Writer, DOS 3.3, DM 128,–
- ☐ Fast-Writer, ProDOS, DM 128,–
- ☐ Double-Hires-Tools für Applesoft, DM 28,–
- ☐ Double-Hires-Tools für Kyan, DM 28,–
- ☐ Kyan-Toolkit Nr._____, DM_____

_____
Datum Unterschrift

Abo-Karte

Name

Firma

Straße

PLZ/Ort

Ich wünsche jährliche Berechnung durch:
☐ Verlagsrechnung ☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank/PschA

Bankleitzahl

Kto.-Nr.

Bitte freimachen

POSTKARTE

Peeker
Leserservice
Dr. Alfred Hüthig Verlag GmbH
Postfach 10 28 69
6900 Heidelberg

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

Buch-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Bitte freimachen

POSTKARTE

Peeker
Buchabteilung
Dr. Alfred Hüthig Verlag
Postfach 10 28 69
6900 Heidelberg 1

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

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Bitte freimachen

POSTKARTE

Peeker
Softwareabteilung
Dr. Alfred Hüthig Verlag
Postfach 10 28 69
6900 Heidelberg 1

DISK 40

Disketten-Organisationsprogramm für DOS-3.3-35-40 Spuren

von Hermann Seibold und Dipl.-Ing. Udo Marin, 1986, Programmdiskette mit Anleitung, DM 48,–

Durch eine einfach zu bedienende Menüführung können DOS-3.3-Disketten umfangreich bearbeitet oder kopiert werden.
– Tabellarische Ausgabe der Diskettenbelegung
– Ordnen des Catalogs
– „Undelete"n von versehentlich gelöschten Dateien
– Vergleichen von Disketten, Dateien oder DOS-Spuren
– Kopieren von Disketten, Dateien oder DOS-Spuren
– Formatieren von Daten-Disketten
– Erweitern auf 40 Spuren bei bestehenden 35-Spur-Disketten
– Ändern des Boot-Programms
– File-Editor zum Editieren von Disketten-Dateien
– Komfortabler Sektor-Editor für Hex- und ASCII-Darstellung
–VTOC-Editor, z. B. zur Freigabe der DOS-Spuren

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

Multiplikationsfehler in der Applesoft-FMULT-Routine

von Hans-Martin Eng

1. Analyse des FMULT-Bugs

Ein kleiner, aber gemeiner Fehler ist den Entwicklern des Applesoft-Interpreters (Firmen Microsoft und Apple) in der Floating-Point-Multiplikationsroutine (FMULT, Einsprung bei $E97F) unterlaufen. Glen Bredon zeigt in seinem Applesoft-Listing praktisch mit dem Finger auf die fehlerhafte Stelle. Exakt in $E9B2 fehlt ein einziger „SEC". Dies führt in bestimmten Fällen letztlich zu Fehlern in den letzten 8 Bits der Mantisse des Multiplikationsprodukts.

Da alle Funktionen über die FMULT-Routine berechnet werden, entstehen unweigerlich weitere Fehler. Sehen Sie sich unter diesem Aspekt einmal **Abb. 1** an. Das ist eine ganz normale Exponentialfunktion!

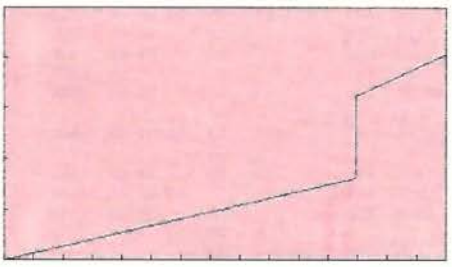


Abb. 1: Fehlerhafte Exponentialfunktion

Sie glauben mir nicht? Dann prüfen Sie es selbst nach. Starten Sie Plot.2.0 oder Plot.3.E und lassen Sie die Funktion „Y=EXP(X+1)-E" im Darstellungsbereich x ε [0 .. 1.5E-7], y ε [0 .. 5E-7] plotten.

Wir erinnern uns einmal kurz an unsere Schulzeit. Dort lernten wir, daß die Exponentialfunktion im gesamten reellen Zahlenbereich unendlich oft stetig differenzierbar ist. Was in Abb. 1 zu sehen ist, ist jedoch im „Knickpunkt" alles andere als differenzierbar, geschweige denn stetig differenzierbar. Um Sie zu beruhigen, kann ich Ihnen versichern, daß für alle ganzzahligen Argumente solch ein Knickpunkt zu beobachten ist.

Genauso übel macht sich der Fehler in der FMULT-Routine bei den trigonometrischen Funktionen bemerkbar. Läuft das Plotprogramm noch? Dann geben Sie als neue Funktion „Y=COS(X)-1.00000001" ein und lassen sie im Bereich x ε [-3E-7 .. 3E-7], y ε [-8E-8 .. 1E-8] plotten oder sehen Sie sich **Abb. 2** an. Das tut schon beinahe weh!

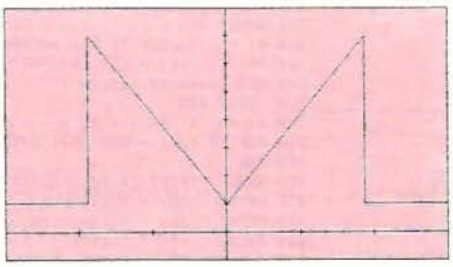


Abb. 2: Fehlerhafte Cosinusfunktion

Den Fehler kann man auch direkt an Zahlenbeispielen nachweisen. Zuerst prüfen wir einmal den Bug in FMULT selbst. Tippen Sie „PRINT 1*988244415" und überprüfen Sie das Ergebnis: „988244384"!. Genauso unschön ist „PRINT 1*10.0000009" – das „ergibt" nämlich „10.0000005". Diese beiden Zahlenbeispiele stammen von G. Bredon.

Läßt man die Cosinus- oder Exponentialfunktion im kritischen Bereich tabellieren, sind die Fehler überraschenderweise nicht allzu groß, aber doch beachtlich.

Tabelle 1 zeigt die Cosinusfunktion: linke Spalte X, mittlere Spalte richtiger(er) Wert, rechte Spalte „Applewert".

In **Tabelle 2** schließlich wurde dasselbe für die Exponentialfunktion durchgespielt. Sie können diese Tabelle auf Ihrem Rechner erstellen, indem Sie das Programm „FMULT.DEMO" starten. Doch darauf kommen wir später zurück.

Tabelle 1: Cosinusfunktion

X	COS(X) mit Patch	COS(X) ohne Patch
-1.88 E-07	1.000000000	1.000000000
-1.87 E-07	1.000000000	0.999999940
-1.86 E-07	1.000000000	0.999999941
-1.85 E-07	1.000000000	0.999999941
-1.84 E-07	1.000000000	0.999999941
-1.83 E-07	1.000000000	0.999999942
-1.82 E-07	1.000000000	0.999999942
-1.81 E-07	1.000000000	0.999999942
-1.80 E-07	1.000000000	0.999999943
-1.79 E-07	1.000000000	0.999999943
-1.78 E-07	1.000000000	0.999999943
-1.77 E-07	1.000000000	0.999999944
-1.76 E-07	1.000000000	0.999999944
-1.75 E-07	1.000000000	0.999999944
-1.74 E-07	1.000000000	0.999999945
-1.73 E-07	1.000000000	0.999999945
-1.72 E-07	1.000000000	0.999999945
-1.71 E-07	1.000000000	0.999999946
-1.70 E-07	1.000000000	0.999999946
-1.69 E-07	1.000000000	0.999999946
-1.68 E-07	1.000000000	0.999999946

Tabelle 2: Exponentialfunktion

X	EXP(X) mit Patch	EXP(X) ohne Patch	
1.00000000	2.71828183	2.71828183	
1.00000001	2.71828186	2.71828184	
1.00000002	2.71828188	2.71828186	
1.00000003	2.71828191	2.71828187	
1.00000004	2.71828194	2.71828188	
1.00000005	2.71828196	2.71828189	
1.00000006	2.71828199	2.71828191	
1.00000007	2.71828202	2.71828192	
1.00000008	2.71828204	2.71828194	
1.00000009	2.71828207	2.71828195	
1.0000001Ø	2.71828210	2.71828196	
1.00000011	2.71828212	2.71828198	
1.00000012	2.71828215	2.71828199	<-
1.00000013	2.71828217	2.71828217	
1.00000014	2.71828220	2.71828220	
1.00000015	2.71828223	2.71828223	

2. Korrektur des FMULT-Bugs

Ein Bug dieser Art ist immer ärgerlich. Da er sich im ROM befindet, kann man nicht einfach hineinpatchen, wie das bei RAM-residenten Programmen prinzipiell möglich ist. Doch zumindest unter DOS 3.3 kann man auf die Language-Card ausweichen und dort den Fehler ausbügeln. Wo der „SEC"-Befehl fehlt, muß man ein „JMP PATCH" einfügen, bei PATCH dann den „SEC"-Befehl nachholen und mit einem weiteren „JMP"-Befehl die Ausführung fortsetzen. Da glücklicherweise in $E9B2 ein JMP-Befehl steht, benötigen wir nur noch vier zusätzliche Bytes. Hier sind drei geeignete Stellen:

1. $F094 .. $F09D: Dort ist die Firma Microsoft mit dem String „MICROSOFT!" verewigt (allerdings sind die Bits 7 und 6 jeweils invertiert). Der Platz ist mehr als ausreichend.
2. $F12F ff.: In der BASIC-Kaltstartroutine werden die Zero-Page-Adressen $00 .. $05 zweimal innerhalb kürzester Zeit mit Werten belegt. Auf das erste Mal kann man verzichten. Dann werden weitere 11 Bytes frei.
3. $FC5D .. $FC61 (nur enhanced Apple IIe!): Dort stehen fünf NOPs. Auch dort kann man hineinpatchen.

Die Vorgehensweise ist klar. Kopieren Sie einfach den ROM-Inhalt in die Language-Card, patchen Sie die gewünschten Stellen und schalten Sie dann auf die LC um. Schon ist der Fehler ausgemerzt.
Diese Arbeit nimmt Ihnen das folgende Assemblerprogramm ab: Starten Sie es einfach mit „BRUN FMULT.PATCH". Als Beweis, daß nun alles stimmt, sehen Sie sich **Abb. 3** und **Abb. 4** an. Das Programm FMULT.DEMO macht nichts anderes, als abwechselnd den gleichen Funktionswert von der Firmware und der LC-Software ausrechnen zu lassen.

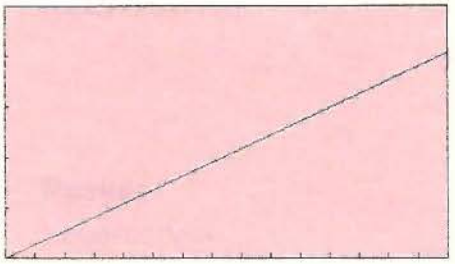


Abb. 3: Korrekte Exponentialfunktion

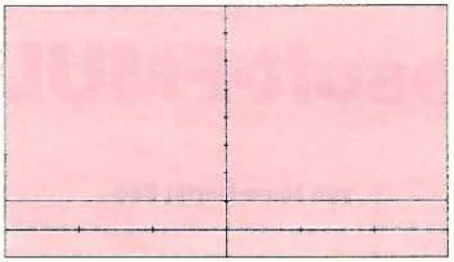


Abb. 4: Korrekte Cosinusfunktion

Besonders mutige Besitzer eines Apple IIe können sich nun den gepatchten Bereich $E000 bis $FFFF in ein 2764-EPROM brennen (lassen) und gegen das EF-ROM auf dem Motherboard austauschen. Apple II+-Besitzer müssen überprüfen, ob sie anstelle der maskenprogrammierten ROMs auch EPROMs vom Typ 2716 in die ROM-Sockel stecken können. Sie müssen dann allerdings zwei EPROMs programmieren.

Kurzhinweise

1. Zweck:
Beheben des Bugs in FMULT
2. Konfiguration:
Apple IIe/c oder II+ mit LC, DOS 3.3 (!)
3. Test:
zunächst BRUN FMULT.PATCH
dann RUN FMULT.DEMO
(Die auf der Sammeldisk enthaltene FMULT.DEMO-Version ist dahingehend modifiziert, daß sie auch direkt gestartet werden kann.)
5. Sammeldisk:
FMULT.PATCH
T.FMULT.PATCH
(Big-Mac-Quellcode)
BUG.DEMO
(Applesoft-BASIC-Programm)

FMULT.DEMO

```
100 REM Demo zu den beiden Bugs
120 LCIN% = - 16256:ROMIN% = - 16255:D$ = CHR$ (4)
130 HOME : PRINT "Bitte wählen:": PRINT : PRINT "1 - EXP-Bug"
140 PRINT : PRINT "2 - COS-Bug"
150 GET A$: ON VAL (A$) GOTO 300,350
160 GOTO 150
170 END
190 REM Drucke Wertetafel
200 FOR X = X1 TO X2 STEP XS
210 PRINT X; TAB( 14);
220 Z = PEEK (LCIN%): PRINT FN Y(X); TAB( 27);
230 Z = PEEK (ROMIN%): PRINT FN Y(X)
240 NEXT : PRINT : PRINT D$"CLOSE": PRINT D$"PR#0"
250 PRINT "Bitte beliebige Taste betätigen!";: GET A$
260 HOME : PRINT "Nochmal? ";
270 GET A$: IF A$ = "J" THEN 110
280 END
290 REM Exponentialfunktion
300 GOSUB 400
310 X1 = 1.000000:X2 = 1.00000015:XS = 1E - 8
320 DEF FN Y(X) = EXP (X): GOTO 200
340 REM Cosinusfunktion
350 GOSUB 400
360 X1 = - 1.88E - 7:X2 = - 1.68E - 7:XS = 1E - 9
370 DEF FN Y(X) = COS (X): GOTO 200
390 REM
400 HOME : PRINT "Ausgabe auf"
410 PRINT " <S> - Slot,"
420 PRINT " <B> - Bildschirm oder"
430 PRINT " <T> - Textfile. Bitte wählen!"
440 GET A$: IF A$ = "S" OR A$ = "s" THEN 550
450 IF A$ = "T" OR A$ = "t" THEN 500
460 IF A$ < > "B" AND A$ < > "b" THEN 440
470 RETURN
490 REM Eingabe Textfile
500 PRINT : INPUT "Filename: ";F$
510 IF F$ = "" THEN F$ = "TABELLE"
520 PRINT : PRINT D$"OPEN"F$: PRINT D$"WRITE"F$: RETURN
540 REM Eingabe Slotnummer
550 PRINT : PRINT "Slot-Nummer (1-7): ";
560 GET F$: IF F$ < "1" AND F$ > "7" THEN 560
570 PRINT : PRINT D$"PR#"F$: RETURN
```

FMULT.PATCH

```
BSAVE FMULT.PATCH, A$3Ø0, L$67

                  1   ********************************
                  2   *                              *
                  3   *  Patchvorschlag, um Bug in   *
                  4   * FMULT-Routine zu beseitigen  *
                  5   *     von Hans-Martin Eng      *
                  6   *      18. August 1986         *
                  7   *                              *
                  8   ********************************
                  9   *
                  1Ø            ORG  $3Ø0
                  11  *
                  12  IND       EQU  $5Ø
                  13  *
                  14  DOSWARM   EQU  $Ø3DØ
                  15  READBK2   EQU  $CØ8Ø
                  16  ROMIN     EQU  $CØ81
                  17  ROMSTART  EQU  $DØØØ
                  18  SHFTRES   EQU  $E8DA
                  19  BUG       EQU  $E9B2
                  2Ø  MICROSFT  EQU  $FØ94
                  21  COUT      EQU  $FDED
                  22  *
                  23  * Kopiert ROM-Bereich $DØØØ..FFFF in
                  24  * LC-Bank2
                  25  *
Ø3Ø0: AD 81 CØ    26  START     LDA  ROMIN       ; Read ROM
Ø3Ø3: AD 81 CØ    27            LDA  ROMIN       ; Write Bank2
Ø3Ø6: AØ ØØ       28            LDY  #<ROMSTART
Ø3Ø8: A9 DØ       29            LDA  #>ROMSTART
Ø3ØA: 84 5Ø       3Ø            STY  IND
Ø3ØC: 85 51       31            STA  IND+1
                  32  *
Ø3ØE: B1 5Ø       33  LOOP      LDA  (IND),Y
Ø31Ø: 91 5Ø       34            STA  (IND),Y
Ø312: C8          35            INY
Ø313: DØ F9       36            BNE  LOOP
Ø315: E6 51       37            INC  IND+1
Ø317: DØ F5       38            BNE  LOOP
                  39  *
Ø319: AØ Ø3       4Ø  PATCH     LDY  #3
Ø31B: B9 5F Ø3    41  PTLOOP1   LDA  PATCH1,Y
Ø31E: 99 B1 E9    42            STA  BUG-1,Y
Ø321: 88          43            DEY
Ø322: DØ F7       44            BNE  PTLOOP1
                  45  *
Ø324: AØ Ø4       46            LDY  #4
Ø326: B9 62 Ø3    47  PTLOOP2   LDA  PATCH2,Y
Ø329: 99 93 FØ    48            STA  MICROSFT-1,Y
Ø32C: 88          49            DEY
Ø32D: DØ F7       5Ø            BNE  PTLOOP2
                  51  *
Ø32F: AD 8Ø CØ    52            LDA  READBK2     ; LC ein
                  53  *
Ø332: B9 4Ø Ø3    54  PRINTMSG  LDA  MSG,Y
Ø335: FØ Ø6       55            BEQ  EXIT
Ø337: 2Ø ED FD    56            JSR  COUT
Ø33A: C8          57            INY
Ø33B: DØ F5       58            BNE  PRINTMSG
                  59  *
Ø33D: 4C DØ Ø3     6Ø  EXIT      JMP  DOSWARM
                  61  *
Ø34Ø: 8D 87       62  MSG       HEX  8D87
Ø342: C6 CD D5    63            ASC  "FMULT-Patch ist installiert!"
Ø35E: 8D ØØ       64            HEX  8DØØ
                  65  *
                  66  PATCH1    EQU  *-1
                  67  PATCH2    EQU  PATCH1+3
                  68  *
                  69            ORG  BUG
E9B2: 4C 94 FØ    7Ø            JMP  MICROSFT    ; Vorher SHFTRES
                  71  *
                  72            ORG  MICROSFT
                  73  * vorher ASC "MICROSOFT!" mit invertierten
                  74  * Bits 7 und 6.
F094: 38          75            SEC
F095: 4C DA E8    76            JMP  SHFTRES
```

Aufspalten von UCSD-Quelltexten

von Jürgen Geiß

Die nachfolgende kleine Utility dient zum Aufspalten von Quelltexten, die unter Apple Pascal 1.2 oder mit einem anderen Editor erstellt wurden, für den Apple-Pascal-1.1-System-Editor, der keine Fremddateien annimmt, die mehr als ca. 3Ø Blöcke umfassen.

```
[$I-]
program Aufspalten (input, output);
{Aufspalten von langen Textdateien in solche, die vom
 "normalen" SYSTEM.EDITOR eingelesen werden koennen}
var Source   : file;
    Dest     : file;
    Srcname  : string;
    Destname : string;
    Helpname : string;
    Blocks   : integer;
    I        : integer;
    Header   : packed array [Ø..1Ø23] of Ø..255;
    Buffer   : packed array [Ø..16383] of Ø..255;
begin {main}
  write ('Quelldatei: ');
  readln (Srcname);
  Srcname := concat (Srcname, '.TEXT');
  write ('Zieldatei : ');
  readln (Destname);
  reset (Source, Srcname);
  if IOresult = Ø then
  begin
    Helpname := '1';
    I := blockread (Source, Header, 2);
    repeat
      Blocks := blockread (Source, Buffer, 32);
      if Blocks > Ø then
      begin
        writeln (concat (Destname, Helpname, '.TEXT'));
        rewrite (Dest, concat (Destname, Helpname, '.TEXT'));
        I := blockwrite (Dest, Header, 2);
        I := blockwrite (Dest, Buffer, Blocks);
        close (Dest, lock);
        Helpname [1] := succ (Helpname [1]);
      end {if}
    until Blocks = Ø
  end; {if}
  close (Source)
end. {main}
```

Referenzliste

Zeilenverweise in Applesoft-Programmen

von Ludger T. Engbert

Bei der Entwicklung umfangreicher Programme sowie der Weiterentwicklung fremder Programme ist es nötig, jederzeit zu wissen, ob und wo bestimmte Subroutinen aufgerufen werden und ob alle Referenzen (Zeilenverweise) erfüllt sind. Hierzu dient die Referenzliste, die mit einem Line-Cross-Referencer erzeugt wird.
Das hier vorgestellte Programm LINE.REFS druckt in numerischer Reihenfolge alle Zeilennummern aus, auf die im Programm durch GOTO, GOSUB und THEN verwiesen wird. Dabei werden auch Aufzählungen (z.B. ON I GOSUB 100,200,300) berücksichtigt. Die Nummern der Zeilen, in denen der Verweis geschieht, werden im Anschluß an die aufgerufene Zeilennummer sortiert ausgedruckt. „Undefined Statements" werden durch ein vorangestelltes „?" gekennzeichnet.

Programmerläuterungen

Beim ersten Aufruf durch „BRUN LINE.REFS" wird der Ampersand-Vektor installiert und HIMEM gesetzt. Die Routine kann dann immer wieder durch „&" aufgerufen werden. Man beachte, daß LINE.REFS im Speicher ab $9000 liegt und deshalb beispielsweise mit dem MACROEDITOR kollidieren würde.

Beim eigentlichen Aufruf mit „&" wird zunächst (Routine REFS) der aktuelle Textpointer (TXTPTR) gesichert. Während des Programmablaufs wird der Textpointer benötigt, um nacheinander auf alle Bytes des analysierten Programms zu zeigen. Der Textpointer wird dann vor Beendung des Programms wieder hergestellt und zeigt dann auf das Byte, das dem Programmaufruf-& folgt.

Einer Applesoft-Programmzeile sind immer 4 Bytes vorangestellt: Die ersten beiden Bytes („Links") zeigen auf die nachfolgende Programmzeile, die nächsten beiden Bytes beinhalten die Zeilennummer (Low Byte zuerst).
Sodann wird in der Routine REFS die Variable HIGH initialisiert. HIGH zeigt jetzt 4 Bytes über das Programmende hinaus. Die 4 Bytes zwischen Programmende und HIGH werden nun mit $FF gefüllt.
In der nachfolgenden NXTLIN-Routine werden für jede Referenz vier Bytes zwischen das Programmende und die 4 $FF-Bytes eingefügt. Diese 4 Bytes beinhalten die Zeilennummer, auf die verwiesen wird, und die Zeilennummer, die verweist (High Byte zuerst). Z.B. würden für die Zeile „100 IF K THEN 200" die Zahlen 200 und 100 in die Tabelle eingetragen werden. Jede nachfolgende Referenz wird gezielt eingefügt, so daß bei Erreichen des BASIC-Programmendes bereits eine sortierte Tabelle vorliegt.
Die Routine NXTLIN prüft zunächst, ob das BASIC-Programmende erreicht ist. Wenn ja, wird nach PRINT gesprungen, um die Tabelle auszudrucken. Wenn nein, wird die derzeitige Zeilennummer nach CURLIN (Current Line) übertragen.
Die Routine NXTBYT (Next Byte) testet das Zeichen, auf das der Textpointer zeigt. Handelt es sich um eine Null, so ist das Zeilenende erreicht, und es wird nach NXTLIN gesprungen. Handelt es sich um ein GOTO, GOSUB oder THEN, gefolgt von einer Ziffer, so wird in der Routine LINE das HIGH um 4 erhöht (HIGH zeigt immer auf Tabellenende), die aufgerufene Zeilennummer nach LINNUM übertragen sowie HIGHTR = HIGH und HIGHDS = HIGH + 4 gesetzt. Beim Aufruf der Routine MOVE zeigt POSITION auf die Stelle in der Tabelle, wo das neue Zeilenpaar eingefügt werden muß. Die Routine BLTU2 (Block Transfer up) verschiebt den Speicherbereich zwischen LOWTR und HIGHTR „nach oben" bis nach HIGHDS. Das somit duplizierte Zeilenpaar wird jetzt mit dem neuen Zeilenpaar (in LINNUM und CURLIN) überschrieben. Folgt auf eine Referenz ein Komma, so wird nach LINE gesprungen, um das nächste Zeilenpaar zu erfassen. Folgt kein Komma, wird auf EOL (End of Line), d.h. auf Null getestet und nach NXTBYT1 gesprungen, um das nächste Byte zu verarbeiten

Die Routine PRINT druckt die sortierte Tabelle aus. Da alle Verweise auf eine Zeile sortiert vorliegen, braucht lediglich überprüft zu werden, ob aufeinanderfolgende Zeilenpaare eine Referenz auf dieselbe Zeilennummer enthalten. Ist dies der Fall, wird die aufgerufene Zeilennummer nur einmal ausgedruckt. Um bei Ausgabe der ersten Zeilennummer das mögliche Mißverständnis zu vermeiden, daß die fragliche Zeilennummer (in LINNUM) bereits ausgedruckt wurde, wird eingangs $FFFF nach LINNUM übertragen. PRINT1 überprüft, ob das Tabellenende erreicht ist, gekennzeichnet durch 2 aufeinanderfolgende $FF.

Die hier nicht beschriebenen Applesoft-Routinen wurden bereits in dem Artikel „Referenztest" (Peeker, 11/1985) erklärt, das mit dem ähnlichen Programm REF.TEST nur sog. Blindverweise ermittelt.

Kurzhinweise

1. Zweck:
Auflistung aller Zeilenverweise in Applesoft-Programmen zur systematischen Programmentwicklung und Programm-Dokumentation
2. Konfiguration:
Apple II+/e/c; nur DOS 3.3 (bei ProDOS könnte es Probleme wegen der HIMEM-Änderung geben)
3. Aufruf:
RUN LINE.REFS.DEMO
Sonst BRUN LINE.REFS, dann eigenes Applesoft-Programm laden:
&
4. Sammeldisk:
LINE.REFS.DEMO
LINE.REFS (Objektcode)
T.LINE.REFS (Big-Mac-Quelltext)

LINE.REFS.DEMO

```
 1Ø REM LINE.REFS.DEMO
 11 :
 2Ø TEXT : HOME : LIST 1ØØ,2ØØ
1ØØ PRINT  CHR$ (4)"BRUN LINE.REFS"
11Ø &
12Ø END
13Ø :
15Ø ONERR  GOTO 2ØØ
16Ø ERR OR
17Ø ON K GOTO 12Ø,13Ø,14Ø
2ØØ PRINT "NEVER MIND"
```

T.LINE.REFS

BSAVE LINE.REFS, A$9ØØØ, L35Ø
(Ab $9ØØØ eingeben. Adressen wurden aus Platzgründen nicht abgedruckt.)

```
                1               ORG  $9ØØØ
                2   *
                3   * LINE.REFS von L.Engbert
                4   *
                5   CH          EQU  $24
                6   LINNUM      EQU  $5Ø
                7   TXTTAB      EQU  $67
                8   HIMEM       EQU  $73
                9   CURLIN      EQU  $75
               1Ø   HIGH        EQU  $7D
               11   HIGHDS      EQU  $94
               12   HIGHTR      EQU  $96
               13   LOWTR       EQU  $9B
               14   PRGEND      EQU  $AF
               15   CHRTST      EQU  $BA
               16   TXTPTR      EQU  $B8
               17   POSITION    EQU  $FE
               18   AMPER       EQU  $3F5
               19   BLTU2       EQU  $D39A
               2Ø   FNDLIN      EQU  $D61A
               21   LINGET      EQU  $DA0C
               22   CRDO        EQU  $DAFB
               23   OUTQUES     EQU  $DB5A
               24   OUTSP       EQU  $DB57
ab $9ØØØ      25   LINPRT      EQU  $ED24
A9 4C          26               LDA  #$4C
8D F5 Ø3       27               STA  AMPER
A9 14          28               LDA  #<REFS
8D F6 Ø3       29               STA  AMPER+1
85 73          3Ø               STA  HIMEM
A9 9Ø          31               LDA  #>REFS
8D F7 Ø3       32               STA  AMPER+2
85 74          33               STA  HIMEM+1
6Ø             34               RTS
A5 B8          35   REFS        LDA  TXTPTR
48             36               PHA
A5 B9          37               LDA  TXTPTR+1
48             38               PHA
A5 67          39               LDA  TXTTAB
85 B8          4Ø               STA  TXTPTR
A5 68          41               LDA  TXTTAB+1
85 B9          42               STA  TXTPTR+1
18             43               CLC
A5 AF          44               LDA  PRGEND
69 Ø4          45               ADC  #$Ø4
85 7D          46               STA  HIGH
A5 BØ          47               LDA  PRGEND+1
69 ØØ          48               ADC  #$ØØ
85 7E          49               STA  HIGH+1
AØ Ø3          5Ø               LDY  #$Ø3
A9 FF          51               LDA  #$FF
91 AF          52   REFS1       STA  (PRGEND),Y
88             53               DEY
1Ø FB          54               BPL  REFS1
3Ø Ø3          55               BMI  NXTLIN1
2Ø 2Ø 91       56   NXTLIN      JSR  BYTGET
2Ø 2Ø 91       57   NXTLIN1     JSR  BYTGET
FØ 7B          58               BEQ  PRINT
2Ø 2Ø 91       59               JSR  BYTGET
85 75          6Ø               STA  CURLIN
2Ø 2Ø 91       61               JSR  BYTGET
85 76          62               STA  CURLIN+1
2Ø 2Ø 91       63   NXTBYT      JSR  BYTGET
FØ E9          64   NXTBYT1     BEQ  NXTLIN
C9 AB          65               CMP  #$AB
FØ 11          66               BEQ  LINE
C9 BØ          67               CMP  #$BØ
FØ ØD          68               BEQ  LINE
C9 C4          69               CMP  #$C4
DØ EF          7Ø               BNE  NXTBYT
AØ Ø1          71               LDY  #1
B1 B8          72               LDA  (TXTPTR),Y
2Ø BA ØØ       73               JSR  CHRTST
BØ E6          74               BCS  NXTBYT
2Ø 39 91       75   LINE        JSR  UPDATE
AØ ØØ          76   SORT        LDY  #$ØØ
A5 51          77               LDA  LINNUM+1
D1 FE          78               CMP  (POSITION),Y
9Ø 2Ø          79               BCC  MOVE
DØ 19          8Ø               BNE  SORT1
C8             81               INY
A5 5Ø          82               LDA  LINNUM
D1 FE          83               CMP  (POSITION),Y
9Ø 17          84               BCC  MOVE
DØ 1Ø          85               BNE  SORT1
C8             86               INY
A5 76          87               LDA  CURLIN+1
D1 FE          88               CMP  (POSITION),Y
9Ø ØE          89               BCC  MOVE
DØ Ø7          9Ø               BNE  SORT1
C8             91               INY
A5 75          92               LDA  CURLIN
D1 FE          93               CMP  (POSITION),Y
9Ø Ø5          94               BCC  MOVE
2Ø 2B 91       95   SORT1       JSR  ADD4
9Ø D8          96               BCC  SORT
A5 FE          97   MOVE        LDA  POSITION
85 9B          98               STA  LOWTR
A5 FF          99               LDA  POSITION+1
85 9C          1ØØ              STA  LOWTR+1
2Ø 9A D3       1Ø1              JSR  BLTU2
AØ ØØ          1Ø2              LDY  #$ØØ
A5 51          1Ø3              LDA  LINNUM+1
91 FE          1Ø4              STA  (POSITION),Y
C8             1Ø5              INY
A5 5Ø          1Ø6              LDA  LINNUM
91 FE          1Ø7              STA  (POSITION),Y
C8             1Ø8              INY
A5 76          1Ø9              LDA  CURLIN+1
91 FE          11Ø              STA  (POSITION),Y
C8             111              INY
A5 75          112              LDA  CURLIN
91 FE          113              STA  (POSITION),Y
2Ø 26 91       114              JSR  BYTGET1
C9 2C          115              CMP  #','
FØ AE          116              BEQ  LINE
C9 ØØ          117              CMP  #$ØØ
4C 4F 9Ø       118              JMP  NXTBYT1
2Ø 55 91       119  PRINT       JSR  INITPOS
A9 FF          12Ø              LDA  #$FF
85 5Ø          121              STA  LINNUM
85 51          122              STA  LINNUM+1
AØ Ø1          123  PRINT1      LDY  #$Ø1
B1 FE          124              LDA  (POSITION),Y
AA             125              TAX
88             126              DEY
31 FE          127              AND  (POSITION),Y
C9 FF          128              CMP  #$FF
FØ 44          129              BEQ  END
B1 FE          13Ø              LDA  (POSITION),Y
C5 51          131              CMP  LINNUM+1
85 51          132              STA  LINNUM+1
DØ Ø4          133              BNE  NOTSAME
E4 5Ø          134              CPX  LINNUM
FØ 18          135              BEQ  SAMELIN
86 5Ø          136  NOTSAME     STX  LINNUM
2Ø FB DA       137              JSR  CRDO
2Ø 1A D6       138              JSR  FNDLIN
BØ Ø3          139              BCS  FOUND
2Ø 5A DB       14Ø              JSR  OUTQUES
A6 5Ø          141  FOUND       LDX  LINNUM
A5 51          142              LDA  LINNUM+1
2Ø 24 ED       143              JSR  LINPRT
A9 Ø7          144              LDA  #$Ø7
85 24          145              STA  CH
AØ Ø3          146  SAMELIN     LDY  #$Ø3
B1 FE          147              LDA  (POSITION),Y
AA             148              TAX
88             149              DEY
B1 FE          15Ø              LDA  (POSITION),Y
2Ø 24 ED       151              JSR  LINPRT
2Ø 57 DB       152              JSR  OUTSP
A5 24          153              LDA  CH
C9 21          154              CMP  #33
9Ø Ø7          155              BCC  NOCR
2Ø FB DA       156              JSR  CRDO
A9 Ø6          157              LDA  #6
85 24          158              STA  CH
2Ø 2B 91       159  NOCR        JSR  ADD4
9Ø BØ          16Ø              BCC  PRINT1
2Ø FB DA       161  END         JSR  CRDO
68             162              PLA
85 B9          163              STA  TXTPTR+1
68             164              PLA
85 B8          165              STA  TXTPTR
6Ø             166              RTS
E6 B8          167  BYTGET      INC  TXTPTR
DØ Ø2          168              BNE  BYTGET1
E6 B9          169              INC  TXTPTR+1
AØ ØØ          17Ø  BYTGET1     LDY  #Ø
B1 B8          171              LDA  (TXTPTR),Y
6Ø             172              RTS
18             173  ADD4        CLC
A5 FE          174              LDA  POSITION
69 Ø4          175              ADC  #$Ø4
85 FE          176              STA  POSITION
A5 FF          177              LDA  POSITION+1
69 ØØ          178              ADC  #$ØØ
85 FF          179              STA  POSITION+1
6Ø             18Ø              RTS
2Ø 2Ø 91       181  UPDATE      JSR  BYTGET
18             182              CLC
2Ø ØC DA       183              JSR  LINGET
18             184              CLC
A5 7D          185              LDA  HIGH
85 96          186              STA  HIGHTR
69 Ø4          187              ADC  #$Ø4
85 7D          188              STA  HIGH
85 94          189              STA  HIGHDS
A5 7E          19Ø              LDA  HIGH+1
85 97          191              STA  HIGHTR+1
69 ØØ          192              ADC  #$ØØ
85 7E          193              STA  HIGH+1
85 95          194              STA  HIGHDS+1
A5 AF          195  INITPOS     LDA  PRGEND
85 FE          196              STA  POSITION
A5 BØ          197              LDA  PRGEND+1
85 FF          198              STA  POSITION+1
6Ø             199              RTS
```

Double-Hires-Routinen für den Apple II+

Eine Ampersand-Befehlssammlung

von Martin Orlamünder

Mit diesem Programm können jetzt auch Benutzer des Apple II+ oder kompatibler Geräte eine Grafikauflösung von 560 * 192 Punkten nutzen. Diese Applesoft-Erweiterung, die auch auf dem Apple IIe und IIc läuft, unterstützt die Double-Hires-Grafik durch bequeme Ampersand-Befehle. Die theoretischen Grundlagen der Pseudo-Double-Hires wurden schon im Peeker-Heft 6/85 beschrieben. Ist das Farbbit (Bit 7 des Grafik-Bytes) gesetzt, werden alle Bits dieses Bytes um einen halben Bildpunkt nach rechts verschoben dargestellt. Damit Punkte in halber wie auch in ganzer Schrittweite gleichzeitig erscheinen, werden alle Farbbits der 2. Hires-Page gesetzt und die von Hires-Page 1 gelöscht. Stellt man nun alle ungeraden X-Koordinaten auf der Seite 1 dar und die geraden auf der Seite 2, so ergeben sich in einer Zeile 560 Bildpunkte. Um HGR 1 und HGR 2 simultan anzuzeigen, wird in einer bestimmten Frequenz, die dem Monitor angepaßt werden muß, per Software zwischen den Grafikseiten umgeschaltet. Nur einfarbige Monitore ermöglichen Double-Hires.

Die Ampersand-Routinen werden initialisiert

– unter DOS 3.3 mit BRUN DHGR.IIPLUS und

– unter ProDOS mit BRUN DHGR. IIPLUS.PRO.

Das Demonstrationsprogramm kann unter DOS 3.3 direkt mit RUN DHGR.GENAU gestartet werden.

1. Die Ampersand-Befehle

Bei allen Anweisungen wird gegebenenfalls eine entsprechende Fehlermeldung ausgegeben. Die X-Koordinate liegt im Bereich von 0 bis 559, der Y-Wert reicht von 0 bis 191, und für Shape-Nummer gilt wie üblich der Bereich 1 bis 255.

&HGR

Der Double-Hires-Bildschirm wird initialisiert und je nach Plotmodus überdeckt, invertiert oder gelöscht. Dieser Befehl muß nach dem Systemstart vor dem ersten &HPLOT-, &DRAW- oder &TRACE-Statement aufgerufen werden.

&PLOT NORMAL
&PLOT INVERSE
&PLOT CLEAR

Nachfolgende &HPLOT-, &DRAW- oder &HGR-Befehle setzen (NORMAL), invertieren (INVERSE) bzw. löschen (CLEAR) Bildpunkte. Der Plotmodus ändert sich bis zum nächsten &PLOT ... nicht.

&ON

Die Double-Hires-Grafik wird angezeigt. Während dieser Zeit kann der Computer keine anderen Operationen ausführen, da er ständig zwischen HGR 1 und HGR 2 umschaltet. Das Drücken von **ESC** schaltet auf normale Textanzeige zurück. Eine 80-Zeichenkarte, die im Grafikmodus nicht automatisch auf 40 Z/Z umschaltet, sollte vorher desaktiviert werden.

&HPLOT x,y

(x: 0..559, y:0..191)

Dieser Befehl bietet dieselben Variationen wie der analoge Applesoft-Befehl. &HPLOT x,y zeichnet einen Punkt an den Koordinaten x,y im vorher definierten Plotmodus. &HPLOT TO x,y TO ... verbindet den zuletzt gezeichneten Punkt mit x,y, diesen mit dem nächsten usw. &HPLOT x1,y1 TO x2,y2 TO ... plottet eine Linie von x1,y1 nach x2,y2 usw. Beispielsweise erhält man durch &HPLOT 0,0 TO 559,191 eine Diagonale von links oben nach rechts unten.

&DRAW s AT x,y

(s: 1..255)

Diese Anweisung ist in gewohnter Weise zu gebrauchen. &DRAW 3 AT 400,80 zeichnet das 3. Shape der Shapetabelle an der Stelle 400, 80. &DRAW 3 plottet das Shape an der letzten Bildschirmkoordinate. Der Aufbau der Vektorentabelle, deren Anfangsadresse in $00E8/$00E9 (232/233) stehen muß, wurde von Applesoft übernommen. Die Shape-Nummern können von 1 bis 255 reichen. Die normalen Applesoft-Befehle SCALE= Vergrößerungsfaktor und ROT= Drehwinkel beziehen sich auch auf Double-Hires-Shapes. Der bisherige Befehl **XDRAW** kann durch &PLOT INVERSE : &DRAW ersetzt werden. Reicht ein Shape über den Bildschirmrand hinaus, so erfolgt kein „Überlauf" wie bei der Interpreter-Routine. Dieser Effekt stört das Hires-Bild meistens nur durch ungewollte Bruchstücke.

&TRACE FLAG AT x,y

Dieser Befehl prüft, ob der Bildpunkt x,y gesetzt oder gelöscht ist. Ist der Punkt sichtbar, wird der Variablen FLAG der Wert 1 zugewiesen, sonst der Wert 0.

&STORE

Dieser Befehl ist für das Drucken eines Double-Hires-Bildes gedacht. Wie CONVERT560 aus Peeker, Heft 5/85 zerlegt &STORE die DHGR-Grafik in zwei Hälften. Nach dem Aufruf befindet sich die linke Hälfte in Hires-Seite 1 und die rechte in 2. Das Drucker-Interface muß in der Lage sein, HGR 1 und HGR 2 nebeneinander in doppelter Dichte zu plotten. Dadurch wird das Bild in normalen Proportionen ausgedruckt.

&RESTORE

Hiermit kann ein durch &STORE zerlegtes Bild in das ursprüngliche Double-Hires-Format zurückverwandelt werden. So braucht man eine DHGR-Grafik vor dem Drucken nicht zwischenzuspeichern – ein DHGR-Bild benötigt immerhin 16K.

2. Technische Einzelheiten

Vor dem Assemblieren von T.DHGR.II PLUS muß das Label **DOSFLAG** je nach DOS-Version definiert werden. Für Nicht-Merlin-Kenner: Ist der Operand hinter „DO" gleich 0 (gilt für DHGR.IIPLUS.PRO = ProDOS), wird nur der Teil nach „ELSE" assembliert; ist er ungleich 0 (gilt für DHGR.IIPLUS = DOS 3.3), wird zwischen DO und ELSE assembliert.
Nach dem Start von DHGR.IIPLUS oder DHGR.IIPLUS.PRO wird das Programm von Diskette in den 1. Hires-Puffer ab $2000 eingelesen; unter ProDOS – hier sind die Interrupt-Vektoren schon in der LC installiert – ist es 1851 Bytes, unter DOS 3.3 1899 Bytes lang. Von dort wird das Hauptprogramm nach $D40C in die Language Card verschoben, nachdem die Tabelle mit den Anfangsadressen der Hires-Zeilen generiert wurde. Es nimmt vom jeweiligen DOS unbenutzten Speicherplatz ein: Die ProDOS-Version liegt in der Bank 2 über dem Reboot-Programm, die DOS 3.3-Version in Bank 1 und verträgt sich mit gemovtem DOS 3.3, Diversi-DOS oder David-DOS.
Der Driver in Page 3 bildet das Zwischenglied vom BASIC-Interpreter zum Hauptprogramm: Er schaltet die LC an oder aus und reicht von $03BA (ProDOS 3.3) bzw. $03A8 (DOS 3.3) bis $03CF. Nur auf diesen geringen Speicherplatz muß der BASIC-Programmierer verzichten. Die Rückkehr von der LC zum BASIC-Programm erfolgt nicht über den Driver: Sobald die LC mit „LDA $C08x" abgestellt ist, enthält der Befehlszähler des 6502 $D40F; jedoch wird der Operationscode jetzt aus dem ROM geholt. Da aber dort ein „RTS" steht, kehrt das Programm korrekt zum BASIC zurück!
Stehen nur 48K Speicher zur Verfügung (gilt für DHGR.IIPLUS.48K), kann PROGBEG auf einen anderen Wert gesetzt werden, z.B. $6000, und in Zeile 267 muß dann ein „RTS" eingefügt werden. Vor dem Assemblieren wird DOSFLAG = 0 gesetzt. Das Programm läuft dann einwandfrei, kann aber vereinfacht werden: Der Driver kann entfallen, die ROM-Routinen können direkt aufgerufen werden, das Programm braucht nicht mehr verschoben zu werden.
Auf der Zero-Page werden außer wenigen vom Interpreter ungenutzten Speicherstellen alle Adressen der normalen Hires-Grafik verwendet – normale und Double-Hires-Grafik kann man ohnehin nicht parallel betreiben.

Warum sind die Routinen HPLOT TO und DRAW so aufwendig gehalten? Eine hohe Grafikauflösung kommt nur mit einem entsprechenden Plot-Algorithmus zur Geltung. Die Linien, die mit normaler Double-Hires-Grafik geplottet wurden, erscheinen unnötig grob; das Applesoft-Programm **DHGR.GENAU** zeigt u.a. den Unterschied zwischen FOR-NEXT-Schleife und HPLOT TO bei Standardgrafik. Bei Double-Hires-II+ sind alle Linien deckungsgleich, besonders weil sie immer von „links nach rechts" geplottet werden. Trotzdem benötigt die &-HPLOT-TO-Routine nicht mehr Zeit für gleichviele Bildpunkte. Um effizienter zu arbeiten, modifiziert Double-Hires-II+ an einigen Stellen seinen eigenen Code. Wegen ähnlicher Funktion wurden hauptsächlich in der Shape-Routine Teile aus dem ROM übernommen.
Das Applesoft-Programm **DHGR.EPSON** veranschaulicht weitere Befehle und druckt eine DHGR-Grafik auf dem Epson-RX80 mit dem 8132W-Interface aus. (Für andere grafikfähige Interface-Karten muß in Zeile 270 der entsprechende Dump-Befehl eingesetzt werden.)

Kurzhinweise

1. Zweck:
Double-Hires-Ampersand-Routinen für Apple II+
2. Konfiguration:
Apple II+ (natürlich auch IIe/c); DOS 3.3 oder ProDOS
3. Test:
RUN DHGR.GENAU
4. Sammeldisk:
T.DHGR.IIPLUS
(allgemeiner Big-Mac-Quelltext)
DHGR.IIPLUS
(LC-Version für DOS 3.3)
DHGR.IIPLUS.PRO
(LC-Version für ProDOS)
DHGR.IIPLUS.48K
(48K-Version, siehe Aufsatz)
DHGR.GENAU
(Demo)
DHGR.EPSON
(Druck-Demo für RX80)
5. Sonstiges:
Man beachte, daß die DHGR-Anzeige jeweils mit ESC abgebrochen werden muß.

DHGR.EPSON

```
  1 HOME : PRINT "SETZT EPSON-RX80 MIT
    8132W VORAUS": PRINT "WENN JA,
    ZEILE 1 LOESCHEN": LIST 1: END
100 REM Druckt Figur und demonstriert
    neue &-Befehle
110 D$ = CHR$ (13) + CHR$ (4):
    HOME : VTAB 12: REM Bei ProDOS
    nur D$ = CHR$(4) ohne Return!
120 PRINT D$"BRUN DHGR.IIPLUS"
130 PRINT "APPLE ZEICHNET -
    BITTE WARTEN!": PRINT
140 & PLOT CLEAR : & HGR :
    & PLOT NORMAL
150 AX = 200:AY = AX:X0 = AX + 100:
    Y0 = 96:S = 1 / AX
160 DEF FN F(X) = X ↑ 2 * SQR (X + 1)
170 FOR X = - 1 TO 0.3 STEP S
180 & HPLOT X0 + AX * X,Y0 -
    AY * FN F(X)
190 & HPLOT X0 + AX * X,Y0 +
    AY * FN F(X)
200 NEXT
210 & ON : REM Grafik anzeigen,
    weiter mit <ESC>
220 & PLOT INVERSE : & HGR : & ON :
    & HGR : & ON
230 & STORE : POKE 49232,0:
    GET T$: REM HGR 1 anzeigen
240 POKE 49237,0: GET T$: REM HGR 2
250 REM HGR 1 & HGR 2 nebeneinander
    in doppelter Dichte
260 REM Drucker: EPSON RX-80,
    Interface: EPSON #8132W
270 PRINT D$"PR#1":
    PRINT CHR$ (9)"GBD": PRINT D$"PR#0"
280 & RESTORE : & ON :
    REM wieder Double-Hires
290 GOSUB 310: GOSUB 310: REM invertiert
300 END
310 & HPLOT 10,10: & TRACE FLAG AT 10,10
320 PRINT "FLAG= "FLAG: RETURN
```

DHGR.GENAU

```
100 REM Zeigt die Feinheit von normaler
    Grafik und Double-Hires-II+
110 DATA 1,0,4,0,44,62,0: REM Quadrat
120 POKE 232,0: POKE 233,3:
    REM Adresse für Shapetable
130 FOR I = 768 TO 774: READ Q:
    POKE I,Q: NEXT
140 PRINT CHR$ (4)"BRUN DHGR.IIPLUS"
150 X1 = 0:Y1 = 0:X2 = 100:Y2 = 191:
    DX = X2 - X1:DY = Y2 - Y1
160 REM ---------Normale Apple-Grafik
170 HGR2 : HCOLOR= 3: HPLOT X1,Y1
    TO X2,Y2
180 FOR I = 0 TO DY: HPLOT DX + I *
    DX / DY + 0.5,I + 0.5: NEXT
190 SCALE= 35: ROT= 7: DRAW 1 AT 200,50:
    GET T$
200 HCOLOR= 0: HPLOT X2,Y2 TO X1,Y1:
    GET T$: REM nicht deckungsgleich
210 REM --------Double-Hires-II+
220 TEXT : HOME : VTAB 12: HTAB 10:
    PRINT "Double-Hires-II+"
230 DX = 2 * DX:X2 = 2 * X2:
    REM 2 * 280 Punkte
240 & PLOT CLEAR : & HGR :
    REM Clear Screen
250 & PLOT NORMAL : & DRAW 1 AT 400,50:
    & HPLOT X1,Y1 TO X2,Y2
260 FOR I = 0 TO DY: & HPLOT DX + I *
    DX / DY + 0.5,I + 0.5: NEXT
270 & ON : & PLOT CLEAR :
    & HPLOT X2,Y2 TO X1,Y1:
    & ON : REM ok.
```

DHGR.IIPLUS

```
M. Orlamünder, Usingen, 1985
DOSFLAG muß vor Assemblierung gesetzt
werden: ProDOS = 0, DOS 3.3 <> 0
DOSFLAG  EQU  1
Applesoft-Tokens
ATTOK    EQU  197
CLEARTOK EQU  189
DRAWTOK  EQU  148
HGRTOK   EQU  145
HPLOTTOK EQU  147
INVTOK   EQU  158
NRMALTOK EQU  157
ONTOK    EQU  180
PLOTTOK  EQU  141
RSTORTOK EQU  174
STORETOK EQU  168
TOTOK    EQU  193
TRACETOK EQU  155
Operationscodes. Einige Routinen werden
vom laufenden Programm aus modifiziert
AND_CODE EQU  $31        ;AND ( ),Y
BCC_CODE EQU  $90
BCS_CODE EQU  $B0
DEX_CODE EQU  $CA
EOR_CODE EQU  $51        ;EOR ( ),Y
INX_CODE EQU  $E8
JMP_CODE EQU  $4C
JSR_CODE EQU  $20
LDA_CODE EQU  $A5        ;LDA Zeropage
LDY_CODE EQU  $A4        ;LDY  "
ORA_CODE EQU  $11        ;ORA ( ),Y
RTS_CODE EQU  $60
Variable Operanden, vom Prog. definiert
ADDR16   EQU  $FFFF      ;16-Bit-Adresse
DATA8    EQU  $FF        ;8Bit immediate
Applesoft- und Monitor-Routinen
CHRGET   EQU  $00B1      ;erhöht TXTPTR,
CHRGOT   EQU  $00B7      ;liest Zeichen
GETBYT   EQU  $E6F8      ; "   8-Bitwert
GETNUM   EQU  $E746      ; "   16,8- "
HPOSN    EQU  $F411      ;calc HGRcursor
PTRGET   EQU  $DFE3      ;Var-Adr./Name
SNGFLT   EQU  $E301      ;YREG -> FAC
MISMTCH  EQU  $DD76      ;TYPE MISMATCH
SYNERR   EQU  $DEC9      ;SYNTAX
IQERR    EQU  $E199      ;ILLEGAL QUANT.
MOVE     EQU  $FE2C      ;A1-A2 nach A4
Tastaturadressen
KBD      EQU  $C000      ;Keyboard Data
KBDSTRB  EQU  $C010      ;löscht Strobe
Benutzte Parameter von Basic und Monitor
A1       EQU  $3C        ;Anfangs-,
A2       EQU  $3E        ;End-,
A4       EQU  $42        ;Zieladresse
AMPERV   EQU  $3F5       ;Ampervektor
FAC      EQU  $9D        ;Float-Akku
GBAS     EQU  $26        ;HGR-Zeilenadr.
HPAG     EQU  $E6        ;Flag: HGR-Page
LINNUM   EQU  $50        ;16-Bit-Zahl
SHAPEPNT EQU  $E8        ;Shapetable
SCALEZ   EQU  $E7        ;Scalefaktor
SUBFLG   EQU  $14        ;Variablenflag
ROTZ     EQU  $F9        ;Drehwinkel
VARNAM   EQU  $81        ;letzter Vname
Programmeigene Variablen
PLOT     EQU  $06        ;HGR-Byte
HIRES1   EQU  PLOT       ;indirekt nach-
HIRES2   EQU  LINNUM     ;indiz.: HGR1/2
X1       EQU  $08        ;Ausgangs-
Y1       EQU  $1A        ;Koordinate
X2       EQU  $1C        ;End-
Y2       EQU  $1E        ;Koordinate
DX       EQU  GBAS       ;Delta-X
DY       EQU  $30        ;Delta-Y
SIGN     EQU  FAC        ;Vorzeichen
Y0       EQU  FAC+1      ;Y-Startwert
EXORMASK EQU  $D0        ;für Plotmodus
PAGEFLAG EQU  $D1        ;Übergabewert
PAGEMASK EQU  $D2        ;für HGR1/HGR2
BITWERT  EQU  $D3        ;Bit(0-6) im
SPALTE   EQU  $D4        ;HGR-Byte(0-39)
PGMSKSAV EQU  $D5        ;keine laufen-
BITWTSAV EQU  $E0        ;den, sondern
SPLTSAV  EQU  $E1        ;Speicherwerte
STEIGFAK EQU  $EB        ;Geradensteigng
VAR      EQU  FAC+2      ;wenn momentane
COSVAR   EQU  $E2        ;VAR überläuft,
SINVAR   EQU  $E3        ;nächstes Pixel
SHAPE    EQU  $E4        ;Shapebyte
PLOTFLAG EQU  HPAG       ;(un)sichtbarer
VEKTOR   EQU  $EA        ;Shapevektor
QDRNT    EQU  FAC+4      ;INT (ROT/16) =
YSAVE    EQU  FAC+5      ;0.-3. Quadrant
HGR1BYT  EQU  COSVAR     ;Bytes der
HGR2BYT  EQU  SINVAR     ;HGR-Seiten
VARPOINT EQU  X1         ;zeigt auf Var.
LC-Softswitches und Interruptvektoren
         DO   DOSFLAG    ;nur DOS 3.3:
NMI      EQU  $FFFA      ;Adressen der
RESET    EQU  $FFFC      ;Interrupt-
IRQ      EQU  $FFFE      ;Vektoren
RDROWRAM EQU  $C089      ;Bank 1 der
READROM  EQU  $C08A      ;Language Card
RWRAM    EQU  $C08B      ;$D000 - $DFFF
DRIVBEG  EQU  $3D0-22-18 ;Driver=40 Byte
         ELSE            ;nur ProDOS:
RDROWRAM EQU  $C081      ;Bank 2
READROM  EQU  $C082
RWRAM    EQU  $C083
DRIVBEG  EQU  $3D0-22    ;Driver=22 Byte
         FIN
PROGBEG  EQU  $D40C      ;Hauptprogramm
```

Initialisierungsteil

```
         ORG  $2000
HGR-Zeilentabelle anlegen
         LDA  #$20       ;Seite 1, aber
         STA  HPAG       ;hier egal
         LDA  #0         ;HPOSN-Entry:
         STA  Y0         ;AREG= Y-Koord.
CALCADDR LDX  #0         ;XREG= Xlow
         LDY  #0         ;YREG= Xhigh
         JSR  HPOSN      ;Exit: Adresse
         LDY  Y0         ;der HGR-Zeile
         LDA  GBAS       ;in GBAS
         STA  INITEND+1+DRIVEND-
              DRIVBEG+1+YTABL-PROGBEG,Y
         LDA  GBAS+1     ;2 MSBs durch
         AND  #%000011111 ;PAGEMASK best.
         STA  INITEND+1+DRIVEND-
              DRIVBEG+1+YTABH-PROGBEG,Y
         INC  Y0
         LDA  Y0
         CMP  #192       ;Y < 192?
         BCC  CALCADDR
Ampersandvektor installieren
         LDA  #JMP_CODE
         STA  AMPERV
         LDA  #<START
         STA  AMPERV+1
         LDA  #>START
         STA  AMPERV+2
Driver unter DOS-vektoren ab $3D0 moven
         LDA  #<INITEND+1
         STA  A1
         LDA  #>INITEND+1
         STA  A1+1
         LDA  #<INITEND+1+
              DRIVEND-DRIVBEG
         STA  A2
         LDA  #>INITEND+1+
              DRIVEND-DRIVBEG
         STA  A2+1
         LDA  #<DRIVBEG
         STA  A4
         LDA  #>DRIVBEG
         STA  A4+1
         LDY  #0         ;Bedingung für
         JSR  MOVE       ;MOVE
Programm in LC (DOS-freier Platz) moven
         LDA  #<INITEND+1+
              DRIVEND-DRIVBEG+1
         STA  A1
         LDA  #>INITEND+1+
              DRIVEND-DRIVBEG+1
         STA  A1+1
         LDA  #<INITEND+1+DRIVEND-
              DRIVBEG+1+PROGEND-PROGBEG
         STA  A2
         LDA  #>INITEND+1+DRIVEND-
              DRIVBEG+1+PROGEND-PROGBEG
         STA  A2+1
         LDA  #<PROGBEG
         STA  A4
         LDA  #>PROGBEG
         STA  A4+1
         LDY  #0
         BIT  RDROWRAM   ;ROM lesen,
         BIT  RDROWRAM   ;LC schreiben
         JSR  MOVE
         DO   DOSFLAG
         LDA  #<GONMI    ;bei DOS 3.3
         STA  NMI        ;müssen die
         LDA  #>GONMI    ;Interrupt-
         STA  NMI+1      ;Vektoren in
         LDA  #<GORESET  ;der LC instal-
         STA  RESET      ;liert werden;
         LDA  #>GORESET  ;der Driver
         STA  RESET+1    ;ist daher
         LDA  #<GOIRQ    ;18 Bytes
         STA  IRQ        ;länger als
         LDA  #>GOIRQ    ;bei ProDOS
```

```
        STA  IRQ+1
        FIN
        BIT  READROM     ;LC aus
INITEND RTS
```

DRIVER für Language Card

```
        ORG  DRIVBEG
        DO   DOSFLAG     ;nur DOS 3.3:
GONMI   BIT  READROM     ;springe zu
        JMP  (NMI)       ;Interrupt-
GORESET BIT  READROM     ;Vektoren über
        JMP  (RESET)     ;ROM-Adresse!
GOIRQ   BIT  READROM
        JMP  (IRQ)
        FIN
START   BIT  RWRAM       ;LC lese- und
        BIT  RWRAM       ;schreibfähig
        JMP  CHKTOKEN    ;zur LC
GOROM   BIT  READROM     ;Dieser Teil
ROUTINE JSR  ADDR16      ;wird benutzt,
        BIT  RWRAM       ;um von der LC
        BIT  RWRAM       ;ROM-Routinen
DRIVEND RTS              ;anzuspringen.
```

Hauptprogramm in der Language Card

```
PROGBEG muß bei $D4ØC liegen!!!
Op-Code nach "BIT $CØ8x" wird aus R O M
gelesen! $D4ØF enthält nötigen RTS-CODE
Ist z.B. PROGBEG = $D4ØØ, so stürzt
das Programm ab!
        ORG  PROGBEG
EXIT    BIT  READROM     ;-> BASIC
$D4ØF:  RTS              ;steht im ROM!
```

&HGR

```
löscht/färbt DHGR-Schirm normal/inverse
INITHGR LDY  #Ø
        STY  HIRES1
        STY  HIRES2
        LDX  #$4Ø        ;Seite 2
        STX  HIRES2+1
        LDX  #$2Ø        ;Seite 1
        STX  HIRES1+1
HGRLOOP LDA  #$FF        ;NORMAL/INV/CLR
        EOR  EXORMASK    ;$ØØ/$ØØ/$FF
PLOT1   ORA  (HIRES1),Y  ;ORA/EOR/AND
        ORA  #%1ØØØØØØØ  ;Farbbit = 1,
        STA  (HIRES1),Y  ;ungerade X-
        LDA  #$FF        ;Werte in HGR1
        EOR  EXORMASK
PLOT2   ORA  (HIRES2),Y
        AND  #%Ø1111111  ;Farbbit = Ø
        STA  (HIRES2),Y
        INY
        BNE  HGRLOOP
        INC  HIRES1+1
        INC  HIRES2+1
        DEX              ;$2Ø mal
        BNE  HGRLOOP
        JMP  EXIT
```

AMPER-START

```
Ampersand-Befehl auswerten;
AREG enthält bereits Token nach "&".
CHKTOKEN TAY
        JSR  CHRGET      ;CHRGET erhöht
        CPY  #HPLOTTOK   ;TXTPTR um 1
        BEQ  HPLOT       ;und geht dann
        CPY  #DRAWTOK    ;über in CHRGOT
        BNE  NODRAW      ;Dieser Teil
        JMP  DRAW        ;holt Token ab
NODRAW  CPY  #ONTOK      ;TXTPTR in AREG
        BEQ  PAGEFLIP
        CPY  #HGRTOK
        BEQ  INITHGR
        CPY  #TRACETOK
        BNE  NO_TRACE
        JMP  TRACE
NO_TRACE CPY #PLOTTOK
        BNE  OLDPLOT
        JMP  PLOTMODE
OLDPLOT CPY  #STORETOK
        BNE  NOSTORE
        JMP  STORE
NOSTORE CPY  #RSTORTOK
        BNE  SYNTERR1    ;kein &-Befehl
        JMP  RESTORE
SYNTERR1 LDX #<SYNERR    ;Adresse der
        LDY  #>SYNERR    ;ROM-Routine
        JMP  PRINTERR    ;laden
```

&ON

```
schaltet schnell zwischen HGR1 und HGR2
hin und her,um eine gleichzeitige
Anzeige vorzutäuschen - Stop mit <ESC>.
PAGEFLIP LDA KBDSTRB
        LDA  $CØ52       ;nur Grafik
        LDA  $CØ57       ;Hi-Res
        LDA  $CØ5Ø       ;Grafikanzeige
        LDX  #Ø          ;Ø oder 1 egal
WECHSEL LDA  $CØ54,X     ;HGR1/HGR2
        TXA
        EOR  #1          ;andere Seite
Verzögerung, damit Pageflipping der
"Monitor-Auffrischung" nicht vorauseilt
        LDX  #$1Ø        ;monitor-
DELAY1  LDY  #$FC        ;spezifische
DELAY2  DEY              ;Konstanten
        BNE  DELAY2
        DEX
        BNE  DELAY1
        TAX
        LDA  KBD         ;wurde <ESC>
        CMP  #$9B        ;gedrückt?
        BNE  WECHSEL     ;nein!
        LDA  $CØ54       ;Seite 1
        LDA  $CØ51       ;Textanzeige
        LDA  KBDSTRB     ;Tastendruck
EXIT1   JMP  EXIT        ;löschen
```

```
&HPLOT x,y / &HPLOT TO x,y TO ... /
&HPLOT x1,y1 TO x2,y2 TO ...
```

```
HPLOT   CMP  #TOTOK      ;HPLOT TO x,y?
        BEQ  HPLOTTO
        JSR  GETXY       ;HPLOT x,y
HPLOT1  STY  X1
        STA  X1+1
        STX  Y1
        JSR  POSITION    ;Parameter für
        STA  BITWERT     ;Plotten holen
        STY  SPALTE
        STX  PAGEMASK
        LDX  Y1
        JSR  PLOTXY
CHECK_TO JSR CHRGOT      ;weitere
        CMP  #TOTOK      ;Verbindungs-
        BNE  EXIT1       ;linie plotten?
HPLOTTO JSR  CHRGET
        JSR  GETXY
        STY  X2
        STA  X2+1
        STX  Y2
HPLOTTO1 JSR POSITION    ;Daten für
        STA  BITWTSAV    ;Verbinduns-
        STY  SPLTSAV     ;punkt
        STX  PGMSKSAV
        TXA              ;Verbindungsp.
        LDX  Y2          ;zeichnen,
        ORA  YTABH,X     ;damit bei
        STA  PLOT+1      ;Vertauschen
        LDA  YTABL,X     ;von P1(X1;Y1)
        STA  PLOT        ;mit P2(X2;Y2)
        LDA  BITWTSAV    ;Anfangspunkt
        EOR  EXORMASK    ;geplottet wird
PLOT3   ORA  (PLOT),Y
        STA  (PLOT),Y
Mit Geradengleichung P1 & P2 verbinden:
Y = M*X + YØ;  M = (Y2-Y1)/(X2-X1);
SIGN = SGN(M); DY/DX = ABS(M)
        SEC              ;DY= ABS(Y2-Y1)
        LDA  Y2          ;zunächst:
        SBC  Y1          ;DY = Y2-Y1
        LDX  #127        ;SIGN positiv
        BCS  DY_OK       ;Y2 >=Y1,ABS ok
        EOR  #$FF        ;DY = ABS(DY)
        ADC  #1          ;Y2 < Y1, daher
        INX              ;SIGN negativ
        SEC
DY_OK   STA  DY
Startpunkt(XØ;YØ) hat kleineren X-Wert,
Endpunkt(Xend;Yend) größeren von P1/P2
        LDA  X2          ;DX= ABS(X2-X1)
        SBC  X1          ;zunächst:
        STA  DX          ;DX = X2-X1
        LDA  X2+1
        SBC  X1+1
        STA  DX+1
        LDY  Y1          ;Default für YØ
        BCS  DX_OK       ;X2 >=X1,ABS ok
        EOR  #$FF        ;DX = ABS(DX)
        STA  DX+1
        LDA  DX
        EOR  #$FF
        ADC  #1
        STA  DX
        TXA
        EOR  #%1ØØØØØØØ ;SIGN = -SIGN
        TAX
        LDA  BITWTSAV    ;P2(X2;Y2) als
        STA  BITWERT     ;Startpunkt
        LDA  PGMSKSAV
        STA  PAGEMASK
        LDA  SPLTSAV
        STA  SPALTE
        LDY  Y2
DX_OK   STY  YØ
        STX  SIGN
        LDA  #Ø          ;Variablen
        STA  VAR         ;initialisieren
        STA  STEIGFAK+1
        STA  STEIGFAK+2
        STA  STEIGFAK+3
        STA  STEIGFAK+4
        LDA  #$8Ø        ;VAR=$8ØØØ: ab
        STA  VAR+1       ;Ø.5 aufrunden
        LDX  #DEX_CODE   ;je nach SIGN
        BIT  SIGN        ;YØ in- bzw.
        BMI  CODE_OK     ;dekrementieren
        LDX  #INX_CODE   ;YØ+/-M*(X2-X1)
Wenn DX>DY, dann X-Wert als Urbild, das
pro Schleifendurchlauf um 1 erhöht wird
Y als Funktion von X. Wenn DX<=DY, dann
Y-Wert als Urbild, X als Funktion von Y
CODE_OK LDA  DX+1
        BNE  X=URBILD    ;DX > DY
        LDY  DY          ;wenn DX(DY) <2
        TYA              ;dann liegen
        ORA  DX          ;keine Punkte
        CMP  #2          ;zwischen P1-P2
        BCC  NEXT_TO1    ;P1&P2 bereits
        CPY  DX          ;geplottet!
        BCC  X=URBLD1    ;DX > DY
        BCS  Y=URBILD    ;DX <= DY
Yplot=M*(Xplot-XØ) +YØ; Xplot=XØ...Xend
X=URBILD CLC
X=URBLD1 STX X_INXDEX
        LDY  DX          ;DX wird als
        STY  DVSORL+1    ;Operand in
        LDA  DX+1        ;Divisions-
        STA  DVSORH+1    ;Routine gepokt
        LDA  #2          ;C=Ø!
        SBC  DX          ;DX = Counter
        STA  DX          ;für Inkrement.
        LDA  #Ø          ;von Xplot
        SBC  DX+1        ;DX = 1-DX
        STA  DX+1        ;   = -(DX-1)
        LDA  DY          ;STEIGFAK =
        JSR  CALCSTGF    ;DY/DX * 2↑16
Linie zeichnen
X_LINE  CLC              ;VAR =
        LDA  VAR         ;VAR + STEIGFAK
        ADC  STEIGFAK    ;Yplot = Yplot
        STA  VAR         ;+/- VAR / 2↑16
        LDA  VAR+1       ;bei VAR>$FFFF:
        ADC  STEIGFAK+1  ;Yplot =
        STA  VAR+1       ;Yplot +/- 1
        BCC  Y=CONST     ;VAR/2↑16 < 1
X_INXDEX INX             ;Yplot=Yplot+-1
Y=CONST JSR  INCXPLOT    ;Xplot=Xplot+1
        INC  DX          ;wurde STEIGFAK
        BNE  X_LINE      ;DX-1 mal
        INC  DX+1        ;zu VAR
        BNE  X_LINE      ;addiert?
NEXT_TO LDA  PGMSKSAV    ;Ausgangspunkt:
        STA  PAGEMASK    ;P1 = P2
        LDY  SPLTSAV
        LDX  BITWTSAV
        STY  SPALTE
        STX  BITWERT
NEXT_TO1 LDA X2          ;Yplot = Yend =
        STA  X1          ;YØ+DX* DY/DX
        LDA  X2+1        ;aus altem
        STA  X1+1        ;Verbindungs-
        LDA  Y2          ;punkt wird
        STA  Y1          ;neuer
        JMP  CHECK_TO    ;weitere Linie?
Xplot=(Yplot-YØ)/M +XØ; Yplot=YØ...Yend
Y=URBILD STX Y_INXDEX
        STY  DVSORL+1    ;DY = Counter
        DEC  DY          ;für Yplot
        LDA  #Ø          ;DY < 256
        STA  DVSORH+1
```

```
          LDA  DX          ;STEIGFAK =
          JSR  CALCSTGF    ;DX/DY * 2↑16
          LDA  #BCC_CODE   ;Carry = MSB
          BCC  SETBRNCH    ;von STEIGFAK
          LDA  #BCS_CODE   ;C=1: STEIGFAK=
SETBRNCH  STA  BRANCH      ;$10000, DX=DY
Y_LINE    CLC
          LDA  VAR         ;Xplot = Xplot
          ADC  STEIGFAK    ;+/- VAR/$10000
          STA  VAR
          LDA  VAR+1
          ADC  STEIGFAK+1
          STA  VAR+1
Y_INXDEX  INX              ;Yplot=Yplot+-1
BRANCH    BCC  X=CONST     ;VAR < $10000
          JSR  INCXPLOT    ;Xplot=Xplot+1
          DEC  DY          ;Yplot DY-1 mal
          BNE  Y_LINE      ;inkrementiert?
          JMP  NEXT_TO     ;fertig!
X=CONST   JSR  PLOTXY      ;Xplot nicht
          DEC  DY          ;erhöhen!
          BNE  Y_LINE
          JMP  NEXT_TO
Subroutinen für HPLOT TO:
-------------------------------------
CALCSTGF berechnet STEIGFAK=2↑16*M(1/M)
Da DY/DX(DX/DY) ein Bruch von Ø bis 1,
Dividend=Dividend * $10000 vor Division
Entry: AREG = Dividend = DY oder DX
CALCSTGF LDX  #16          ;$10000 = 2↑16
MULTIPLY ASL               ;eine Geraden-
         ROL  STEIGFAK+1   ;steigung M=Ø.5
         ROL  STEIGFAK+2   ;entspricht
         DEX               ;STEIGFAK=$8000
         BNE  MULTIPLY
         STA  STEIGFAK
24Bit/16Bit - Division
Entry: STEIGFAK = Dividend, Divisor in
DVSORL+1/H+1. Exit: STEIGFAK = Quotient
         LDX  #24          ;24Bit-Dividend
         CLC
DIVLP    ROL  STEIGFAK     ;shift Carry in
         ROL  STEIGFAK+1   ;Bit Ø des
         ROL  STEIGFAK+2   ;Dividenden;
         ROL  STEIGFAK+3   ;dies ergiebt
         ROL  STEIGFAK+4   ;Quotient.
         SEC               ;versuche,
         LDA  STEIGFAK+3   ;Divisor zu
DVSORL   SBC  #DATA8       ;subtrahieren
         TAY
         LDA  STEIGFAK+4
DVSORH   SBC  #DATA8
         BCC  DECCNT       ;geht nicht!
         STY  STEIGFAK+3   ;Dividend =
         STA  STEIGFAK+4   ;Dividend-Dvsor
DECCNT   DEX
         BNE  DIVLP
         ROL  STEIGFAK     ;letztes Carry
         ROL  STEIGFAK+1   ;-> Quotient
         LDX  YØ           ;Yplot = YØ
         LDY  SPALTE
         RTS
INCXPLOT inkrementiert die laufende
X-Koordinate Xplot, d.h.BITWERT & SPALTE
INCXPLOT LDA  PAGEMASK     ;HGR-Seite
         EOR  #%Ø11ØØØØØ   ;wechseln
         STA  PAGEMASK     ;1&2 i.Wechsel:
         ASL               ;1)Farbbit Ø=>1
         BPL  PLOTXY       ;d.h.HGR2=>HGR1
         ASL  BITWERT      ;2)shift HGRBit
         BPL  PLOTXY       ;nur Bit Ø-6,
         INY               ;sonst next
         LDA  #1           ;Byte & mit
         STA  BITWERT      ;Bit Ø starten
Punkt setzen, invertieren, löschen
PLOTXY   LDA  YTABL,X      ;hole Adresse
         STA  PLOT         ;der HGR-Zeile,
         LDA  YTABH,X      ;Ø. Spalte
         ORA  PAGEMASK     ;HGR1/HGR2
         STA  PLOT+1
         LDA  BITWERT
         EOR  EXORMASK     ;Plotmodus be-
PLOT4    ORA  (PLOT),Y     ;rücksichtigen
         STA  (PLOT),Y
         RTS
-------------------------------------
&DRAW ausnm / &DRAW ausnm AT x,y
=====================================
DRAW     LDA  #<GETBYT     ;Hole Shape-
         STA  ROUTINE+1    ;nummer "ausnm"
         LDA  #>GETBYT     ;aus Basictext
         STA  ROUTINE+2    ;-> XREG
         JSR  GOROM
```

```
         LDA  SHAPEPNT
         STA  SHAPE
         LDA  SHAPEPNT+1
         STA  SHAPE+1
         TXA               ;ist Shape in
         LDX  #Ø           ;Shapetable
         CMP  (SHAPE,X)    ;definiert?
         BEQ  SHAPE_OK     ;letztes Shape
         BCC  SHAPE_OK
         JMP  ILLQUERR     ;nicht enthalt.
SHAPE_OK ASL               ;2 Byte geben
         BCC  DP2          ;relatives
         INC  SHAPE+1      ;Displacement
         CLC               ;vom Beginn der
DP2      TAY               ;Shapetable bis
         LDA  (SHAPE),Y    ;zum Shape an.
         ADC  SHAPE        ;Displacement
         TAX               ;zum Beginn der
         INY               ;Shapetable
         LDA  (SHAPE),Y    ;addieren,
         ADC  SHAPEPNT+1   ;erhalte
         STA  SHAPE+1      ;Anfangsadresse
         STX  SHAPE        ;des Shapes
         JSR  CHRGOT
         CMP  #ATTOK
         BNE  DRAW1        ;bei letztem
         JSR  CHRGET       ;Punkt starten
         JSR  GETXY        ;lege neuen
         STX  Y1           ;Startpunkt
         STY  X1           ;fest!
         STA  X1+1
         STX  Y2
         STY  X2
         STA  X2+1
DRAW1    LDA  ROTZ         ;4 MSBs
         TAX               ;bestimmen, in
         LSR               ;welchen
         LSR               ;Quadranten der
         LSR               ;1.Shapevektor
         LSR               ;zeigt
         STA  QDRNT        ;Ø=Up, 1=Right,
         TXA               ;2=Down, 3=Left
         AND  #$F          ;4 LSB'S =
         TAX               ;Winkel Ø..84.4'
         LDY  COSTAB,X     ;COS als
         STY  COSINUS+1    ;Operand poken
         EOR  #$F          ;SIN(ß) =
         TAX               ;COS(90'-ß)
         LDY  COSTAB+1,X   ;max.SIN-Wert=
         INY               ;SIN(90*15/16')
         STY  SINUS+1      ;=Ø.99 =>$FE+1
         LDA  #RTS_CODE    ;HPLOT als
         STA  CHECK_TO     ;Subroutine
         LDY  #Ø           ;8 MSBs von Y
         STY  Y1+1         ;nur Counter
         STY  Y2+1         ;bei Übertret.
         LDA  (SHAPE),Y    ;d.Bildschirms
         STA  VEKTOR
         AND  #%ØØØØØ1ØØ  ;Bit3 bestimmt:
         BEQ  SETFLAG      ;unsichtbar
         LDY  #%1ØØØØØØØ   ;sichtbar
SETFLAG  STY  PLOTFLAG     ;1.Vktr: Bit6=Ø
         LDY  #Ø           ;vorwärts
         STY  DIRCTION+1   ;zeichnen
DRAW2    LDA  #$8Ø         ;$8Ø = Ø.5 als
         STA  COSVAR       ;Startwert
         STA  SINVAR       ;-> ab Ø.5
         LDX  Y2           ;aufrunden, wie
         LDY  SCALEZ       ;bei HPLOTTO
DRAW3    SEC               ;$1ØØ*COS - 1
         LDA  COSVAR       ;COS(Ø) = $FF
COSINUS  ADC  #DATA8       ;+ 1(C=1) =$1ØØ
         STA  COSVAR       ;weil COSmax=1
         BCC  DRAW4        ;d.h.= $1ØØ
         JSR  LRUD         ;move 1 Step
         CLC               ;max.SIN-Wert<1
DRAW4    LDA  SINVAR       ;d.h. < $1ØØ
SINUS    ADC  #DATA8       ;wenn SINVAR
         STA  SINVAR       ;>$FF, move in
         BCC  DRAW5        ;COS-Richtung
         JSR  LRUD_9Ø      ;+ 9Ø'
DRAW5    DEY               ;move SCALE-mal
         BNE  DRAW3
         STX  Y2

Verbindungspunkt auf Bildschirm?
RTSFLG   LDY  X2
         LDA  Y2+1         ;Y2 < 256?
         BNE  AWAY         ;Y2 >= 256
         CPX  #192         ;Y2 < 192?
         BCS  AWAY         ;Y2 >= 192
         LDA  X2+1         ;Y2 ok.
         CMP  #2
```

```
         BCC  P2_OK        ;X2 < 512
         BNE  AWAY         ;X2 >= 3*256
         CPY  #56Ø-512     ;X2 < 56Ø?
         BCS  AWAY         ;X2 >= 56Ø
P2_OK    BIT  PLOTFLAG     ;Vektor ist?
         BPL  AWAY         ;unsichtbar!
         LDA  Y1+1         ;P1(Ausgangs-
         BNE  CHANGE       ;punkt) ok.?
         LDA  Y1
         CMP  #192
         BCS  CHANGE
         LDA  X1+1
         CMP  #2
         BCC  P1_P2
         BNE  CHANGE
         LDA  X1
         CMP  #56Ø-512
         BCC  P1_P2

P1 außerhalb, P2 innerhalb der Screen;
VOR Plotten P1 & P2 vertauschen, dann
Linie von P1 bis Screenrand mit P1_RAND
CHANGE   LDA  X2+1
         STY  LINNUM       ;für POSITION
         JSR  HPLOT1
         LDY  #Ø           ;Y2+1 = Ø
         STY  Y1+1         ;=> Y1+1 = Ø
         LDY  #%1ØØØØØØØ   ;plot rückwärts
         STY  DIRCTION+1   ;d.h. Vektor um
         LDX  #RTS_CODE    ;18Ø' gedreht
         STX  RTSFLG       ;Programmteile
         STX  NEXT_TO      ;als Subroutine
         JSR  DRAW2        ;benutzen
         STY  DIRCTION+1   ;YREG=Ø,vorwrts
         STY  Y2+1         ;inzwischen
         LDA  #LDY_CODE    ;Y2+1 <Ø / >191
         STA  RTSFLG
         LDA  #LDA_CODE
         STA  NEXT_TO
         LDX  Y1           ;hier:
         LDY  X1           ;P2 = P1
         LDA  X1+1
         STX  Y2
         STY  X2
         STA  X2+1
         JMP  NXTVEKTR
P1 & P2, beide auf Screen, verbinden
P1_P2    JSR  DRAWLINE
         JMP  AWAY1
AWAY     LDA  X2+1         ;Verbindungsp.
         STX  Y1           ;wird neuer
         STY  X1           ;Ausgangspunkt
         STA  X1+1
AWAY1    LDA  Y2+1
         STA  Y1+1
NXTVEKTR LDY  #Ø
         LDA  VEKTOR
         LSR               ;1 Shapevektor
         LSR               ;= 3 Bit
         LSR
         BEQ  NXTSHP
GODRAW2  STA  VEKTOR       ;Bit7 von
         AND  #%ØØØØØ1ØØ  ;PLOTFLAG für
         CMP  #%ØØØØØ1ØØ  ;momentanen
         ROR  PLOTFLAG     ;Vektor, Bit6 =
         JMP  DRAW2        ;letzter Vektor
NXTSHP   INC  SHAPE        ;nächstes
         BNE  OKSHPTBL     ;Shape-Byte
         INC  SHAPE+1      ;holen
OKSHPTBL LDA  (SHAPE),Y
         BNE  GODRAW2      ;Shape zuende?
         LDA  #JSR_CODE    ;eigenständiger
         STA  CHECK_TO     ;Betrieb der
         JMP  EXIT         ;HPLOT-Routinen
Left-, Right-, Up-, Down-Subroutinen
LRUD     CLC               ;normal
LRUD_9Ø  LDA  VEKTOR       ;addiere 9Ø' zu
         ADC  QDRNT        ;1ft,rt,up,down
         AND  #%ØØØØØØ11   ;un/sichtb egal
         CMP  #%ØØØØØØ1Ø   ;U od.D,L od.R
         ROR               ;Up/Dn oder L/R
DIRCTION EOR  #%1ØØØØØØØ   ;kann obiges
         BCS  LFTRT        ;umkehren =
         BMI  DOWN         ;vor/rückwärts
         CPX  #Ø
         BNE  UP1
         JSR  P1_RAND      ;nicht über
         DEC  Y2+1         ;Bildschirmrand
UP1      DEX               ;hinaus plotten!
         RTS
DOWN     CPX  #191
         BNE  DOWN1
         JSR  P1_RAND
```

```
DOWN1   INX
        BNE  RETURN1
        INC  Y2+1
RETURN1 RTS
LFTRT   BPL  RIGHT
        LDA  X2
        BNE  LEFT2
        LDA  X2+1
        BNE  LEFT1
        JSR  P1_RAND
LEFT1   DEC  X2+1
LEFT2   DEC  X2
        RTS
RIGHT   LDA  X2+1
        CMP  #2
        BCC  RIGHT1
        LDA  X2
        CMP  #559-512
        BNE  RIGHT1
        JSR  P1_RAND
RIGHT1  INC  X2
        BNE  RETURN2
        INC  X2+1
        RTS
Zeichnet Linie von P1 bis Screenrand,
entspricht einem Teil der Strecke P1-P2
P1_RAND BIT  PLOTFLAG
        BPL  RETURN2     ;unsichtbar!
        LDA  Y2+1        ;Koordinaten
        BNE  RETURN2     ;wirklich
        CPX  #192        ;auf dem
        BCS  RETURN2     ;Bildschirm?
        LDA  X2+1
        CMP  #2
        BCC  P1_RAND0
        BNE  RETURN2
        LDA  X2
        CMP  #560-512
        BCS  RETURN2
P1_RAND0 LDA  Y1+1
        BNE  RETURN2
        LDA  Y1
        CMP  #192
        BCS  RETURN2
        LDA  X1+1
        CMP  #2
        BCC  P1_RAND1
        BNE  RETURN2
        LDA  X1
        CMP  #560-512
        BCS  RETURN2
P1_RAND1 STY  YSAVE
        STX  Y2
        LDA  #RTS_CODE   ;nach HPLOT TO
        STA  NEXT_TO     ;NICHT
        LDA  X2+1        ;P1neu = P2alt
        JSR  DRAWLINE
        LDA  #LDA_CODE
        STA  NEXT_TO
        LDX  Y2
        LDY  YSAVE
RETURN2 RTS
Zeichnet Linie von P1 bis P2
DRAWLINE BIT  PLOTFLAG   ;voriger Vektor
        BVS  DRAWLIN1    ;sichtbar?
        LDX  Y1          ;nein,
        LDY  X1          ;Ausgangspunkt
        STY  LINNUM      ;P1 muß noch
        LDA  X1+1        ;gesetzt werden
        JSR  HPLOT1
DRAWLIN1 LDA  X2+1
        LDY  X2
        STY  LINNUM
        JSR  HPLOTTO1
        RTS
```

&STORE

zerlegt Double-Hires- in normales Hires-Bild (280*192 Pixel); danach ist linke Hälfte in HGR 1, rechte in HGR 2.

```
STORE   LDX  #191+2
        STX  Y0
STORE1  JSR  NEXTLINE
        LDY  #39         ;40 Bytes/Zeile
ST_LINE LDA  (HIRES1),Y
        STA  HGR1BYT
        LDA  (HIRES2),Y
        STA  HGR2BYT
        LDA  #0          ;1.Normalgrafik
        JSR  MIX         ;Byte = 4 Bits
        LSR  HGR2BYT     ;aus Page2,
        ROR              ;3 Bits a.Pagel
        LSR              ;BIT7 = 0
        STA  BUFFER,Y    ;save in BUFFER
        LDA  #0          ;2.Byte= 3 Bits
        LSR  HGR1BYT     ;aus Seite 2,
        ROR              ;4 Bits aus
        JSR  MIX         ;Seite 1
        LSR              ;Farbbit = 0
        STA  BUFFER+40,Y
        DEY
        BPL  ST_LINE
Erst rechte, dann linke Hälfte des HGR-
Bildes aus BUFFER in HGR2 & HGR1 moven.
        LDX  #39
ST_MOVE1 LDY  #39
ST_MOVE2 LDA  BUFFER+40,X
        STA  (HIRES2),Y
        LDA  BUFFER,X
        DEY
        STA  (HIRES2),Y
        DEX
        DEY
        BPL  ST_MOVE2
        LDA  HIRES1+1    ;erst HGR 2,
        STA  HIRES2+1    ;jetzt HGR 1
        CPX  #$FF        ;80 Bytes aus
        BNE  ST_MOVE1    ;BUFFER
        BEQ  STORE1
Abwechselnd je 3 Bit aus HGR2/1 -> AREG
MIX     LSR  HGR2BYT
        ROR
        LSR  HGR1BYT
        ROR
        LSR  HGR2BYT
        ROR
        LSR  HGR1BYT
        ROR
        LSR  HGR2BYT
        ROR
        LSR  HGR1BYT
        ROR
        RTS
```

&RESTORE

regeneriert aus zwei Bildhälften in HGR 1/2 ein Double-Hires-Bild.

```
RESTORE LDX  #191+2
        STX  Y0
RESTORE1 JSR  NEXTLINE
        PHA              ;HIRES1+1 saven
        LDX  #0
RE_LINE1 LDY  #0
RE_LINE2 LDA  #0
        STA  BUFFER,X
        STA  BUFFER+40,X
        LDA  (HIRES1),Y  ;DHGR2-Byte =
        JSR  MIXBACK     ;4 Bits, DHGR1-
        LSR              ;Byte = 3 Bits
        ROR  BUFFER+40,X ;aus 1.HGRByte
        INY              ;DHGR2-Byte =
        LDA  (HIRES1),Y  ;3, DHGR1-Byte=
        LSR              ;4 Bits aus
        ROR  BUFFER,X    ;2.Normal-HGR-
        JSR  MIXBACK     ;Byte
        LSR  BUFFER+40,X ;Bit7=0, Page2
        SEC              ;Farbbit=1,
        ROR  BUFFER,X    ;Page1
        INX
        INY
        CPY  #40
        BCC  RE_LINE2
        LDA  HIRES2+1    ;erst HGR 1,
        STA  HIRES1+1    ;nun HGR 2
        CPX  #40         ;40*2 Bytes
        BNE  RE_LINE1    ;-> BUFFER
DHGR-Bild aus BUFFER -> HGR 1/2
        PLA              ;gerettet
        STA  HIRES1+1
        LDY  #39
RE_MOVE LDA  BUFFER,Y
        STA  (HIRES1),Y
        LDA  BUFFER+40,Y
        STA  (HIRES2),Y
        DEY
        BPL  RE_MOVE
        BMI  RESTORE1
Je 3 Bit aus AREG -> BUFFER+40/BUFFER
MIXBACK LSR
        ROR  BUFFER+40,X
        LSR
        ROR  BUFFER,X
        LSR
        ROR  BUFFER+40,X
        LSR
        ROR  BUFFER,X
        RTS
```

Hires-Zeile in Y0 um 1 erniedrigen

```
NEXTLINE DEC  Y0         ;XREG= Y-Wert+1
        BEQ  NXTLEXIT    ;wegen "BEQ"
        LDX  Y0
        LDA  YTABL-1,X
        STA  HIRES1
        STA  HIRES2
        LDA  YTABH-1,X
        TAX
        ORA  #%01000000  ;Seite 2
        STA  HIRES2+1
        TXA
        ORA  #%00100000  ;Seite 1
        STA  HIRES1+1
        RTS
NXTLEXIT PLA             ;Return-Adresse
        PLA              ;entfernen
        JMP  EXIT
```

Subroutinen für Screen-Parameter

GETXY holt X/Y-Koordinate aus Basictext prüft Wertebereich. Exit: YREG/LINNUM = Xlow, AREG/LINNUM+1 = Xhigh, XREG = Y

```
GETXY   LDA  #<GETNUM    ;hole 16Bitzahl
        STA  ROUTINE+1   ;-> LINNUM,
        LDA  #>GETNUM    ;prüfe,ob Komma
        STA  ROUTINE+2   ;hole 8Bitwert
        JSR  GOROM       ;-> XREG
Wertebereich (X=0...559; Y=0...191) ok?
        CPX  #192        ;Y <= 191?
        BCS  ILLQUERR    ;Y > 191
        LDY  LINNUM      ;Xlow
        LDA  LINNUM+1    ;Xhigh
        CMP  #2          ;X <=2*256=512?
        BCC  WERTE_OK    ;X < 512
        BNE  ILLQUERR    ;X >= 3*768
        CPY  #560-512    ;X <= 559?
        BCS  ILLQUERR    ;X >= 560
WERTE_OK RTS
```

Entry: LINNUM=Xlow, AREG=Xhigh, XREG=Y
Exit: AREG=BITWERT, XREG=PAGEFLAG, YREG=SPALTE

```
POSITION LSR             ;X = X/2
        TAY              ;gerade X-Werte
        LDA  LINNUM      ;(C=0) liegen
        ROR              ;in HGR2,
        LDX  #%00100000  ;ungerade: HGR1
        BCS  SETPAGE     ;Seite 1
        LDX  #%01000000  ;Seite 2
SETPAGE STX  PAGEFLAG
X=X/7, SPALTE=Quotient; 280/7= 40 Bytes
        CPY  #0
        BEQ  DIV2        ;X < 256
        LDY  #36-1       ;36 =INT(256/7)
        ADC  #256-7*36   ;256 MOD 7
DIV1    INY              ;Quotient=Qot+1
DIV2    SBC  #7          ;Xlow = Xlow/7
        BCS  DIV1
        TAX              ;Rest -7  =
        LDA  BITTAB-$F9,X ;Rest + $F9
        LDX  PAGEFLAG
        RTS
```

&PLOT NORMAL / INVERSE / CLEAR

definiert Plotmodus für &HGR und &HPLOT

```
PLOTMODE LDY  #0
        CMP  #NRMALTOK
        BNE  INVCLR
        LDA  #ORA_CODE   ;NORMAL:
        BNE  SETMODE     ;EOR #0, ORA
INVCLR  CMP  #INVTOK
        BNE  CLEAR
        LDA  #EOR_CODE   ;INVERSE:
        BNE  SETMODE     ;EOR #0, EOR
CLEAR   CMP  #CLEARTOK
        BNE  SYNTERR2
        DEY              ;CLEAR:
        LDA  #AND_CODE   ;EOR #$FF, AND
SETMODE STY  EXORMASK
        STA  PLOT1
        STA  PLOT2
        STA  PLOT3
        STA  PLOT4
```

```
        JSR  CHRGET     ;TXTPTR für
        JMP  EXIT       ;neuen Befehl
-------------------------------------
SYNTERR2 LDX  #<SYNERR   ;Adresse der
        LDY  #>SYNERR   ;Applesoft-
        BNE  PRINTERR   ;Routine
TYPMISMA LDX  #<MISMTCH  ;laden, die
        LDY  #>MISMTCH  ;die Fehler-
        BNE  PRINTERR   ;meldung
ILLQUERR LDX  #<IQERR    ;ausgiebt.
        LDY  #>IQERR
PRINTERR STX  ROUTINE+1  ;Fehlermeldung
        STY  ROUTINE+2  ;ausgeben
        JMP  GOROM
-------------------------------------
&TRACE varnm AT x,y
===================
prüft, ob DHGR-Punkt gesetzt ist; wenn
ja, Variable varnm = 1, sonst varnm = 0
TRACE   LDA  #0          ;String-, Real-
        STA  SUBFLG     ;u. Integer-
        LDA  #<PTRGET   ;Variablen ok.
        STA  ROUTINE+1  ;Adresse & Name
        LDA  #>PTRGET   ;der Variablen
        STA  ROUTINE+2  ;ab TXTPTR
        JSR  GOROM      ;bestimmen.
        STA  VARPOINT   ;Adresse des
        STY  VARPOINT+1 ;3.Var.-Bytes
        LDA  VARNAM     ;Typ feststellen
        BMI  VARTYP     ;INTEGER!
        LDA  VARNAM+1   ;REAL?
        BMI  TYPMISMA   ;STRING verbot.
VARTYP  PHP             ;Typflag
```

```
        JSR  CHRGOT     ;TXTPTR von
        CMP  #ATTOK     ;PTRGET erhöht
        BNE  SYNTERR2
        JSR  CHRGET
        JSR  GETXY
        STX  Y1
        JSR  POSITION
        PHA             ;BITWERT
        LDX  Y1
        LDA  YTABL,X
        STA  PLOT
        LDA  YTABH,X
        ORA  PAGEFLAG
        STA  PLOT+1
        LDX  #0
        PLA
        AND  (PLOT),Y   ;Pixel gesetzt?
        BEQ  NOTSET     ;Flag = 0
        INX             ;Flag = 1
NOTSET  TXA
        PLP             ;REAL/INTEGER?
        BPL  REAL
        LDY  #1           ;High-Byte in
        STA  (VARPOINT),Y ;niedrigerer
        DEY             ;Adresse
        LDA  #0           ;Zahlenwert
        STA  (VARPOINT),Y ;nur 0 oder 1
        JMP  EXIT
REAL    TAY             ;unsigned
        LDA  #<SNGFLT   ;INTEGER in
        STA  ROUTINE+1  ;YREG nach
        LDA  #>SNGFLT   ;FAC schreiben
        STA  ROUTINE+2
```

```
        JSR  GOROM
        LDY  #4         ;FAC nach
VAR=FAC LDA  FAC,Y      ;Variable
        STA  (VARPOINT),Y ;übertragen
        DEY
        BPL  VAR=FAC
        LDY  #1         ;0 und 1 haben
        LDA  FAC+1      ;positives
        AND  #%01111111 ;Vorzeichen
        STA  (VARPOINT),Y
        JMP  EXIT
-------------------------------------
Tabellen
========
Bitpositionen in einem Hires-Byte
BITTAB  DFB  %00000001  ;BIT 0
        DFB  %00000010  ; "  1
        DFB  %00000100  ; "  2
        DFB  %00001000  ; "  3
        DFB  %00010000  ; "  4
        DFB  %00100000  ; "  5
        DFB  %01000000  ; "  6
COS(X*90/16')*$100 - 1; X=0...16:
COSTAB  HEX  FFFEFAF4ECE1D4C5
        HEX  B4A18D7861493118
        HEX  FF
Zwischenspeicher für 80 Bytes
BUFFER  DS   80
        ASC  "M.ORLAMÜNDER 1985"
Y-Koordinaten: 192 HGR-Zeilen
YTABL   DFB  0          ;lower Byte
YTABH   EQU  YTABL+192  ;higher "
PROGEND EQU  YTABH+191
```

DHGR.IIPLUS Hex-Dump

```
BSAVE DHGR.IIPLUS, A$2000,L$870
```

```
$2000: A9 20 85 E6 A9 00 85 9E
$2008: A2 00 A0 00 20 11 F4 A4
$2010: 9E A5 26 99 6A 27 A5 27
$2018: 29 1F 99 2A 28 E6 9E A5
$2020: 9E C9 C0 90 E3 A9 4C 8D
$2028: F5 03 A9 BA 8D F6 03 A9
$2030: 03 8D F7 03 A9 96 85 3C
$2038: A9 20 85 3D A9 BD 85 3E
$2040: A9 20 85 3F A9 A8 85 42
$2048: A9 03 85 43 A0 00 20 2C
$2050: FE A9 BE 85 3C A9 20 85
$2058: 3D A9 E9 85 3E A9 28 85
$2060: 3F A9 0C 85 42 A9 D4 85
$2068: 43 A0 00 2C 89 C0 2C 89
$2070: C0 20 2C FE A9 A8 8D FA
$2078: FF A9 03 8D FB FF A9 AE
$2080: 8D FC FF A9 03 8D FD FF
$2088: A9 B4 8D FE FF A9 03 8D
$2090: FF FF 2C 8A C0 60 2C 8A
$2098: C0 6C FA FF 2C 8A C0 6C
$20A0: FC FF 2C 8A C0 6C FE FF
$20A8: 2C 8B C0 2C 8B C0 4C 3E
$20B0: D4 2C 8A C0 20 FF FF 2C
$20B8: 8B C0 2C 8B C0 60 2C 8A
$20C0: C0 A0 00 84 06 84 50 A2
$20C8: 40 86 51 A2 20 86 07 A9
$20D0: FF 45 D0 11 06 09 80 91
$20D8: 06 A9 FF 45 D0 11 50 29
$20E0: 7F 91 50 C8 D0 E9 E6 07
$20E8: E6 51 CA D0 E2 4C 0C D4
$20F0: A8 20 B1 00 C0 93 F0 64
$20F8: C0 94 D0 03 4C 54 D6 C0
$2100: B4 F0 27 C0 91 F0 BA C0
$2108: 9B D0 03 4C C5 D9 C0 8D
$2110: D0 03 4C 7F D9 C0 A8 D0
$2118: 03 4C 59 D8 C0 AE D0 03
$2120: 4C B5 D8 A2 C9 A0 DE 4C
$2128: BC D9 AD 10 C0 AD 52 C0
$2130: AD 57 C0 AD 50 C0 A2 00
$2138: BD 54 C0 8A 49 01 A2 10
$2140: A0 FC 88 D0 FD CA D0 F8
$2148: AA AD 00 C0 C9 9B D0 E8
$2150: AD 54 C0 AD 51 C0 AD 10
$2158: C0 4C 0C D4 C9 C1 F0 1E
$2160: 20 3E D9 84 08 85 09 86
$2168: 1A 20 5E D9 85 D3 84 D4
$2170: 86 D2 A6 1A 20 3F D6 20
$2178: B7 00 C9 C1 D0 DB 20 B1
$2180: 00 20 3E D9 84 1C 85 1D
$2188: 86 1E 20 5E D9 85 E0 84
$2190: E1 86 D5 8A A6 1E 1D 78
$2198: DB 85 07 BD B8 DA 85 06
$21A0: A5 E0 45 D0 11 06 91 06
$21A8: 38 A5 1E E5 1A A2 7F B0
```

```
$21B0: 06 49 FF 69 01 E8 38 85
$21B8: 30 A5 1C E5 08 85 26 A5
$21C0: 1D E5 09 85 27 A4 1A B0
$21C8: 1E 49 FF 85 27 A5 26 49
$21D0: FF 69 01 85 26 8A 49 80
$21D8: AA A5 E0 85 D3 A5 D5 85
$21E0: D2 A5 E1 85 D4 A4 1E 84
$21E8: 9E 86 9D A9 00 85 9F 85
$21F0: EC 85 ED 85 EE 85 EF A9
$21F8: 80 85 A0 A2 CA 24 9D 30
$2200: 02 A2 E8 A5 27 D0 0F A4
$2208: 30 98 05 26 C9 02 90 4C
$2210: C4 26 90 03 B0 55 18 8E
$2218: 92 D5 A4 26 8C 15 D6 A5
$2220: 27 8D 1A D6 A9 02 E5 26
$2228: 85 26 A9 00 E5 27 85 27
$2230: A5 30 20 F8 D5 18 A5 9F
$2238: 65 EB 85 9F A5 A0 65 EC
$2240: 85 A0 90 01 E8 20 2D D6
$2248: E6 26 D0 E9 E6 27 D0 E5
$2250: A5 D5 85 D2 A4 E1 A6 E0
$2258: 84 D4 86 D3 A5 1C 85 08
$2260: A5 1D 85 09 A5 1E 85 1A
$2268: 4C C5 D4 8E E1 D5 8C 15
$2270: D6 C6 30 A9 00 8D 1A D6
$2278: A5 26 20 F8 D5 A9 90 90
$2280: 02 A9 B0 8D E2 D5 18 A5
$2288: 9F 65 EB 85 9F A5 A0 65
$2290: EC 85 A0 E8 90 0A 20 2D
$2298: D6 C6 30 D0 E9 4C 9E D5
$22A0: 20 3F D6 C6 30 D0 DF 4C
$22A8: 9E D5 A2 10 0A 26 EC 26
$22B0: ED CA D0 F8 85 EB A2 18
$22B8: 18 26 EB 26 EC 26 ED 26
$22C0: EE 26 EF 38 A5 EE E9 FF
$22C8: A8 A5 EF E9 FF 90 04 84
$22D0: EE 85 EF CA D0 E3 26 EB
$22D8: 26 EC A6 9E A4 D4 60 A5
$22E0: D2 49 60 85 D2 0A 10 09
$22E8: 06 D3 10 05 C8 A9 01 85
$22F0: D3 BD B8 DA 85 06 BD 78
$22F8: DB 05 D2 85 07 A5 D3 45
$2300: D0 11 06 91 06 60 A9 F8
$2308: 8D C7 03 A9 E6 8D C8 03
$2310: 20 C3 03 A5 E8 85 E4 A5
$2318: E9 85 E5 8A A2 00 C1 E4
$2320: F0 05 90 03 4C B8 D9 0A
$2328: 90 03 E6 E5 18 A8 B1 E4
$2330: 65 E4 AA C8 B1 E4 65 E9
$2338: 85 E5 86 E4 20 B7 00 C9
$2340: C5 D0 12 20 B1 00 20 3E
$2348: D9 86 1A 84 08 85 09 86
$2350: 1E 84 1C 85 1D A5 F9 AA
$2358: 4A 4A 4A 4A 85 A1 8A 29
$2360: 0F AA BC 46 DA 8C EA D6
$2368: 49 0F AA BC 47 DA C8 8C
$2370: F6 D6 A9 60 8D C5 D4 A0
$2378: 00 84 1B 84 1F B1 E4 85
$2380: EA 29 04 F0 02 A0 80 84
```

```
$2388: E6 A0 00 8C B1 D7 A9 80
$2390: 85 E2 85 E3 A6 1E A4 E7
$2398: 38 A5 E2 69 FF 85 E2 90
$23A0: 04 20 A6 D7 18 A5 E3 69
$23A8: FF 85 E3 90 03 20 A7 D7
$23B0: 88 D0 E5 86 1E A4 1C A5
$23B8: 1F D0 6B E0 C0 B0 67 A5
$23C0: 1D C9 02 90 06 D0 5F C0
$23C8: 30 B0 5B 24 E6 10 57 A5
$23D0: 1B D0 14 A5 1A C9 C0 B0
$23D8: 0E A5 09 C9 02 90 41 D0
$23E0: 06 A5 08 C9 30 90 39 A5
$23E8: 1D 84 50 20 B1 D4 A0 00
$23F0: 84 1B A0 80 8C B1 D7 A2
$23F8: 60 8E 03 D7 8E 9E D5 20
$2400: DC D6 8C B1 D7 84 1F A9
$2408: A4 8D 03 D7 A9 A5 8D 9E
$2410: D5 A6 1A A4 08 A5 09 86
$2418: 1E 84 1C 85 1D 4C 80 D7
$2420: 20 40 D8 4C 7C D7 A5 1D
$2428: 86 1A 84 08 85 09 A5 1F
$2430: 85 1B A0 00 A5 EA 4A 4A
$2438: 4A F0 0B 85 EA 29 04 C9
$2440: 04 66 E6 4C DC D6 E6 E4
$2448: D0 02 E6 E5 B1 E4 D0 EB
$2450: A9 20 8D C5 D4 4C 0C D4
$2458: 18 A5 EA 65 A1 29 03 C9
$2460: 02 6A 49 80 B0 1A 30 0B
$2468: E0 00 D0 05 20 F6 D7 C6
$2470: 1F CA 60 E0 BF D0 03 20
$2478: F6 D7 E8 D0 02 E6 1F 60
$2480: 10 10 A5 1C D0 09 A5 1D
$2488: D0 03 20 F6 D7 C6 1D C6
$2490: 1C 60 A5 1D C9 02 90 09
$2498: A5 1C C9 2F D0 03 20 F6
$24A0: D7 E6 1C D0 4C E6 1D 60
$24A8: 24 E6 10 45 A5 1F D0 41
$24B0: E0 C0 B0 3D A5 1D C9 02
$24B8: 90 08 D0 35 A5 1C C9 30
$24C0: B0 2F A5 1B D0 2B A5 1A
$24C8: C9 C0 B0 25 A5 09 C9 02
$24D0: 90 08 D0 1D A5 08 C9 30
$24D8: B0 17 84 A2 86 1E A9 60
$24E0: 8D 9E D5 A5 1D 20 40 D8
$24E8: A9 A5 8D 9E D5 A6 1E A4
$24F0: A2 60 24 E6 70 0B A6 1A
$24F8: A4 08 84 50 A5 09 20 B1
$2500: D4 A5 1D A4 1C 84 50 20
$2508: D8 D4 60 A2 C1 86 9E 20
$2510: 1E D9 A0 27 B1 06 85 E2
$2518: B1 50 85 E3 A9 00 20 A2
$2520: D8 46 E3 6A 4A 99 57 DA
$2528: A9 00 46 E2 6A 20 A2 D8
$2530: 4A 99 7F DA 88 10 DD A2
$2538: 27 A0 27 BD 7F DA 91 50
$2540: BD 57 DA 88 91 50 CA 88
$2548: 10 F1 A5 07 85 51 E0 FF
$2550: D0 E7 F0 BB 46 E3 6A 46
$2558: E2 6A 46 E3 6A 46 E2 6A
```

```
$2560: 46 E3 6A 46 E2 6A 60 A2
$2568: C1 86 9E 20 1E D9 48 A2
$2570: 00 A0 00 A9 00 9D 57 DA
$2578: 9D 7F DA B1 06 20 05 D9
$2580: 4A 7E 7F DA C8 B1 06 4A
$2588: 7E 57 DA 20 05 D9 5E 7F
$2590: DA 38 7E 57 DA E8 C8 C0
$2598: 28 90 D8 A5 51 85 07 E0
$25A0: 28 D0 CE 68 85 07 A0 27
$25A8: B9 57 DA 91 06 B9 7F DA
$25B0: 91 50 88 10 F3 30 B4 4A
$25B8: 7E 7F DA 4A 7E 57 DA 4A
$25C0: 7E 7F DA 4A 7E 57 DA 4A
$25C8: 7E 7F DA 4A 7E 57 DA 60
$25D0: C6 9E F0 17 A6 9E BD B7
$25D8: DA 85 06 85 50 BD 77 DB
$25E0: AA 09 40 85 51 8A 09 20
$25E8: 85 07 60 68 68 4C 0C D4
$25F0: A9 46 8D C7 03 A9 E7 8D
$25F8: C8 03 20 C3 03 E0 C0 B0
$2600: 69 A4 50 A5 51 C9 02 90
$2608: 06 D0 5F C0 30 B0 5B 60
$2610: 4A A8 A5 50 6A A2 20 B0
$2618: 02 A2 40 86 D1 C0 00 F0
$2620: 05 A0 23 69 04 C8 E9 07
$2628: B0 FB AA BD 46 D9 A6 D1
$2630: 60 A0 00 C9 9D D0 04 A9
$2638: 11 D0 0F C9 9E D0 04 A9
$2640: 51 D0 07 C9 BD D0 17 88
$2648: A9 31 84 D0 8D 21 D4 8D
$2650: 2B D4 8D F2 D4 8D 4F D6
$2658: 20 B1 00 4C 0C D4 A2 C9
$2660: A0 DE D0 0A A2 76 A0 DD
$2668: D0 04 A2 99 A0 E1 8E C7
$2670: 03 8C C8 03 4C C3 03 A9
$2678: 00 85 14 A9 E3 8D C7 03
$2680: A9 DF 8D C8 03 20 C3 03
$2688: 85 08 84 09 A5 81 30 04
$2690: A5 82 30 D0 08 20 B7 00
$2698: C9 C5 D0 C2 20 B1 00 20
$26A0: 3E D9 86 1A 20 5E D9 48
$26A8: A6 1A BD B8 DA 85 06 BD
$26B0: 78 DB 05 D1 85 07 A2 00
$26B8: 68 31 06 F0 01 E8 8A 28
$26C0: 10 0C A0 01 91 08 88 A9
$26C8: 00 91 08 4C 0C D4 A8 A9
$26D0: 01 8D C7 03 A9 E3 8D C8
$26D8: 03 20 C3 03 A0 04 B9 9D
$26E0: 00 91 08 88 10 F8 A0 01
$26E8: A5 9E 29 7F 91 08 4C 0C
$26F0: D4 01 02 04 08 10 20 40
$26F8: FF FE FA F4 EC E1 D4 C5
$2700: B4 A1 8D 78 61 49 31 18
$2708: FF 00 00 00 00 00 00 00
$2710  Hex $00 bis $76F
(Tabelle wird vom Init-Teil
automatisch generiert.)
```

Wordstar-Modifikationen

Teil 2*: Spezielle Druckerfunktionen für Wordstar

von Bernhard Husch

Wenn auch schon viele Beiträge zum Patchen von Wordstar geliefert wurden, so sollen doch noch einige nützliche Hinweise gegeben werden.

1. Die Philosophie von Wordstar

Textverarbeitungsprogramme wie Wordstar sind von ihren Entwicklern so allgemein geschrieben worden, daß sie hardwareunabhängig eingesetzt werden können. Für die große Familie der CP/M-Rechner ist dies der richtige Weg, ein erfolgreiches Softwareprodukt herzustellen; gleichzeitig Möglichkeiten zur Modifikation des angebotenen Produkts bereitzustellen, ist eine weitere große Aufgabe für die Softwarehäuser. Die Gerätevielfalt und die Berücksichtigung der Grenzen mancher Geräte, z.B. begrenzte Kapazitäten im Arbeitsspeicher, zwingen die Softwareanbieter zu gewissen Einschränkungen. Dies zeigt sich insbesondere bei der Textverarbeitung in bezug auf die Ausnutzung der Druckoptionen bei Ausgabegeräten.

2. Kritische Druckfunktionen

Drucker wie der EPSON-FX 80 und kompatible Geräte bieten viel mehr Möglichkeiten zur Gestaltung des Druckbildes, als ein „normal" installierter Wordstar ausnutzen kann. Ein Streifzug durch das Menü der Druckroutine zeigt, daß Wordstar von Haus aus nur wenige Druckoptionen anbietet – zudem entsprechen die angebotenen Optionen nicht mehr dem heute erwünschten Standard (übrigens: in der Version für IBM-PCs und ähnliche Rechner ist dieser Mangel noch nicht behoben). Die Wordstar-Schöpfer gingen von dem Gedanken aus, daß der Anwender über einen „unintelligenten" Drucker verfügt; mittlerweile sind die Ausgabegeräte jedoch mit einer umfangreichen Firmware ausgestattet, die nahezu schreibmaschinenähnliche Druckqualitäten liefert. Besonders kritische Punkte bei der Textverarbeitung sind:

- Doppeldruck
- Boldface-Druck
- Unterstreichung
- Hoch-, Tiefstellung (Potenzierung/Indizierung).

3. Nachteile der Software-Realisierung

Die genannten kritischen Funktionen werden von Wordstar zwar angeboten, jedoch werden sie über die Software realisiert, anstatt die den Druckern eigenen Firmwareroutinen auszunutzen. Das ist kein Wunder, denn die entsprechenden Befehle zur Auslösung der jeweiligen Druckfunktion sind bei Druckern unterschiedlicher Fabrikate ebenfalls recht verschieden. Für die im Vormarsch befindlichen PCs dürfte dieses Problem leichter zu lösen sein, weil sich hier ein gewisser Standard durchzusetzen scheint.

Problematisch ist die Lösung der softwareseitigen Realisierung von Druckoptionen deshalb, weil sie einen erhöhten Zeitaufwand bedingt. Bei einer Textdatei vom Umfang dreier Zeilen mag dies noch nicht ins Gewicht fallen, druckt man jedoch eine Datei in der Größenordnung mehrerer Kilobyte aus, ist der Zeitverlust durch das

```
Beispiele

Normaldruck ist ELITE.
^PA Alternativdruck ist PICA ^PN
Elite
^PY Breitschrift ELITE ^PY
^PY ^PA Breitschrift PICA
^PT1 Subscript ^PT ELITE
^PT0 Superscript ^PT ELITE
^PA PICA einschalten ^PQ Condensed-Schrift ^PW
Nun wieder PICA
^PN Jetzt ELITE
^PE-1 Unterstreichung ^PE-0 Unterstreichung aufgehoben
^PS Unterstreichung ^PS mit WORDSTAR-eigener Funktion
^PEG Doppeldruck ein/aus ^PEH

^PE0 Zeilenabstand auf 1/8" setzen
     Zeilenabstand auf 1/8" setzen
     Zeilenabstand auf 1/8" setzen
     Zeilenabstand auf 1/8" setzen

^PE1 Zeilenabstand auf 7/72" setzen
     Zeilenabstand auf 7/72" setzen
     Zeilenabstand auf 7/72" setzen
     Zeilenabstand auf 7/72" setzen

^PE2 Zeilenabstand auf 1/6" setzen
     Zeilenabstand auf 1/6" setzen
     Zeilenabstand auf 1/6" setzen
     Zeilenabstand auf 1/6" setzen

^PE4 Kursivschrift ein/aus ^PE5

^PA ^PR1 ^PEE  Proportional Fettdruck ein/aus ^PN ^PRO PEF
```

Abb. 1

* In seinem Artikel „Weitere Wordstar-Modifikationen" im Peeker, Heft 3/86, beschrieb D. Conrad bereits Patches für den Epson-Drucker FX-80 mit DDT.

wiederholte Ausdrucken der Zeilen beträchtlich. Besonders bemerkbar macht sich die Zeitkomponente beim Unterstreichen von Textteilen und bei Kombinationen von Doppeldruck/Boldface-Druck und Unterstreichen.

Beschränkte Möglichkeiten bietet Wordstar hinsichtlich der von den Druckern angebotenen Schrifttypen. Aufgrund der wenigen Optionen, die vom Anwender eingebaut werden können, lassen sich bei einer Installation des Textverarbeitungsprogramms nach den Anweisungen des Handbuches nur maximal zwei oder drei unterschiedliche Schrifttypen in den modifizierten Wordstar einbauen.

Der Textprozessor Wordstar liefert die Funktionen „Unterstreichung" und „Doppeldruck", welche von der programmeigenen Software beim Drucken ausgeführt werden. Die Unterstreichung fällt dabei sehr unbefriedigend aus, da jeder gedruckte Buchstabe durch einen Underline-Character (ASCII 5FH) unterstrichen wird, so daß sich keine geschlossene Linie ergibt; Ästheten unter den Textverarbeitern fällt das als Negativum auf. Beim Doppel- (oder Boldface-) Druck wird jede Zeichenfolge entsprechend der bei der Installierung gesetzten Anzahl vom Drucker mehrfach überdruckt, wobei der Druckerkopf jedesmal an den Anfang der Zeichenfolge zurückfährt und dieselbe Zeichenfolge noch einmal schreibt; das kann bei langen Textdateien einen hohen Zeitaufwand beim Ausdrucken mit sich bringen.

4. Drucker-Firmware

Die Firmware des EPSON-Druckers (auch des STAR-Druckers) liefert aber u.a. die beiden Funktionen mit, die bei Wordstar problematisch sind. Die in diesen Drukkern eingebaute Unterstreichungsfunktion arbeitet wesentlich schöner, weil man eine geschlossene Linie unter der zu unterstreichenden Zeichenfolge erhält. Nutzt

Tabelle 1

LABEL	Adresse	Druckbefehl	Codesequenz	Funktion
PSINIT:	Ø6E7	——	Ø7 1B 64 1B 52 Ø2 1B 4D	Initialisierung des Druckers, Auswahl des deutschen Zeichensatzes, Festlegung der Schriftart "Elite" als Standardschriftart.
PSFINI:	Ø6F8	——	Ø2 1B 64	Rückstellung des Druckers in Grundposition nach Druckende.
PSTD:	Ø6BA	↑PA	Ø2 1B 4D	Legt die Standardschriftart auf Elite fest.
PALT:	Ø6B5	↑PN	Ø2 1B 5Ø	Legt Pica als Alternativschrift fest.
ROLUP:	Ø6BF	↑PT<n>	Ø2 1B 53	Leitet den Super- bzw. Subscriptmodus ein (vgl. Beitrag D. Conrad im Peeker 3/86).
ROLDOW:	Ø6C4	↑PT	Ø2 1B 54	Deaktiviert Super- bzw. Subscriptmodus
USR1:	Ø6C9	↑PQ	Ø2 1B ØF	Druckmodus "Condensed" einschalten.
USR2:	Ø6CE	↑PW	Ø1 12	"Condensed" ausschalten.
USR3:	Ø6D3	↑PE	Ø1 1B	Offene ESCAPE-Sequenz zur Nutzung vielfältiger Druckfunktionen.
USR4:	Ø6D8	↑PR<n>	Ø2 1B 7Ø	Proportionaldruck ein- bzw. ausschalten.
RIBBON:	Ø6DD	↑PY	Ø3 1B 57 Ø1	Bereitschaft einschalten.
RIBOFF:	Ø6E2	↑PY	Ø3 1B 57 ØØ	Bereitschaft ausschalten.

man auch die in der Firmware enthaltenen Möglichkeiten zum Fettdruck (oder Boldface-Druck), erhält man einen nicht unerheblichen Geschwindigkeitsgewinn beim Ausdrucken.

5. Anpassung von Wordstar

Die eingeschränkten Möglichkeiten der Wordstar-Modifikationen lassen sich durch die Verwendung der „offenen ESCAPE-Sequenz" umgehen; d.h. an einer der möglichen Patch-Adressen, die dem Anwender zugänglich sind, wird lediglich der Einleitungscode für einen Druckerbefehl (für den EPSON-Drucker ESC – Hex-Code 1B) abgelegt. Dies sollte man beim Wordstar-Label *USR3:* vornehmen: dann kann man dieses offene ESCAPE mit dem Ctrl-Befehl ↑ PE aktivieren. Fügt man nun ein oder mehrere Zeichen an das ↑ PE an, wird der Drucker veranlaßt, seinen Druckmodus entsprechend umzuschalten.

Beispiel: Es soll aus dem Standarddruck in den Kursivdruck umgeschaltet werden. Nach dem EPSON-Handbuch ist die entsprechende Befehlssequenz dafür

```
ESC 4 (HEX 1B 34)
```

Von Wordstar wird dieser Druckerbefehl mit Hilfe der offenen ESCAPE-Sequenz aufgerufen durch

```
↑PE4.
```

Das Abschalten des Kursivdruckes geschieht in ähnlicher Weise durch den Druckerbefehl

```
↑PE5
```

Ein Blick in das Handbuch zum EPSON-Drucker zeigt, daß man eine Vielzahl von Druckeroptionen mit einer offenen ESCAPE-Sequenz verfügbar machen kann; es ist so z.B. auch möglich, unterschiedliche Zeilenabstände im Text zu verwenden.

Um den vollen Umfang der Möglichkeiten des EPSON-Druckers nutzbar zu machen, nahm ich mit Hilfe des Programms „DDT" die in **Tabelle 1** aufgeführten Modifikationen an meiner Version vor. Nun sind folgende Befehlssequenzen für die Druckersteuerung erreichbar:

```
<↑PA> Pica
<↑PN> Elite
<↑PQ> Schmalschrift einschalten
<↑PW> Schmalschrift ausschalten
<↑PE> offene ESC-Sequenz zum Absetzen
      weitreichender Druckeroptionen,
      z.B.:
  <↑PE-1> Unterstreichung ein-,
  <↑PE-Ø> Unterstreichung aus-
          schalten
  <↑PEG>  Fettdruck einschalten
  <↑PEH>  Fettdruck ausschalten
  <↑PE4>  Kursivschrift einschalten
  <↑PE5>  Kursivschrift ausschalten
  <↑PR1>  Proportionalschrift ein-
          schalten
  <↑PRØ>  Proportionalschrift aus-
          schalten
  <↑PTØ>  Hochgestellte Schrift
          einschalten
  <↑PT>   Hochgestellte Schrift
          ausschalten
  <↑PT1>  Tiefgestellte Schrift
          einschalten
  <↑PT>   Tiefgestellte Schrift
          ausschalten
  <↑PY>   Breitschrift einschalten,
          wiederholtes <↑PY> schal-
          tet ab
```

Die Schriftart Elite als Normaldruckmodus wurde deshalb gewählt, weil sie eher ein korrespondenzfähiges Schriftbild ergibt, als dies bei der Schriftart Pica der Fall ist.

D. Conrad wies in seinem Beitrag (Peeker Heft 3/86) bei der Beschreibung des Verfahrens zur Installation der Superscript- und Subscript-Druckfunktionen schon den Weg der offenen ESCAPE-Sequenz: Durch Anfügen der Zeichen „0" und „1" an den Druckbefehl ↑ PT wird der jeweilige Modus ausgewählt. Dieses Vorgehen wurde auch bei der Einrichtung der Möglichkeit des Proportionaldrucks angewandt: Der Druckbefehl

```
↑PR1
```

aktiviert die Proportionalschrift, der Befehl

```
↑PRØ
```

schaltet die Proportionalschrift ab. Zu beachten ist bei der Verwendung dieser Funktionen, daß der Drucker zuvor in die Schriftart PICA geschaltet werden muß (s. Handbuch zum EPSON-Drucker: Mischen der Druckmodi).
Ähnlich verfährt man auch, wenn man die Unterstreichung aktivieren will: Mit der Befehlssequenz

```
↑PE-1
```

wird die Unterstreichungsfunktion aus der Firmware des EPSON-Druckers aufgerufen, mit dem Befehl

```
↑PE-Ø
```

wird sie wieder abgeschaltet.

Schlußbemerkung

Vorsicht ist bei dieser gepatchten Wordstar-Version insofern geboten, als die nach dem < ↑ PE> gesetzten Zeichen als normaler Text interpretiert werden, d.h. sie werden beim Randausgleich mit ausgewertet; hier ist es angeraten, derartige Druckerbefehle erst nachträglich in den Text einzusetzen, damit man einen korrekten Randausgleich bekommt.

PEEKER Börse

Gelegenheitsanzeigen / Kleinanzeigen

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 (nicht gewerblich)

1 Druckzeile à 32 Buchstaben nur DM 5,50 zuzügl. ges. MwSt. Mindestens 2 Druckzeilen

Beispiel:

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

nur DM 18,81 inkl. MwSt.

Musteranzeige gewerblich

1 Druckzeile à 32 Buchstaben nur DM 11,– zuzügl. ges. MwSt. Mindestens 2 Druckzeilen

Beispiel:

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

nur DM 62,70 inkl. MwSt.

PEEKER

Peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

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

☐ suche Hardware ☐ suche Software ☐ Verschiedenes ☐ gewerblich
☐ biete Hardware ☐ biete Software ☐ Chiffre ☐ nicht gewerblich

Bitte den Text mit Schreibmaschine oder in Druckbuchstaben ausfüllen

Jeweils 32 Buchstabe pro Zeile – einschl. Satzzeichen und Wortzwischenräume. Bitte Absender nicht vergessen. Mindestens 2 Zeilen (1 Druckzeile à 32 Buchstaben DM 5,50 nicht gewerblich, DM 11,– gewerblich + MwSt.) – Chiffregebühr DM 6,– + MwSt.

PEEKER

Produkt-Karte

Zu der in **Peeker**, Heft _____, Seite _____ erschienen

Anzeige 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

Umfrage-Karte

Mit dieser Umfrage-Karte können Sie Einfluß auf die redaktionelle Gestaltung des Peeker nehmen. Deshalb Umfrage-Karte möglichst bald zurücksenden. Einsendeschluß: 30. September 1986

Ich besitze einen

☐ Apple-II-Plus
☐ Apple-II-e/c
☐ Mac/Lisa
☐ Apple-II-Plus-Kompatiblen
☐ Apple-II-e/c-Kompatiblen
☐ Sonstigen _____

Ich interessiere mich für den

☐ Apple-II-Nachfolger
☐ IBM-PC
☐ Mac-Nachfolger
☐ Atari-ST

Ich bin

☐ mehr Anwender
☐ mehr Programmierer

Peeker-Börse

Vorname, Name

Firma

Straße

Wohnort

PLZ/Ort

Bitte veröffentlichen Sie den umstehenden Text von _____ Zeilen à _____ DM in der nächsterreichbaren Ausgabe vom **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

Umfrage-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Bitte freimachen

POSTKARTE

Peeker-Börse
Anzeigen-Service

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

Bitte freimachen

POSTKARTE

Inserent

Straße

PLZ/Ort

Bitte freimachen

POSTKARTE

Peeker
Redaktion

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

Produkt-Karte

Wünschen Sie weitere Informationen zu einer der im Heft erschienenen Anzeigen?

Nichts einfacher als das. Produkt-Karte ausfüllen, frankieren und an den Inserenten (nicht an die Peeker-Redaktion) senden.

Bitte 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 und Ihren Absender ein.

PEEKER

Pascal-Kompaktkurs

UCSD-Pascal

Teil 2: Dateiverwaltung, Grafik und Bibliotheken

von Jürgen Geiß

Der erste Teil des Pascal-Kompaktkurses im Peeker, Heft 12/85, S. 33-48, befaßte sich mit den Grundlagen von UCSD- *und* Turbo-Pascal, während sich dieser zweite Teil ausschließlich UCSD-Pascal widmet, weil es in den Bereichen Dateiverwaltung und Grafik beträchtliche Unterschiede zwischen UCSD- und Turbo-Pascal gibt. Ferner sei darauf hingewiesen, daß die UCSD-Datentypen hier nur stichwortartig wiederholt werden, da sie im „Kyan-Pascal-Aufbaukurs" (Heft 3/86, S. 32-44) sowie in dem Aufsatz „Kyan-Pascal und Assembler" (Heft 4/86, S. 48-57) im Abschnitt über den internen Aufbau der Datentypen bereits ausführlich behandelt worden sind.

1. Datentypen

In UCSD-Pascal gibt es folgende elementare Standardtypen, wobei im nachfolgenden links vom „:" eine Variable und rechts deren Typ steht:

I: **INTEGER**;
Vorzeichenbehaftete Ganzzahl, die im Bereich von -32768 bis +32767 liegt. Intern: 2 Bytes (Low Byte – High Byte) als 16-Bit-Zahl mit Bit 15 als Vorzeichen.

C: **CHAR**;
Ein einzelnes Zeichen, das einen ASCII-Wert von 0 bis 255 annehmen kann. Intern: 2 Bytes, wobei das 2. Byte nicht benutzt wird. Ist C vom Typ CHAR, so kann mittels der CHR-Funktion dessen Wert im angegebenen Bereich gesetzt werden. Beispiel:
C := CHR (0);
C := CHR (255);
C := CHR (65);
was äquivalent mit
C := 'A';
ist.

R: **REAL**;
Fließkommazahl, die im Bereich 10↑37 bis 10↑-37 liegt. Intern: 4 Bytes, davon 3 Bytes für Mantisse (= 24-Bit-Mantisse) und 1 Byte (= 8 Bits) für Vorzeichen und Exponent. Aufbau also analog zu Applesoft, das jedoch über ein weiteres Mantissenbyte verfügt (32-Bit-Mantisse). Dies bedeutet praktisch, daß in UCSD-Apple-Pascal maximal 6 Dezimalstellen angezeigt werden (gegenüber maximal 9 Stellen in Applesoft).

S: **STRING**;
Dies ist eine Zeichenkette (intern: Längenbyte + je 1 Byte pro Zeichen) mit dem Vorteil gegenüber Array of Char, daß sie direkt durch den WRITELN-Befehl ausgegeben werden kann, und mit der mehrere Stringoperationen durchgeführt werden können. Die Länge beträgt 80 Zeichen, wenn diese nicht explizit angegeben wurde. Beispiel:
S1: STRING[10]; String mit maximaler Länge 10
S2: STRING[255]; String mit maximaler Länge 255

B: **BOOLEAN**;
Ein Wahrheitswert, der nur die beiden Werte TRUE und FALSE annehmen kann. Intern: 2 Bytes, wobei nur Bit 0 benutzt wird. Dieser Datentyp wird häufig bei Vergleichen oder als Funktionswert benutzt. Beispiel:
B := 5 > 3; {also TRUE}
B := KEYPRESS;
{KEYPRESS ist boolesche Funktion}
B := FALSE;

2. Prozeduren und Funktionen

Auch über sie soll hier nur noch knapp berichtet werden (s. „Kyan-Pascal-Aufbaukurs"). Prozeduren und Funktionen dienen hauptsächlich dem Aufspalten von großen Programmteilen in kleinere überschaubare Module. Sie können aber auch dazu benutzt werden, um speicherplatzsparend zu programmieren, da sie mit verschiedenen Parametern aufgerufen werden können.

2.1. Prozedurdeklarationen

Sie haben die Form:
PROCEDURE <Bezeichner> (<Liste von formalen Parametern>);
{Label-, Konstanten-, Typen-, Variablenvereinbarung wie bei der Programmdeklaration}
BEGIN
{Rumpf}
END;

Dabei ist <Bezeichner> der Name der Prozedur, der beliebig lang sein darf. In UCSD-Pascal sind allerdings nur die ersten acht Zeichen signifikant, d.h daß

```
PROCEDURE LangerName1;
BEGIN
  {...}
END;
und
PROCEDURE LangerName2;
BEGIN
  {...}
END;
```

zu einem Fehler beim Übersetzen führen würde.

Die <Liste von formalen Parametern> kann leer sein, wie im obigen Beispiel zu sehen ist. Ist sie es nicht, dann spaltet sie sich noch in 2 Teile auf, und zwar in die sog. Variablenparameter und die Wertparameter. Kennzeichen von Variablenparametern ist das reservierte Wort VAR vor einem Bezeichner, während Wertparameter nur durch ihre Bezeichner zu erkennen sind. Hinter dem Bezeichner steht nach einem Doppelpunkt der Datentyp. Beispiele:

```
PROCEDURE p1 (VAR x, y: INTEGER);
BEGIN ... END;
PROCEDURE p2 (a, b: CHAR);
BEGIN ... END;
PROCEDURE p3 (VAR x: INTEGER; y: CHAR);
BEGIN ... END;
```

2.2. Prozeduraufruf

Der Aufruf erfolgt durch Angabe des Namens einer Prozedur und, falls bei der Deklaration vorhanden, sog. aktueller Parameter, welche in Klammern angegeben werden müssen.
Beispiel (p1, p2 und p3 wie oben):
Es seien x, y vom Typ INTEGER und c vom Typ CHAR.

```
x := 3;
y := 2;
p1 (x, y);
{hier müssen Variable stehen}
c := 'A';
p2 (c, 'B');
{Variablen oder Konstanten sind erlaubt}
p3 (y, 'H');
{Der erste Parameter muß eine Variable sein,
der zweite kann eine Variable oder Konstante
sein}
```

Ist der formale Parameter ein VAR-Parameter, so wird der aktuelle Parameter nur als ein Zeiger auf sich selbst übergeben. Wird dann im Rumpf dem formalen Parameter ein Wert zugewiesen, so erhält der aktuelle Parameter genau diesen Wert. Beispiel:

```
PROGRAM VarParameter;
VAR I: INTEGER;

PROCEDURE Erhoehe (VAR X: INTEGER);
BEGIN
  X := X + 1;
END;

BEGIN {Hauptprogramm}
  I := 3;
  WRITELN (I);
  Erhoehe (I);
  WRITELN (I)
END.
```

Zuerst wird die Zahl 3 ausgegeben. Beim Aufruf von „Erhoehe" wird im Rumpf dem formalen Parameter X, der einen Zeiger auf die Variable I enthält, der Wert von X + 1 zugewiesen. Dadurch wird intern die Variable I um 1 erhöht, was sich beim darauffolgenden WRITELN (I) durch Ausgabe von 4 bemerkbar macht.
Ist der formale Parameter ein Wertparameter, so wird beim Aufruf der Prozedur der aktuelle Parameter in den formalen Parameter kopiert. Das entspricht dem Anlegen einer lokalen Variablen mit Wertzuweisung. Beispiel:

```
PROCEDURE Value (I: INTEGER);
BEGIN
  I := I + 2;
END;
```

Es sei X := 3. Ein Aufruf von Value (X) kommt folgendem gleich:

```
PROCEDURE Value;
VAR I : INTEGER;
BEGIN
  I := X;
  I := I + 2;
END;
```

2.3. Funktionen

Sie unterscheiden sich von Prozeduren nur darin, daß sie einen Ausgabewert – den Funktionswert – liefern, der beispielsweise einer Variablen zugewiesen werden kann. Dieser Ausgabewert muß als Typ im Kopf angegeben werden. Beispiel:

```
FUNCTION Summe (a, b :INTEGER): INTEGER;
BEGIN
  Summe := a + b;
END;
```

Ist c eine Variable vom Typ INTEGER, so kann die Funktion mit c := Summe (3, 4) aufgerufen werden, wobei der Funktionswert 7 durch die Zuweisung an den Funktionsnamen Summe im Rumpf entsteht. Es ist auch erlaubt
WRITELN (Summe (3, 4));
zu schreiben, wobei der Funktionswert 7 dann gleich an die WRITELN-Prozedur übergeben wird.

3. Dateiverwaltung

In UCSD-Pascal unterscheidet man 4 verschiedene Ebenen der Ein- und Ausgabe. Das sind von unten nach oben:
a) Hardware-orientierte Ein/Ausgabe mittels UNITREAD und UNITWRITE, auf die hier aber nicht näher eingegangen werden soll.
b) Ein/Ausgabe mittels typloser Dateien (BLOCKREAD, BLOCKWRITE).
c) Ein/Ausgabe mittels typgebundener Dateien (GET, PUT und SEEK).
d) Ein/Ausgabe von Textdateien (READ, READLN, WRITE und WRITELN).

3.1. Öffnen und Schließen von Dateien

Um eine neue, nicht vorhandene Datei anzulegen, bietet UCSD-Pascal die Prozedur
REWRITE (FILEVARIABLE, DATEINAME)
an. Sie öffnet eine Datei für Ein- und Ausgabe. DATEINAME kann eine Stringkonstante oder eine Stringvariable sein, die beispielsweise vorher über die Tastatur eingelesen worden ist. Unter diesem Namen wird dann die Datei im Inhaltsverzeichnis der Diskette zu finden sein.

FILEVARIABLE muß als FILE OF <TYP> deklariert werden. Der <TYP> kann beliebig gewählt werden. Beispiele:

```
PROGRAM FILE1;
CONST NAME = 'INTEGERDATEI';
 VAR F : FILE OF INTEGER;
 {Datei, die nur Ganzzahlen enthält}
BEGIN
  REWRITE (F, NAME);
  {...}
END.

PROGRAM FILE2;
VAR F : FILE OF REAL;
      {enthält nur Fließkommazahlen}
    S : STRING;
BEGIN
  WRITE ('Dateiname: ');
  READLN (S);
  REWRITE (F, S);
  {...}
END.
```

Im letzten Beispiel wird ein Dateiname interaktiv während des Programmlaufs erfragt und in die Stringvariable eingelesen. Der nachfolgende REWRITE-Befehl erzeugt dann die Datei mit entsprechendem Namen.
Zu jeder Filevariablen existiert intern eine Puffervariable mit gleichem Namen, auf die mit dem Zeiger-Operator "↑" zugegriffen werden kann. Auf deren Bedeutung wird bei GET und PUT näher eingegangen.

Weiß man, daß eine Datei bereits existiert, so kann sie mittels
RESET (FILEVARIABLE, DATEINAME)
zum Lesen und Schreiben geöffnet werden. FILEVARIABLE und DATEINAME haben die gleiche Bedeutung wie bei REWRITE. Die beiden Beispiele können ebenfalls übernommen werden.
Wird eine Datei mit RESET geöffnet und existiert sie nicht, so wird das laufende Programm unterbrochen und die Fehlermeldung
FILE NOT FOUND
ausgegeben. Durch eine Compileroption kann dies aber verhindert werden und mittels der vordefinierten Funktion IORESULT das Ergebnis selbst ausgelesen werden. Mehr darüber in "Tips und Tricks in Pascal, Teil 4".
Beim Beschreiben einer Datei wird intern ein Zeiger mitgeführt, der auf die nächste zu beschreibende Position zeigt. Manchmal ist es von Nutzen, diesen Zeiger wieder auf die Anfangsposition zu setzen. Dazu existiert die Prozedur
RESET (FILEVARIABLE),
die nur auf bereits geöffnete Dateien angewandt werden kann.
Noch einiges zum RESET-Befehl:
Definiert man beispielsweise eine Filevariable
F : FILE OF INTEGER;
und existiert eine Integerdatei mit Namen "GANZZAHLEN", so wird bei
RESET (F, 'GANZZAHLEN')
die Filevariable F bereits mit der ersten INTEGER-Zahl gefüllt. Diese Zahl kann mittels F↑ angesprochen werden. Beispiel:
WRITELN (F↑);
Intern wird bei jedem RESET automatisch eine GET-Operation durchgeführt, die den ersten Datensatz, im Beispiel einen Integerwert, holt. Einzige Ausnahme bilden Dateien vom Typ INTERACTIVE, deren Verhalten beim Öffnen in 3.2 besprochen werden.

Um Dateien zu schließen, gibt es in UCSD-Pascal die
CLOSE (FILEVARIABLE, OPTION)
Prozedur. FILEVARIABLE muß dabei der Bezeichner einer zuvor geöffneten Datei sein, OPTION kann weggelassen werden, wobei dann NORMAL eingesetzt wird, oder folgende vier Werte annehmen:
NORMAL – normales Schließen der Datei . Eine mit REWRITE geöffnete Datei wird dann aus dem Inhaltsverzeichnis gelöscht.
LOCK – eine mit REWRITE geöffnete Datei wird in das Inhaltsverzeichnis der Diskette eingetragen und verbleibt dort auch nach Beendigung des Programmlaufs.
PURGE – eine zuvor geöffnete Datei wird aus dem Inhaltsverzeichnis der Diskette gelöscht.
CRUNCH – entspricht LOCK, nur daß die Datei an der Stelle, an der der interne Zeiger gerade steht, abgeschnitten wird. Sie verkürzt sich also, wenn der Dateizeiger nicht am Ende steht. Daher der Name CRUNCH. Beispiel:

```
PROGRAM FILE3;
VAR F : FILE OF REAL;
BEGIN
  REWRITE ('REALZAHLEN');
  {...}
  CLOSE (F, LOCK)
 {Permanent im Inhaltsverz. eintragen}
END.
```

3.2. Ende einer Zeile und einer Datei

Liest man Textdateien ein, so möchte man gerne wissen, wann eine Zeile beendet ist. Dazu bedient man sich der booleschen Funktion
EOLN oder
EOLN (FILEVARIABLE).
Im ersten Fall ist die Datei, die eingelesen werden soll, die Tastatur selbst, im zweiten Fall eine Textdatei, die vorher mit RESET (FILEVARIABLE, NAME) geöffnet wurde.

Um zu erkennen, ob eine Datei beim Lesen das Ende erreicht hat, kann die boolesche Funktion
EOF oder
EOF (FILEVARIABLE)
benutzt werden.
Im ersten Fall wird die Tastatur auf Ende der Eingabe geprüft, was dadurch erreicht werden kann, daß <Ctrl-C> eingegeben wird. Beispiel:

```
PROGRAM FILE4;
VAR C : CHAR;
    I : INTEGER;
BEGIN
  I := 0;
  WHILE NOT EOF DO
  BEGIN
    READ (C);
    IF EOLN THEN WRITELN
    ('RETURN eingegeben');
    I := I + 1;
  END;
  WRITELN (I, 'Zeichen eingegeben')
END.
```

Solange das Ende der Eingabe nicht erreicht ist, wird ein Zeichen von der Tastatur gelesen. Das Programm wird nur durch Drücken der Taste <Ctrl-C> beendet. Wird ein Return (ASCII 13) eingegeben, so erhält EOLN den Wert TRUE. Danach wird die Anzahl der eingegebenen Buchstaben auf dem Bildschirm ausgegeben.

Im zweiten Fall wird das Ende einer Datei geprüft, die vorher mit einer Filevariablen geöffnet wurde. Ein Beispiel folgt im nächsten Abschnitt.

3.3. Textdateien

Die READ und WRITE-Prozeduren ermöglichen die einfachste Form der Dateiverwaltung und sind sehr leicht zu programmieren. Ihr Nachteil ist aber, daß sie äußerst langsam sind.

Bevor Ein/Ausgabe von Textdateien stattfinden kann, benötigen wir Variablen vom Typ **TEXT** oder **INTERACTIVE**. TEXT ist vordefiniert und ist eine Abkürzung für FILE OF CHAR, also eine Datei, die nur aus ASCII-Zeichen besteht. Interaktive Dateien sind ebenfalls Textdateien, verhalten sich aber beim Öffnen etwas anders. Bei ihnen wird kein automatisches GET durchgeführt, um den ersten Record bereits bei RESET zu erhalten. Mehr darüber in 3.4 bei GET und PUT.

Die schon bekannten WRITE- und WRITELN-Prozeduren können auch auf Dateien vom Typ TEXT oder INTERACTIVE angewandt werden. Mit ihnen können Einzelzeichen, also Characters, Strings, Integer- und Realzahlen auf Diskette gespeichert werden. Eine Mischung ist ebenfalls möglich. Die so erzeugten Dateien können dann mit dem Editor eingelesen und verändert werden. Die Syntax lautet:
WRITE (FILEVARIABLE, ELEMENT) oder
WRITELN (FILEVARIABLE, ELEMENT)
Dabei muß FILEVARIABLE ein Bezeichner einer Textdatei sein, und ELEMENT muß von einem der o.g. Typen sein. Das folgende Beispiel soll eine Textdatei erstellen, die zu jedem ASCII-Zeichen den zugehörigen Ordnungswert ausgibt. Dabei sollen nur Zeichen, die größer als ' ' (ASCII 32) sind, benutzt werden.

```
PROGRAM ORD_NACH_ASCII;
CONST NAME = 'ORD->ASCII.TEXT';
VAR F : TEXT;
    I : INTEGER;
BEGIN
  REWRITE (F, NAME);
  FOR I := 32 TO 127 DO
    WRITELN (F, I : 3, CHR (I) : 2);
  CLOSE (F, LOCK)
END.
```

Zunächst wird eine Textdatei mit Namen „ORD →ASCII" zum Schreiben geöffnet. Anschließend wird in einer FOR-Schleife jeder Zahlenwert von 32 bis 127 als 3stellige Dezimalzahl und das zugehörige ASCII-Zeichen 2stellig, also mit führendem Leerzeichen, auf Diskette geschrieben. Schließlich wird die so erzeugte Datei permanent im Inhaltsverzeichnis festgehalten. Sie kann nun in den PASCAL-Editor eingelesen werden.
Tip: Setzen Sie die Konstante NAME auf „CONSOLE:" oder „PRINTER:" und beobachten Sie das Ergebnis.

Um mit WRITE und WRITELN erzeugte Dateien wieder einlesen zu können, können die Prozeduren
READ (FILEVARIABLE, VARIABLE) oder
READLN (FILEVARIABLE, VARIABLE)
benutzt werden. FILEVARIABLE und VARIABLE haben die gleiche Bedeutung wie oben.
Im Beispiel soll die Datei „SYSTEM.WRK.TEXT", also das folgende Programm, das sich auf der Boot-Diskette befindet, eingelesen und auf dem Bildschirm ausgegeben werden.

```
PROGRAM LIES_SYNTAX;
VAR F : TEXT;
    C : CHAR;
BEGIN
  RESET (F, '*SYSTEM.WRK.TEXT');
  WHILE NOT EOF (F) DO
  BEGIN
    READ (F, C);
    WRITE (C);
    IF EOLN (F) THEN WRITELN
  END;
  CLOSE (F)
END.
```

Nach Öffnen der Datei auf der Boot-Diskette (daher der „*") und Test auf Ende wird ein einzelnes Zeichen eingelesen und auf dem Bildschirm ausgegeben. War das eingelesene Zeichen ein <Return>, so wird es intern in ein Leerzeichen umgewandelt und EOLN auf TRUE gesetzt. Daher muß der Zeilentrenner mittels WRITELN-Befehl selbst ausgegeben werden. Das Programm benötigt circa 6.48 Sekunden.
Eine schnellere Version von LIES_SYNTAX würde eine ganze Zeile auf einmal einlesen.

```
PROGRAM SCHNELLER;
VAR F : TEXT;
    S : STRING;
BEGIN
  RESET (F, '*SYSTEM.WRK.TEXT');
  WHILE NOT EOF (F) DO
  BEGIN
    READLN (F, S);
    WRITELN (S)
    {IF EOLN (F) THEN WRITELN}
  END
END.
```

Hier spart man sich außer Zeit (4.2 Sekunden) auch den Test auf Zeilenende (EOLN).

3.4. Typgebundene Dateien

Damit sind Dateien gemeint, die einen einzigen festen Datentyp enthalten. Dieser Datentyp kann aber beliebig komplex sein, wie z.B. ein RECORD, in dem selbst wieder RECORDS und ARRAYS vorkommen können. Um Elemente dieses Datentyps auf Diskette zu schreiben oder zu lesen, gibt es die Prozeduren
PUT (FILEVARIABLE) und
GET (FILEVARIABLE).
Des weiteren ist es möglich, auf einen beliebigen RECORD in einer Datei zuzugreifen. Dazu bedient man sich des Befehls
SEEK (FILEVARIABLE, RECORDNUMMER)

Ein einfaches Beispiel: Es sollen die 10 Ganzzahlen von 1 bis 10 rückwärts auf Diskette geschrieben werden und mit einem zweiten Programm wieder eingelesen werden.

```
PROGRAM SCHREIBEN;
VAR F : FILE OF INTEGER;
    I : INTEGER;
BEGIN
  REWRITE (F, 'ZAHLEN');
  FOR I := 10 DOWNTO 1 DO
  BEGIN
    F↑ := I;
    PUT (F)
  END;
  CLOSE (F, LOCK)
END.
```

Die zu beschreibende Datei wird mit REWRITE geöffnet. In der nachfolgenden Schleife werden die Zahlen 10 bis 1 auf Diskette geschrieben. Das geschieht dadurch, daß der zur Filevariablen F gehörigen Puffervariablen F↑ das I zugewiesen wird. Mit PUT (F) wird dann der Datensatz, der aus einer Ganzzahl besteht, auf Diskette geschrieben. Schließlich wird die Datei ordnungsgemäß geschlossen. Ein Versuch, sie in den Editor einzulesen, würde fehlschlagen, da die Ganzzahlen nicht als ASCII-Zeichen wie beim WRITE-Befehl, sondern als echte 16-Bit-Zahlen abgespeichert werden.
Das entsprechende Gegenstück zu SCHREIBEN wäre:

```
PROGRAM LESEN;
VAR F : FILE OF INTEGER;
BEGIN
  RESET (F, 'ZAHLEN');
  WHILE NOT EOF (F) DO
  BEGIN
    WRITELN (F↑);
    GET (F)
  END;
  CLOSE (F, NORMAL)
END.
```

Hier wird der Vorgang bei einem RESET noch einmal klar: Die Puffervariable F↑ wird schon mit der ersten Ganzzahl gefüllt, die sofort mit WRITELN (F↑) ausgegeben werden kann. Erst danach erfolgt ein GET (F), um die nächste Zahl zu bekommen. Ist die Datei ausgelesen, wird bei GET (F) das Dateiende durch EOF (F) angezeigt, was den Abbruch der WHILE-Schleife bewirkt. Das Schließen der Datei kann mit der Option NORMAL geschehen, da sie sich ja schon auf Diskette befindet und nicht mehr geLOCKed werden muß. Es wäre auch ein CLOSE (F, PURGE) möglich, um die Datei beim Programmende aus dem Inhaltsverzeichnis zu löschen.
Um auch komplexere Dateien zu handhaben, benutzen wir den SEEK-Befehl. Es soll eine Adreßliste mit Namen und Wohnort erzeugt und auf Diskette gerettet werden. Die zugehörige Datenstruktur ist ein RECORD.

```
PROGRAM ADRESSEN;
TYPE EINTRAG = RECORD
                 NAME    : STRING;
                 STRASSE : STRING;
                 PLZ     : INTEGER;
                 ORT     : STRING
               END;
VAR ADRESSE : EINTRAG;
    DATEI   : FILE OF EINTRAG;
    REC     : INTEGER;
BEGIN
  REWRITE (DATEI, 'ADRESSEN');
  WITH ADRESSE DO
    REPEAT
      WRITE ('NAME :');
      READLN (NAME);
      WRITE ('STRASSE :');
      READLN (STRASSE);
      WRITE ('PLZ: ');
      READLN (PLZ);
      WRITE ('ORT: ');
      READLN (ORT);
      IF NAME <> '' THEN
      BEGIN
        DATEI↑ := ADRESSE;
        PUT (DATEI);
      END {IF}
    UNTIL NAME = '';

  WRITE ('WELCHE RECORDNUMMER? ');
  READLN (REC);
  SEEK (DATEI, REC - 1);
  GET (DATEI);
  IF NOT EOF (DATEI) THEN
    WITH DATEI↑ DO
    BEGIN
      WRITELN (NAME);
      WRITELN (STRASSE);
      WRITE (PLZ, ' ');
      WRITELN (ORT)
    END; {WITH, IF}
  CLOSE (DATEI, LOCK);
END.
```

Die Variable ADRESSE dient als Hilfsvariable zum Eingeben der Adressen. Die Filevariable Datei muß als FILE OF EINTRAG erklärt werden. Dadurch weiß das Pascal-System, wie groß ein Record-Element ist. Nach dem Starten des Programms können beliebig viele Adressen eingegeben werden. Das Ende wird durch einen Leerstring im Namen bewirkt (REPEAT...UNTIL NAME = ''), wobei dieser Eintrag nicht mehr auf Diskette geschrieben wird. Nehmen wir an, wir hätten 3 Adressen eingegeben. Dann könnte auf die Frage „WELCHE RECORDNUMMER?" mit einer Zahl zwischen 1 und 3 geantwortet werden. Da die Zählung der Records auf Diskette bei 0 beginnen, muß SEEK mit einem um 1 verminderten Wert aufgerufen werden. Das anschließend GET holt den gewünschten Record, falls das Dateiende nicht erreicht wurde, und gibt die Einzelkomponenten auf dem Bildschirm aus.

3.5. Typlose Dateien

Alle Ein- und Ausgabemöglichkeiten, die in 3.2 und 3.3 besprochen wurden, können einen Softwareentwickler nicht zufriedenstellen, da einerseits die Datenstrukturen genau festgelegt sein müssen und, was noch viel wichtiger ist, die Verarbeitungsgeschwindigkeit im Schneckentempo vor sich geht. Aus diesen Gründen gibt es typlose Dateien, mit denen alle Daten einer Diskette eingelesen, verarbeitet und wieder beschrieben werden können. Dabei spielt die Blockstruktur der Diskette eine wichtige Rolle.
Eine normale Apple-Diskette mit 140K hat in Pascal-Format 280 Blöcke zu 512 Bytes. Jede Datei beginnt an einer Blockgrenze, wobei Block 0 einer Datei der erste Block ist und die Blöcke fortlaufend durchnumeriert werden. Mit den Funktionen
BLOCKREAD (FILEVARIABLE, VARIABLE, BLOCKS, RELATIVBLK) und
BLOCKWRITE (FILEVARIABLE, VARIABLE, BLOCKS, RELATIVBLK)
ist es möglich, die ganze Datei mit einem Schlag zu lesen und zu beschreiben, was beim Lesen zu einer Übertragungsrate von 16K/Sekunde und beim Schreiben von 8K/Sekunde führt. Der Funktionswert ist die Anzahl der übertragenen Blöcke.
Die Filevariable muß vom Typ FILE sein. Die VARIABLE kann beispielsweise ein Array sein. BLOCKS ist die Anzahl der Blöcke, die gelesen oder geschrieben werden sollen, und RELATIVBLK ist der relative Block, bei dem die Übertragung beginnen soll. Der letzte Parameter kann auch fehlen, wobei bei Block 0 begonnen wird. Als Beispiel soll die Datei „SYSTEM.SYNTAX" von der Boot-Diskette in eine Datei mit anderem Namen kopiert werden.

```
PROGRAM COPY;
VAR BLOECKE : INTEGER;
    F       : FILE;
    PUFFER  : PACKED ARRAY [Ø..16383]
              OF Ø..255;
    S       : STRING;
BEGIN
  RESET (F, '*SYSTEM.SYNTAX');
  BLOECKE := BLOCKREAD (F, PUFFER, 32);
  IF BLOECKE = Ø THEN
  BEGIN
    WRITELN ('FEHLER BEIM LESEN');
    EXIT (PROGRAM)
  END;
  CLOSE (F);
  WRITE ('Name der Kopie: ');
  READLN (S);
  REWRITE (F, S);
  IF BLOCKWRITE (F, PUFFER, BLOECKE)
   <> BLOECKE THEN
  BEGIN
    WRITELN ('FEHLER BEIM SCHREIBEN');
    EXIT (PROGRAM)
  END;
  CLOSE (F, LOCK);
  WRITELN ('ENDE')
END.
```

BLOECKE wird benutzt, um das Funktionsresultat von BLOCKREAD, nämlich die aktuelle Anzahl von Blöcken zu merken. Ist sie gleich Null, so konnte nichts gelesen werden. Bei BLOCKREAD (F, PUFFER, 32) werden in den Puffer maximal 32 Blöcke zu 512 Bytes eingelesen. Da SYSTEM.SYNTAX aber kürzer ist, wird als Funktionsresultat von BLOCKREAD die aktuelle Länge zurückgegeben. Anschließend wird nach dem Namen der Kopie gefragt und die Datei mit REWRITE eröffnet. Mit BLOCKWRITE (F, PUFFER, BLOECKE) wird genau die oben eingelesene Anzahl von Blöcken wieder geschrieben. War die Diskette voll oder defekt, so ist das Funktionsresultat ungleich BLOECKE, was zu einem Fehlerausgang führt. Schließlich wird die Datei geLOCKed.
Mit BLOCKREAD und BLOCKWRITE können auch ARRAYS von Ganzzahlen oder ARRAYS von RECORDS schnell von und zur Diskette transferiert werden. Dazu muß nur die Deklaration des Puffers geändert werden.

Drucker: Da Anfänger meist Probleme mit der Druckausgabe (Datei „PRINTER:") haben, soll hier noch einmal kurz gezeigt werden, wie der Drucker angesprochen werden kann. Um die Ausgabe auf den Drucker umzulenken, muß eine Filevariable vom Typ „text" oder „interactive" deklariert werden und mit dieser der Drukker geöffnet werden.

Beispiel:

```
PROGRAM AUSGABE;
CONST
  S = 'PRINTER';
 {S = '#6:' geht auch}
VAR F : TEXT;
BEGIN
  REWRITE (F, S);
  WRITELN ('Ausgabe auf Bildschirm');
  WRITELN (F, 'Ausgabe auf Drucker');
  CLOSE (F, LOCK);
END.
```

4. Hochauflösende Grafik

In UCSD-Pascal existieren keinerlei Grafik-Befehle, wie etwa in Applesoft-BASIC. Erst eine spezielle Unit aus der SYSTEM.LIBRARY, die Turtlegraphics-Unit, machen den Apple auch in Pascal grafikfähig. Die Grafikbefehle ähneln teilweise denen aus Logo, da auch hier von einer Schildkröte (Turtle) ausgegangen wird, die einen Zeichenstift mit sich führt und die sich drehen oder eine bestimmte Anzahl von Schritten wandern kann. Die Unit enthält einige Grundbefehle zum Zeichnen von Linien, zum Beschriften von Zeichnungen, Setzen von Farben und einiges mehr. Die doppelt hochauflösende Grafik des Apple IIe, der mit einer erweiterten 80-Zeichenkarte versehen ist, oder eines Apple IIc ist nicht möglich. Inzwischen existiert jedoch von meinem Bruder und mir eine neue Turtlegraphics-Unit, die nicht nur wesentlich schneller als die alte ist, sondern auch eine Fülle von Prozeduren bietet, mit denen der Grafikbildschirm manipuliert werden kann. Die doppelt hochauflösende Grafik wird genauso unterstützt wie das Programmieren von Fenstern, Pull-down-Menüs und die Verwendung der Maus. Sie ist beim Hüthig-Software-Service erhältlich.

4.1. Initialisierungen

Um die in der Library enthaltenen Prozeduren und Funktionen benutzen zu können, muß direkt nach dem Programmkopf ein
USES TURTLEGRAPHICS;
geschrieben werden. Der Compiler holt sich beim Übersetzen dann alle Informationen aus der Unit in der Library.
Der Apple-Grafikbildschirm ist eine Matrix, bestehend aus 280 Punkten in der Horizontalen und 192 Punkten in der Vertikalen. Diese Punkte sind beginnend mit (X = 0, Y = 0) in der linken unteren Ecke bis (X = 279, Y = 191) in der rechten oberen Ecke numeriert. Dies entspricht einem kartesischen Koordinatensystem im Gegensatz zur Numerierung in Applesoft-BASIC, bei der ja links oben begonnen wird.

Um den Grafikbildschirm zu initialisieren, sollte zu Beginn eines Programms die Prozedur
INITTURTLE
aufgerufen werden. Sie wirkt ähnlich wie HGR in BASIC. Der Bildschirm wird gelöscht und von Text auf Grafik umgeschaltet, die Farbe wird auf unsichtbar und der Bildausschnitt auf volle Größe gesetzt.

Manchmal ist es sinnvoll, zwischen Text- und Grafikschirm hin und her zu schalten. Die Prozeduren
GRAFMODE und
TEXTMODE
bewirken dies. Ein Umschalten von Grafik auf Text zerstört nicht den Inhalt der Grafikseite und umgekehrt.

4.2. Bildausschnitte

Über den Bildausschnitt wurde schon gesprochen. Es ist möglich, nur einen Teil des Grafikschirms zu benutzen, ohne daß Änderungen im anderen Teil stattfinden. Damit können unsichtbare Fenster gesetzt werden. Bei der Prozedur
VIEWPORT (Lrand, Rrand, Urand, Orand)
müssen die vier Parameter ein Rechteck irgendwo im gesamten Grafikschirm definieren. Beispiel:
VIEWPORT (0, 20, 10, 30)
definiert ein Quadrat der Seitenlänge 21 Bildpunkte mit den Eckpunkten (X = 0, Y = 10) bis (X = 20, Y = 30).
Alle Grafikoperationen – z.B. das Zeichnen von Linien – würden nur Änderungen des Grafikschirms im definierten VIEWPORT zur Folge haben. Alles, was außerhalb läge, würde abgeschnitten werden. Damit können Bildschirmbereiche vor versehentlichem Überschreiben geschützt werden.
Um wieder den ganzen Bildschirm zu benutzen, muß
VIEWPORT (0, 279, 0, 191)
benutzt werden.

4.3. Farbe

Um auch Farbe auf den Bildschirm zu bringen, gibt es einen vordefinierten Typ von der Form
TYPE SCREENCOLOR (NONE, WHITE, BLACK, REVERSE, RADAR, BLACK1, GREEN, VIOLET, WHITE1, BLACK2, ORANGE, BLUE, WHITE2);
Damit kann eine Variable
VAR COLOR : SCREENCOLOR
alle Werte des oben beschriebenen Aufzählungstyps annehmen. Die Farbe REVERSE hat eine besondere Bedeutung. Sie invertiert den Bildschirm an der entsprechenden Stelle. Aus WHITE wird BLACK, aus GREEN wird ORANGE usw.
Um dem Zeichenstift der Schildkröte eine andere Farbe zu verleihen, muß die Prozedur
PENCOLOR (FARBE)
benutzt werden. Dabei kann FARBE einen Wert des Typs SCREENCOLOR besitzen. Beispiel:
PENCOLOR (NONE)
PENCOLOR (GREEN)
Es scheint merkwürdig, daß es mehrere WHITE- und BLACK-Farben gibt. Das liegt aber am Aufbau der Apple-Farbgrafik. So müssen WHITE1 und BLACK1 zusammen mit GREEN und VIOLET und WHITE2 und BLACK2 mit ORANGE und BLUE benutzt werden. Bei einem Schwarz-Weiß-Monitor sollte man nur WHITE und BLACK benutzen, da so die feinste Grafik erzeugt werden kann.
Um den gesamten Bildschirm mit einer Farbe zu füllen, existiert die Prozedur
FILLSCREEN (FARBE)
Sie wirkt immer auf den eingestellten Bildausschnitt. FARBE hat dabei die gleiche Bedeutung wie bei PENCOLOR. Im nachfolgenden Beispiel soll das Zusammenspiel von SCREENCOLOR, VIEWPORT und FILLSCREEN gezeigt werden. Das Programm erzeugt schrumpfende Bildausschnitte mit verschiedenen Farben.

```
PROGRAM SCHRUMPF;
USES TURTLEGRAPHICS;
CONST XMAX  = 279;
      YMAX  = 191;
VAR   FARBE : SCREENCOLOR;
      I     : INTEGER;
BEGIN
  I := Ø;
  INITTURTLE;
  FOR FARBE := NONE TO WHITE2 DO
  BEGIN
    VIEWPORT (I, XMAX - I, I, YMAX - I);
    FILLSCREEN (FARBE);
    I := I + 7
  END;
  READLN {Zeit zum Anschauen}
END.
```

In der FOR-Schleife werden nacheinander alle möglichen Farben ausgewählt. Der VIEWPORT-Aufruf läßt den Bildausschnitt kontinuierlich um 2 * 7 Punkte schrumpfen. Das geschrumpfte Rechteck wird dann durch FILLSCREEN mit der entsprechenden Farbe gefüllt. Zum Beenden des Programms muß ein Return eingegeben werden. Ohne READLN würde die Kommandozeile des Pascal-Systems sofort nach Ablauf des Programms erscheinen, wobei vorher in den Textmodus geschaltet und die Grafik zerstört würde.

4.4. Turtle-Bewegungen

Wie schon zuvor erwähnt, existiert auf dem Grafikschirm eine (unsichtbare) Schildkröte, die einen Zeichenstift mit sich führt. Um zu verhindern, daß der Zeichenstift auf dem „Papier" aufsitzt und beim Bewegen der Kröte eine Linie erzeugt, muß PENCOLOR auf NONE gesetzt werden. Um die Schildkröte um einige Bildpunkte vorwärts zu bewegen, wird
MOVE (ENTFERNUNG)
benutzt. Beispiel:
MOVE (60)
bewegt die Schildkröte um 60 Bildpunkte in die Richtung, in die ihre „Nase" zeigt. Die Nase zeigt nach Aufruf von INITTURTLE nach rechts, was einem Winkel von 0 Grad entspricht. Wie in einem kartesischen Koordinatensystem entspricht ein Winkel von 90 Grad einer Richtung nach Norden, 180 Grad nach Westen und 270 Grad nach Süden. Um die Schildkröte in eine beliebige Richtung zu drehen, benutzt man
TURNTO (WINKLEL) oder
TURN (WINKEL)
TURNTO dreht die Schildkröte absolut in die Richtung WINKEL, während TURN eine relative Drehung vom derzeitigen Winkel aus erzeugt. Alle Winkelangaben müssen in Grad gemacht werden. Beispiele:
TURNTO (90) {Turtle zeigt nach oben}
TURNTO (-90) {Turtle zeigt nach unten}
TURN (200) {Turtle dreht sich um 200 Grad nach links}
TURN (-80) {Turtle dreht sich um 80 Grad nach rechts}
Ein kurzes Programm soll ein Rechteck mit selbst eingegebener Seitenlänge zeichnen.

```
PROGRAM RECHTECK;
USES TURTLEGRAPHICS;
VAR SEITE : INTEGER;
BEGIN
  WRITE ('Seitenlänge: ');
  READLN (SEITE);
  INITTURTLE;
  PENCOLOR (WHITE);
  FOR I := 1 TO 4 DO
  BEGIN
    MOVE (SEITE);
    TURN (9Ø)
  END;
  READLN
END.
```

Mit MOVE bewegt sich die Schildkröte um soviele Bildpunkte in die eingestellte Richtung, wie SEITE enthält. Anschließend wird um 90 Grad nach links gedreht.

Um Ablesen zu können, an welcher Position oder unter welchem Winkel die Schildkröte sich befindet, können folgende Funktionen benutzt werden:
TURTLEX
TURTLEY
TURTLEANG
TURTLEX und TURTLEY geben die X- bzw. die Y-Koordinate der augenblicklichen Schildkrötenposition an.
TURTLEANG ist eine Funktion, die den Winkel, unter dem die Schildkröte steht, angibt. Beispiel:

```
MOVETO (1ØØ, 8Ø);
TURNTO (12Ø);
TEXTMODE;
WRITELN (TURTLEX: 4, TURTLEY: 4,
TURTLEANG: 4);
{Die Ausgabe wäre:  1ØØ  8Ø 12Ø.}
```

Es existiert noch eine Turtlegraphics-Funktion, die zwar nicht direkt hier hineingehört, aber der Vollständigkeit halber erklärt werden soll. Oft ist es notwendig zu wissen, ob ein Bildschirmpunkt gesetzt ist oder nicht. Die boolesche Funktion
SCREENBIT (X, Y)
gibt TRUE zurück, wenn der Bildschirm im Punkt (X, Y) nicht schwarz war, ansonsten erfolgt die Antwort FALSE.

Derjenige, dem die Bewegungen mit der Schildkröte nicht behagen und der lieber wie in Applesoft programmieren möchte, kann sich der Prozedur
MOVETO (X, Y)
bedienen. Damit bewegt sich die Schildkröte an die absolute Position (X, Y). Ein BASIC-ähnlicher Befehl wie HPLOT könnte leicht wie folgt implementiert werden:

```
PROCEDURE HPLOT (X1, Y1, X2, Y2: INTEGER);
BEGIN
  PENCOLOR (NONE);
  MOVETO (X1, Y1);
  PENCOLOR (WHITE);
  MOVETO (X2, Y2)
END;
```

An die Prozedur werden 2 Punktepaare übergeben, die die Eckpunkte einer Linie darstellen sollen. Bevor die Schildkröte an die absolute Position von (X1, Y1) bewegt werden soll, muß die Farbe auf NONE gestellt werden, um zu verhindern, daß eine Linie vom augenblicklichen Standpunkt aus gezeichnet wird. Anschließend wird zum Startpunkt geMOVEd, die Farbe auf WHITE gesetzt und zum Endpunkt geMOVEd.

4.5. Text im Grafikschirm

Um Zeichnungen auch beschriften zu können, werden 3 Prozeduren angeboten.
WCHAR (ZEICHEN) und
WSTRING (ZEICHENKETTE)
geben an der aktuellen Turtleposition ein einzelnes Zeichen bzw. eine ganze Zeichenkette (STRING) aus. Beispiele:

```
WCHAR ('a');

VAR S : STRING;
TEXTMODE;
READLN (S);
GRAFMODE;
MOVETO (2Ø, 3Ø);
WSTRING (S);
```

Die Prozedur
CHARTYPE (MODUS)
bestimmt, *wie* Zeichen auf den Bildschirm kommen. Dabei kann Modus einen Wert zwischen 0 und 15 annehmen. Im folgenden sei C das Zeichen, das ausgegeben werden soll, und S die Stelle des Bildschirms unter dem Zeichen. Dann hat MODUS den folgenden Effekt:

MODUS 0: Die Zeichenmatrix (7 x 8 Bit) wird als schwarzer Block gesetzt.
MODUS 1: C NOR S, invertiertes (C ODER S)
MODUS 2: C AND NOT S, UND von C mit invertiertem Inhalt des Schirms
MODUS 3: NOT S, invertiert die Fläche auf dem Schirm
MODUS 4: NOT C AND S, invertiertes C UND Schirm
MODUS 5: NOT C, invertiertes C auf Schirm kopieren
MODUS 6: C XOR S, ausschließliches ODER von C und S
MODUS 7: C NAND S, invertiertes (C UND S)
MODUS 8: C AND S, C UND Schirm
MODUS 9: C = S, Äquivalenz von C und S (inverses XOR)
MODUS 10: C, kopiert C auf den Bildschirm
MODUS 11: C OR NOT S, C ODER invertierter Schirm
MODUS 12: S, Schirm wird durch Schirm ersetzt, also passiert nichts.
MODUS 13: NOT C ODER S, ODER von invertiertem C und Schirm
MODUS 14: C ODER S
MODUS 15: füllt die Zeichenmatrix mit Weiß (Gegenteil von MODE 0)

Ein Beispiel:

```
PROGRAM MODETEST;
USES TURTLEGRAPHICS;
BEGIN
  INITTURTLE;
  CHARTYPE (6); {exklusives oder}
  WCHAR ('A');
  READLN;
  WCHAR ('A');
  READLN;
END.
```

Durch Setzen des Chartypes auf 6 wird bewirkt, daß beim ersten Setzen des Buchstabens „A" auf dem noch schwarzen Bildschirm dieser wie erwartet erscheint. Beim zweiten Setzen werden genau die Bits wieder invertiert, die beim Buchstaben „A" gesetzt sind. Also verschwindet das A wieder. Mit Modus 6 ist es also möglich, verschiedene Elemente auf dem Schirm übereinander zu legen und wieder zu löschen, ohne genau wissen zu müssen, welches Bit jetzt gelöscht werden muß und welches nicht.

4.6. Zeichnen von selbst definierten Objekten

Bisher konnten wir nur mit Hilfe der Schildkröte Linien ziehen oder Teile des Bildschirms mit FILLSCREEN füllen. Für viele Anwendungen reicht dies jedoch nicht aus. Man möchte z.B. eigene Muster, Piktogramme oder einen Maus-Pfeil wie beim Macintosh erstellen. In Applesoft gibt es etwas Ähnliches, die sog. Shapes. In UCSD-Pascal gibt es Blöcke, die als zweidimensionaler PACKED ARRAY OF BOOLEAN angesehen werden können. Eine Deklaration von

```
CONST XMAX = 4;
      YMAX = 3;
VAR BLOCK : PACKED ARRAY
 [1..YMAX, 1..XMAX] OF BOOLEAN;
```

erzeugt die Variable BLOCK, die als zweidimensionale Matrix von 3 Zeilen zu 4 Spalten angesehen werden kann. Im Bild sieht das so aus:

```
  X:
Y:* * * *
  * * * *
  * * * *
```

Jedes Element dieser Matrix kann auf TRUE oder FALSE gesetzt werden. Durch die Prozedur DRAWBLOCK kann diese Matrix auf dem Grafikschirm als Bitmuster ausgegeben werden, d.h. an den Stellen des Bildschirms, an denen das Matrixelement einen TRUE-Wert enthält, wird ein Punkt auf weiß gesetzt.

Bevor auf ein Beispiel näher eingegangen wird, müssen noch die Parameter der Prozedur DRAWBLOCK erläutert werden. Sie lauten:
DRAWBLOCK (QUELLE, BYTES, XSKIP, YSKIP, BREITE, HOEHE, XPOS, YPOS, MODUS)
QUELLE ist der oben definierte zweidimensionale PACKED ARRAY OF BOOLEAN. Die Dimensionen können frei gewählt werden, aber ein XMAX von mehr als 280 und ein YMAX von mehr als 192 sind nicht sinnvoll, da das ARRAY dann größer als der gesamte Grafikschirm wäre.
BYTES ist die Anzahl der Bytes pro Zeile. Sie ist mindestens 2 (16 Bits, da Pascal in Wortgrenzen arbeitet) und kann mit der Formel
2 * ((XMAX + 15) DIV 16)
berechnet werden. In unserem Beispiel mit XMAX = 4 bedeutet dies 2 * 1 = 2.
XSKIP gibt die Anzahl der Punkte in X-Richtung an, die übergangen werden sollen, bevor das Kopieren des ARRAYS auf den Schirm beginnt.
YSKIP gibt die Anzahl der Punkte in Y-Richtung an, die übergangen werden sollen, bevor das Kopieren beginnt. Achtung: Der ARRAY wird von unten nach oben gezeichnet.
BREITE gibt die Anzahl der Punkte in X-Richtung an, die aus dem ARRAY kopiert werden sollen.
HOEHE gibt entsprechend die Anzahl der Punkte in Y-Richtung an.
XPOS und YPOS ergeben den linken, unteren Eckpunkt auf dem Bildschirm an, wo das Kopieren beginnen soll.
MODUS entspricht dem MODUS aus der CHARTYPE-Prozedur im Abschnitt 4.5.

Nun aber zu handfesten Beispielen. Wir benutzen die Vereinbarung der Variablen BLOCK von oben und wollen ein Kästchen der Breite von 4 und der Höhe von 3 Bildpunkten mehrere Male auf den Bildschirm bringen.

```
PROGRAM KAESTCHEN;
USES TURTLEGRAPHICS;
CONST MODUS  = 1Ø;
      XMAX   =  4;
      YMAX   =  3;
VAR   BLOCK  : PACKED ARRAY
      [1..YMAX, 1..XMAX] OF BOOLEAN;
      ZEILE  : INTEGER;
      SPALTE : INTEGER;
      X      : INTEGER;
BEGIN
  INITTURTLE;
  FOR ZEILE := 1 TO YMAX DO
    FOR SPALTE := 1 TO XMAX DO
      BLOCK [ZEILE, SPALTE] := TRUE;
      {Setze alle Punkte}
  FOR SPALTE 2 TO 3 DO
    BLOCK [2, SPALTE] := FALSE;
    {Lösche Inneres}
{Initialisierungen beendet}
  FOR X := Ø TO 27 DO
    DRAWBLOCK (BLOCK, 2, Ø, Ø,
     XMAX, YMAX, X * 1Ø, Ø, MODUS);
  READLN;
END.
```

Die doppelte FOR-Schleife füllt alle Matrixelemente mit dem Wert TRUE, was später bedeutet, daß all diese Punkte auf dem Bildschirm gesetzt werden. Die zweite FOR-Schleife löscht aus dem Kästchen die zwei mittleren Punkte, so daß BLOCK folgende Form hat:

```
  X:
Y:* * * *
  *     *
  * * * *
```

Die dritte FOR-Schleife schließlich zeichnet 28 mal den so entstandenen Block in einer Zeile auf den Bildschirm, wobei der Abstand zwischen je zwei Startpunkten 10 Bildpunkte beträgt (X * 10). Der MODUS beträgt 10, was bedeutet, daß der ARRAY ohne Rücksicht auf den alten Bildschirminhalt auf diesen kopiert wird.
Ein letztes Beispiel soll die Möglichkeit von bewegten Objekten veranschaulichen. Wir nehmen das obige Kästchen und lassen es von links nach rechts wandern, indem wir die jeweils alte Position löschen. Mit MODUS = 6 kann dies sehr leicht bewerkstelligt werden. Das Programm sieht dann wie folgt aus:

```
PROGRAM BEWEGUNG;
USES TURTLEGRAPHICS;
CONST MODUS  = 6;
      XMAX   = 4;
      YMAX   = 3;
VAR   BLOCK  : PACKED ARRAY
      [1..YMAX, 1..XMAX] OF BOOLEAN;
      ZEILE  : INTEGER;
      SPALTE : INTEGER;
      X      : INTEGER;
BEGIN
  INITTURTLE;
  FOR ZEILE := 1 TO YMAX DO
    FOR SPALTE := 1 TO XMAX DO
      BLOCK [ZEILE, SPALTE] := TRUE;
      {Setze alle Punkte}
  FOR SPALTE 2 TO 3 DO
    BLOCK [2, SPALTE] := FALSE;
    {Lösche Inneres}
{Initialisierungen beendet}
```

```
  DRAWBLOCK (BLOCK, 2, Ø, Ø,
   XMAX, YMAX, Ø, Ø, MODUS);
  FOR X := Ø TO 27 DO
  BEGIN
    READLN;
    {Lösche zuerst alten Block}
    DRAWBLOCK (BLOCK, 2, Ø, Ø,
     XMAX, YMAX, (X - 1) * 1Ø, Ø, MODUS);
    {Zeichne dann neuen Block}
    DRAWBLOCK (BLOCK, 2, Ø, Ø,
     XMAX, YMAX, X * 1Ø, Ø, MODUS)
  END {FOR}
END.
```

Nach den Initialisierungen wird zunächst das erste Kästchen links unten auf den Bildschirm gebracht. Da die nachfolgende Schleife zu schnell ablaufen würde, um etwas zu sehen, muß nach jedem Durchgang ein <Return> eingegeben werden. Danach wird das Kästchen an der alten X-Position (X - 1) gelöscht und an der neuen Position gezeichnet. Es empfiehlt sich hier, MODUS 6 zu benutzen, da so das Programm bei beliebigem Hintergrund richtig arbeitet. Ein FILLSCREEN (WHITE) nach INITTURTLE bestätigt dies.

5. Bibliothekskonzept

Erst durch die Einführung von Segmenten und Bibliotheken ist es möglich geworden, lange und komplexe Programme in UCSD-Pascal zu entwickeln. Da der Speicher in Version 1.1 auf 64K und in Version 1.2 auf 128K beschränkt ist, muß es Möglichkeiten geben, auch wesentlich längere Programme darin unterzubringen. Dazu wird die Segmentierung, auch Overlay-Technik genannt, benutzt. Dabei werden gewisse Programmteile erst bei Benutzung von der Diskette nachgeladen, andere dafür ausgelagert.
Das Bibliothekskonzept ermöglicht eine getrennte Übersetzung und damit die Verwirklichung des modularen Aufbaus von Programmen. Komplexe Programme können so in überschaubare Module gespalten werden, die von verschiedenen Programmierteams implementiert werden können.

5.1. Segmentierung in UCSD-Pascal

Das Pascal-System selbst ist so lang, daß es eine Speicherplatzverschwendung wäre, wenn es sich immer gänzlich im Speicher befinden würde. Teile davon werden immer von Diskette nachgeladen. Man merkt dies daran, daß beim Verlassen des Filers das Laufwerk #4: kurz anläuft. Dann wird nämlich die Funktion GETCOMMAND, ein Segment von SYSTEM.PASCAL, nachgeladen. Sie enthält unter anderem die bekannte Kommandozeile, die nach dem Booten erscheint.
In UCSD-Pascal existieren maximal 32 Segmente, die durch ihre Nummern gekennzeichnet sind. So belegt das Pascalsystem selbst die Nummern 0 und 2 bis 6. Ein normal ausführbares Programm erhält die Segmentnummer 1, die beim Übersetzen mit der $L-Option in Spalte 2 angezeigt werden kann. Das bedeutet, daß die Nummern 7 bis 31 frei sind. Es existieren allerdings noch einige Beschränkungen, die unter 5.3 erläutert werden. Ein normales Programm kann, wenn es nicht in den Speicher paßt oder wenn mehr Speicher für Daten zur Verfügung stehen soll, in Segmente aufgeteilt werden. Die einfachste Art, dies zu bewerkstelligen, ist die Benutzung von Segment-Prozeduren und -Funktionen.

5.2. Segment-Prozeduren und -Funktionen

Sie sind nichts anderes als normale Prozeduren und Funktionen, denen das reservierte Wort **„SEGMENT"** vorangeht. Beispiel:

```
PROGRAM OVERLAY;
VAR C : CHAR;

SEGMENT PROCEDURE INIT;
BEGIN
  {Viele Anweisungen}
END;

SEGMENT PROCEDURE MENUE (C : CHAR);
  PROCEDURE MENUE1;
  BEGIN
    {...}
  END;
  PROCEDURE MENUE2;
  BEGIN
    {...}
  END;
BEGIN {MENUE}
  {Anweisungen}
END; {MENUE}

BEGIN {Hauptprogramm}
  READLN;
  INIT;
  REPEAT
    READ (C);
    MENUE (C)
  UNTIL C = 'Q';
END.
```

Beim Starten des Programms OVERLAY werden die beiden Prozeduren INIT und MENUE zunächst nicht in den Speicher geladen. Dies geschieht erst beim Aufruf der Prozeduren. Dabei muß die Diskette, auf der sich das Hauptprogramm befindet, in dem Laufwerk liegen, in dem dieses gestartet wurde. Nach Drücken der <Return>-Taste kann man sehen, daß das Laufwerk noch einmal anläuft: Die Prozedur INIT wird eingelesen und ausgeführt. (Zuvor sollte man das Laufwerk aber zur Ruhe kommen lassen.) Ist sie beendet, so wird der Speicher, den sie belegt hat, wieder freigegeben. Nachdem ein Buchstabe eingegeben wurde (READ (C)), läuft das Laufwerk erneut an, um die Prozedur MENUE nachzuladen. Sie kann jetzt den zusätzlichen freien Speicher von INIT mitbenutzen.
Zu Beginn von längeren Programmen werden häufig Initialisierungen vorgenommen, die nur einmal gemacht werden müssen. Es empfiehlt sich dann immer eine SEGMENT-Prozedur INIT, da sonst unnötigerweise Speicherplatz verschwendet wird.

Es existiert eine Einschränkung: SEGMENT-Prozeduren und -Funktionen müssen vor allen anderen Prozeduren deklariert werden.
Es kann mitunter vorkommen, daß eine SEGMENT-Prozedur eine Prozedur benötigt, die keine SEGMENT-Prozedur ist. Da aber SEGMENTE vor allen anderen Prozeduren übersetzt werden müssen, wäre die „normale" Prozedur nicht bekannt, also auch nicht aufrufbar. Um dieses Manko zu beheben, müssen alle „normalen" Prozeduren und Funktionen „FORWARD" erklärt werden. Im nachfolgenden Beispiel soll die SEGMENT-Prozedur INIT die „normale" Prozedur CLEARSCREEN benutzen:

```
PROGRAM DEMO_FORWARD;

PROCEDURE CLEARSCREEN;
FORWARD;

SEGMENT PROCEDURE INIT;
BEGIN
  CLEARSCREEN;
  {andere Initialisierungen}
END;

PROCEDURE CLEARSCREEN;
BEGIN
  WRITE (CHR (12)) {ASCII FORMFEED}
END;

BEGIN
  INIT;
END.
```

Die erste Prozedur, die Code erzeugt, ist die SEGMENT-Prozedur INIT, womit den Regeln der Segmentierung Genüge getan wurde.
Zu den SEGMENT-Nummern braucht hier nicht viel gesagt zu werden. Die erste freie Nummer, nämlich 7, wird der ersten SEGMENT-Prozedur zugewiesen, Nummer 8 bekommt die zweite usw. Beim Erzeugen eines Compilerlistings ist dies aus Spalte 2 zu erkennen.

5.3. Reguläre und Intrinsic-Units

In sog. **UNITS** können Sammlungen von Unterprogrammen getrennt übersetzt und von verschiedenen Programmen oder anderen UNITS genutzt werden. Somit wird die Idee der modularen Programmierung voll unterstützt.
In UCSD-Pascal unterscheidet man reguläre Units und Intrinsic-Units. Der einzige Unterschied besteht darin, daß reguläre Units mit dem Linker vor der Laufzeit in einen bestehenden Codefile eingebunden werden, womit der Programmcode länger wird, während Intrinsic-Units aus der bestehenden SYSTEM.LIBRARY beim Starten eines Programms mit in den Speicher geladen werden. Das hat den Vorteil, daß erstens nicht mehr gelinkt werden muß und zweitens der Codefile wesentlich kürzer sein kann. Der dritte Vorteil besteht darin, daß beim Ändern des Programmes eine Intrinsic-Unit nicht neu übersetzt werden muß. Da Intrinsic-Units also wesentlich praktischer sind, werden wir diese genauer besprechen als reguläre Units.
Eine UNIT ist uns in Abschnitt 4 schon begegnet. In der SYSTEM.LIBRARY befindet sich neben anderen wichtigen UNITS noch die UNIT TURTLEGRAPHICS. Sie enthält viele Prozeduren und Funktionen, die von verschiedenen Programmen genutzt werden können. Dazu stellt man ein

USES TURTLEGRAPHICS;

nach den Programmkopf, und schon sind alle Konstanten, Typen, Variablen, Prozeduren und Funktionen dem Hauptprogramm bekannt. Damit der Compiler auch weiß, wo er die Units zu suchen hat, nimmt er zunächst an, daß sie sich in der SYSTEM.LIBRARY befinden. Es gibt auch noch weitere Möglichkeiten durch Benutzung der $U-Option. Sie kann in „Pascal Tips und Tricks, Teil 4" nachgelesen werden.
Wir wollen nun eine eigene UNIT entwickeln, die von jedermann benutzt werden kann. Eine UNIT besteht aus 4 Teilen:
- dem Kopf
- dem INTERFACE-Teil
- dem IMPLEMENTATION-Teil
- dem Initialisierungsteil

Der Kopf von regulären Units besteht aus:
UNIT Name;

bei Intrinsic-Units dagegen aus:
UNIT Name;
INTRINSIC CODE Codesegnum <DATA Datasegnum>;
Codesegnum und Datasegnum sind Segment-Nummern, die selbst gewählt werden müssen. Es empfiehlt sich die Benutzung von Nummer 16 an aufwärts, um eventuelle Konflikte mit SEGMENT-Prozeduren, deren Nummern bei 7 beginnen, zu vermeiden. Einige Segment-Nummern sind auch schon durch die SYSTEM.LIBRARY besetzt. Es sind dies:

20: TURTLEGRAPHICS CODE
21: TURTLEGRAPHICS DATA
22: APPLESTUFF
28: CHAINSTUFF
30: LONGINTIO
31: PASCALIO

Im INTERFACE-Teil, das dem Kopf einer UNIT folgt, wird die Schnittstelle zum Hauptprogramm beschrieben. Dort werden alle Konstanten, Typen, Variablen, Prozeduren usw. vereinbart, die später dem Hauptprogramm bekannt sein sollen. So steht beispielsweise im INTERFACE-Teil der TURTLEGRAPHICS-UNIT die Prozedur INITTURTLE, die wir im Abschnitt 4 benutzt haben.

Im IMPLEMENTATION-Teil stehen weitere Konstanten, Typen usw., die nur der UNIT selbst bekannt sein dürfen. Darauf folgen die Prozeduren und Funktionen, die im INTERFACE-Teil deklariert wurden. Hier werden sie aber ohne Parameter aufgeführt.

Im Initialisierungsteil schließlich stehen Statements, die nach dem Laden einer UNIT einmal ausgeführt werden. Dort könnte man Initialisierungen unterbringen, die der Benutzung der Unit vorangehen sollen. Möchte man keine Initialisierung vornehmen, so genügt ein leeres BEGIN END.
Noch eine Vorbemerkung: Direkt vor dem Kopf einer UNIT muß die Compiler-Option $S+ stehen, da sonst ein STACK OVERFLOW beim Übersetzen auftreten würde (nicht bei 128K-Pascal).
In unserem Beispiel sollen in einer Intrinsic-Unit mit Namen SCREENSTUFF die Prozeduren CLEARSCREEN, INVERSE, NORMAL und PRINTINVERSE jedem Programm zur Verfügung gestellt werden.

CLEARSCREEN soll den Bildschirm löschen,
INVERSE schaltet auf inverse Schrift,
NORMAL wieder zurück, und
PRINTINVERSE druckt einen übergebenen String in inverser Schrift.
Die UNIT sieht dann wie folgt aus:

```
{$S+}
UNIT SCREENSTUFF;
INTRINSIC CODE 16;

INTERFACE

PROCEDURE CLEARSCREEN;
PROCEDURE INVERSE;
PROCEDURE NORMAL;
PROCEDURE PRINTINVERSE (S : STRING);

IMPLEMENTATION

PROCEDURE CLEARSCREEN;
BEGIN
  WRITE (CHR (12))
END;

PROCEDURE INVERSE;
BEGIN
  WRITE (CHR (15))
END;

PROCEDURE NORMAL;
BEGIN
  WRITE (CHR (14))
END;

PROCEDURE PRINTINVERSE {S : STRING};
BEGIN
  INVERSE;
  WRITE (S);
  NORMAL
END;

BEGIN
  PRINTINVERSE ('Dieser String wird
    nur einmal nach dem Laden gedruckt')
END.
```

Die Segment-Nummer des Codes der UNIT ist 16. Ein Datensegment wird nicht benötigt, da im Interface-Teil keine globalen Variablen deklariert wurden bzw. im Implementation-Teil keine privaten. Wer sich unsicher ist, wann ein Datensegment benötigt wird und wann nicht, der übersetze die UNIT zunächst ohne es und überlasse dem Compiler die Entscheidung. Im anschließenden Interface-Teil stehen die 4 Prozeduren aufgelistet, die später einem Pascal-Programm bekannt sein sollen. Im Implementation-Teil werden die 4 Prozeduren explizit programmiert, wobei darauf zu achten ist, daß Parameter nicht mehr aufgeführt werden dürfen. In obigem Beispiel gilt dies für die Prozedur PRINTINVERSE, deren Parameter in Kommentarklammern angegeben worden sind. Dadurch wird die Lesbarkeit der Unit wesentlich verbessert, da man nicht dauernd zwischen INTERFACE- und IMPLEMENTATION-Teil hin- und herblättern muß, um die Parameter einer Prozedur zu finden. Der Initialisierungsteil besteht aus dem Ausdrucken einer Meldung in inverser Schrift und soll lediglich zeigen, wann dieser Teil beim Laden ausgeführt wird.

Nach dem erfolgreichen Übersetzen benennen wir den SYSTEM.WRK.TEXT und -CODE im FILER in SCREENSTUFF.TEXT und SCREENSTUFF. CODE um. Nun muß SCREENSTUFF noch in der SYSTEM.LIBRARY installiert werden. Dazu benutzt man das LIBRARY-Programm, das sich auf der APPLE3-Systemdiskette befindet.
Nach dem Starten muß auf die Frage
OUTPUT CODE FILE
mit „*" für SYSTEM.LIBRARY geantwortet werden. Auf die Frage
LINK CODE FILE
antwortet man ebenfalls mit „*", da alle Segmente der SYSTEM.LIBRARY kopiert werden sollen. Um dies zu bewerkstelligen, gibt man ein „=" ein. Daraufhin werden alle Segmente, die sich in einem der 16 Slots befinden, kopiert. Nun benötigen wir noch die eben übersetzte SCREENSTUFF-Unit. Dazu wählen wir ein „N" für N(EW FILE und antworten auf
LINK CODE FILE
mit SCREENSTUFF. Es erscheint im Slot 1 die Segmentnummer 16, gefolgt vom Segmentnamen SCREENST. Wie man sieht, werden nur die ersten 8 Buchstaben benutzt. Wir kopieren Slot 1 in Slot 7, indem wir
1 <Space> 7 <Space>
eingeben. Damit ist die Installierung beendet, und wir können das Programm mit „Q" verlassen. Auf die Frage
NOTICE
reicht es hier, mit einem <Return> zu antworten.

Das Benutzen unserer neuen SYSTEM.LIBRARY ist sehr einfach. Es muß nur ein
USES SCREENSTUFF;
nach dem Programmkopf stehen. Ein Testprogramm könnte etwa so aussehen:

```
PROGRAM TESTLIB;
USES SCREENSTUFF;
VAR I : INTEGER;
BEGIN
  INVERSE;
  WRITELN ('Inverse Schrift:');
  FOR I := 1 TO 10 DO
    WRITELN ('I: ', I : 2);
  NORMAL;
  WRITELN ('und wieder normal');
  PRINTINVERSE ('Inverser String');
  WRITELN ('Normaler String')
END.
```

5.4. Erweiterungen in Pascal 1.2 (128K-Version)

Da mit 128K Hauptspeicher wesentlich größere und komplexere Aufgaben bewältigt werden können, wurde die Anzahl der verfügbaren Segmente auf 64 erhöht. Das heißt, daß die Segmentnummern hier im Bereich von 0 bis 63 liegen können.
Die Flexibilität bei der Benutzung von LIBRARIES wurde wesentlich erhöht. So kann ein CODEFILE, der beispielsweise unter dem Namen
XXX.CODE
auf Diskette steht, seine eigene Library, die
XXX.LIB
heißen muß, benutzen. Es können also programmspezifische LIBRARIES erstellt werden. Diese werden beim Starten des Programmes zuerst nach den zu benutzenden Segmenten durchforstet, bevor in der SYSTEM.LIBRARY gesucht wird.
Eine letzte Möglichkeit besteht darin, daß
XXX.LIB
keine LIBRARY-Code-Datei ist, sondern eine Textdatei, in der sich die Namen von weiteren LIBRARIES befinden, die beim Starten von XXX.CODE benutzt werden sollen. Dies führt hier aber zu weit, es sollte nur der Vollständigkeit halber erwähnt werden. Wer sich dafür interessiert, muß in den entsprechenden Handbüchern nachschlagen.

Literatur

Apple: Apple Pascal, Operating System Reference Manual
Apple: Apple Pascal, Language Reference Manual
Apple: Addendum to the Apple Pascal Language Reference Manual
Apple: Apple Pascal 1.2 Update Manual

Peeker bietet Ihnen Monat für Monat aktuelle Fachinformationen. Lesen Sie deshalb ständig Peeker, damit Sie früher mehr wissen als andere

Bei eiligen Rückfragen oder Bestellungen erreichen Sie uns über Telex 461 727 huehd d mit Vermerk: bestimmt für „Peeker" – oder per Telefon über die Durchwahl (06221) 489206.

KIX System Shell

UNIX-ähnliches User-Interface für das ProDOS-Betriebssystem

von Matthias Meyer

1. Einsatzgebiet

KIX erweitert das ProDOS-Betriebssystem um eine RAM-residente, standardisierte Benutzeroberfläche, die dem UNIX-Betriebssystem nachempfunden wurde. Es stellt dem Anwender viele systemnahe Utilities zur Verfügung, die man sonst nur auf großen Multi-User-Systemen wiederfindet. Bei KIX handelt es sich also nicht um ein neues Betriebssystem, sondern um eine Sammlung externer Befehle zur Verwaltung von Directories, Files, Disketten und vielem mehr. KIX erfordert eine gewisse Einarbeitungszeit, sofern man nicht mit dem Betriebssystem UNIX vertraut ist. Es eignet sich deshalb vor allem für denjenigen Anwender oder Programmierer, dem der ProDOS-Filer zu umständlich, zu primitiv oder nicht effizient genug ist.

2. Einführung

Um Einsteigern die Lektüre dieses Artikels zu erleichtern, werden hier die wichtigsten im nachfolgenden Text verwendeten Begriffe erläutert.

2.1. Betriebssysteme

Betriebssysteme organisieren und leiten den Informationsfluß innerhalb eines Computersystems. Wenn Informationen gespeichert werden sollen, dann ist es das Betriebssystem, das den Computer anweist, auf welche Art, wann und wo genau diese Daten im System abgelegt werden sollen. Wenn Informationen aus dem Computer zu lesen sind, ist es wiederum Aufgabe des Betriebssystems, Daten von den externen Speichermedien zu lesen und in der gewünschten Form weiterzuleiten.

Zusätzlich zu den Datenspeichern werden vom Betriebssystem weitere Ein-/Ausgabegeräte unterstützt, z.B. Tastatur, Bildschirm und Drucker. Ein gutes Betriebssystem verwaltet die gesamte Hardware des Computers. Dazu gehören auch nachträglich eingebaute RAM-Karten, Uhrenkarten, 80-Zeichenkarten und andere nützliche Erweiterungen. Es ist die Aufgabe des Betriebssystems, alle diese Geräte effizient zu verwalten und die Produktivität der Hardware zu optimieren.

2.2. Computersysteme

Ein typisches Computersystem besteht aus folgenden Komponenten:

– *Mikroprozessor*
Die CPU (Central Processing Unit) ist das „Gehirn" des Computers. Sie erhält Befehle vom Betriebssystem und führt sie aus, ist also die Ursache aller Aktivitäten des Computers.

– *Kurzzeitspeicher*
Die CPU verwendet den RAM (Random Access Memory) als Arbeitsspeicher für Daten, die momentan bearbeitet werden. Der RAM ist ein elektronischer Speicher. Informationen können im RAM in sehr kurzer Zeit gespeichert und abgerufen werden. Die im RAM abgelegten Informationen gehen beim Ausschalten des Computers verloren.

– *Langzeitspeicher*
a) Magnetische Speichermedien: Der Computer verwendet Disketten, Festplatten oder Magnetbänder für die dauerhafte Speicherung von Programmen und Informationen. Das Betriebssystem kontrolliert bei diesen Magnetspeichern die Organisation (das Format) der abgespeicherten Informationen und sorgt dafür, daß die gewünschten Daten so schnell wie möglich gefunden bzw. abgespeichert werden.
b) ROM (Read Only Memory): Eine weitere Art von Langzeitspeicher ist der „Nur-Lese-Speicher". Der ROM enthält Programme, die fest in einem Chip eingebrannt wurden. Er ist ein innerer Bestandteil der Computerschaltkreise. Die CPU kann von ROM Informationen wesentlich schneller lesen, als dies bei magnetischen Speichern möglich ist. Gewöhnlich ist der ROM nach dem Einschalten eines Computers der zuerst verwendete Speicher; er enthält für die CPU Anweisungen zum Laden und Starten des Betriebssystems.

3. Aufbau von KIX

3.1. Installation

Der Kommando-Interpreter nimmt die Anweisungen des Benutzers entgegen, entspricht also dem CCP eines CP/M-Systems. Er befindet sich im File KIX.SYSTEM und wird direkt nach dem Booten von ProDOS ab Adresse $2000 eingelesen. Danach wird der Kommando-Interpreter gegen das ProDOS-Reboot-Programm ausgetauscht; das vorher installierte ProDOS-Reboot-Programm wird dazu in den QUIT-Befehl geschrieben. Der KIX-Shell belegt von nun an den Speicherbereich $D100-$D3FF (LC-Bank 2). Diese Vorgänge laufen alle automatisch ab, der Anwender braucht sich darum nicht zu kümmern.

KIX meldet sich danach mit dem „%"-Prompt. Dieses Symbol besagt: „KIX ist bereit, Ihre Kommandos auszuführen". Durch Eingabe von CFG <CR> gelangt man in ein Konfigurationsprogramm, mit dem man die im Boot-File KIX.SYSTEM eingetragenen Grundeinstellungen verändern kann. Man kann hier z.B. angeben, ob die KIX-Utilities beim Starten automatisch in eine RAM-Disk kopiert werden sollen, ob man mit 40- oder 80-Zeichen Zeilenbreite arbeiten möchte usw.

3.2. File-Strukturen

Um ein Betriebssystem effektiv einsetzen zu können, ist es wichtig, die Directory- und File-Struktur genau zu verstehen.
Die Benutzerinformationen werden auf den Disketten in Form von Files abgelegt. Ein File kann jede beliebige Größe annehmen und beliebige Informationen enthalten.
Files werden vom Benutzer benannt, damit sie auf der Diskette von anderen Files unterschieden werden können. Diese Filenamen müssen bei jedem Betriebssystem verschiedene Kriterien erfüllen.

Unter ProDOS dürfen die Namen nicht länger als 15 Zeichen sein, müssen mit einem Buchstaben beginnen und dürfen nur Buchstaben, Zahlen und den Dezimalpunkt enthalten. Leerzeichen und andere Spezialzeichen sind in ProDOS-Filenamen nicht erlaubt.

Das Betriebssystem stellt dem Benutzer ein oder mehrere Inhaltsverzeichnisse, sog. *Directories*, zur Verfügung, in denen die einzelnen Files mit Namen, Typ und Größe eingetragen und verwaltet werden. Möchte der Benutzer auf einen bestimmten File zugreifen, so wählt er zunächst das gewünschte Directory, in dem der File-Eintrag steht. Dann kann er auf den einzelnen File zugreifen, in dem die gesuchten Informationen enthalten sind.

Auf einer ProDOS-Diskette kann man eine nahezu beliebige Anzahl von Directories erzeugen. Zur effizienten Organisation der auf Diskette gespeicherten Informationen ist es manchmal sinnvoll, sog. *Subdirectories* zu verwenden. Subdirectories sind Unter-Inhaltsverzeichnisse, d.h. in einem Directory befindet sich der Eintrag für ein weiteres, untergeordnetes Directory, das eigene File-Einträge enthält.

Für die Directory- und Subdirectory-Bezeichnungen gelten die gleichen Regeln wie für Filenamen. Directory-Namen dürfen nicht länger als 15 Zeichen sein, dürfen keine Leerzeichen oder andere Spezialzeichen enthalten und müssen mit einem Buchstaben beginnen.

Das System, mit dessen Hilfe man Files in Directories und Subdirectories abspeichern kann, wird als „hierarchisches File-System" bezeichnet. Ein hierarchisches File-System dient dazu, große Informationsmengen zu organisieren; es ermöglicht dem Computer und dem Benutzer einen sehr schnellen Informationszugriff.

3.3. File-Typen

Das Betriebssystem verwendet einige generell einsetzbare Filetypen.

Textfiles sind Benutzerfiles. Sie enthalten Informationen, die vom Benutzer gelesen werden können: gewöhnliche Texte, Briefe, Zahlen, Tabellen usw. Die meisten Files, die ein Benutzer erstellt, sind Textfiles.

Anwendungsprogramme sind Aneinanderreihungen von Instruktionen, die den Computer durch das Betriebssystem bestimmte Aufgaben für den Benutzer erledigen lassen. Ein Textverarbeitungsprogramm ist ein Anwendungsprogramm, mit dem der Anwender z.B. einen Brief schreiben kann. Anwendungsprogramme sind nicht Teil des Betriebssystems, sondern sie verwenden es als Mittel für eine bestimmte Anwendung und berücksichtigen dabei die vom Betriebssystem vorgegebenen Datenstrukturen.

Utilities: Das Betriebssystem speichert Informationen in Files und Directories. Um File-Strukturen zu erstellen und zu erhalten, benötigt man Hilfsmittel zum Kopieren und Übertragen von Files, Erstellen und Löschen von Directories usw. Das Betriebssystem enthält Programme für diese Aufgaben, die sog. Hilfsprogramme oder Utilities. ProDOS enthält einen kleinen Teil dieser Utilities im FILER. KIX erhöht die Anzahl der verfügbaren Utilities durch eine Sammlung von 25 UNIX-ähnlichen Befehlen. Eine vollständige Implementation des UNIX-Betriebssystems enthält über 200 Hilfsprogramme.

In KIX sind folgende File-Typen vordefiniert: BIN (Binärfile), DIR (Directory), SYS (System File), TXT (Textfile).

Weitere File-Typen des Betriebssystems ProDOS: BAS (Applesoft-BASIC-Programm), VAR (Applesoft-Variablen), AWP (AppleWorks Textverarbeitung), ASP (AppleWorks Tabellenkalkulation), ADB (AppleWorks Datenbankdatei), REL (Relocatable Code File).

3.4. ProDOS-Pfadnamen

Um Files in einem hierarchischen System zu speichern und wiederzufinden, benötigt das Betriebssystem einen Wegweiser durch das Netz von Directories und Subdirectories. Ein *Pfadname* gibt den genauen Weg an, den das Betriebssystem gehen muß, um einen gesuchten File zu finden.

Ein vollständiger Pfadname beginnt immer mit einem Schrägstrich („/") und dem Volume-Namen. Dieser bezeichnet das Root-Directory oder Volume-Directory (Hauptinhaltsverzeichnis). Jede Diskette hat ein Volume- oder Root-Directory, das eine Liste aller Files und Subdirectories dieser Diskette beinhaltet. Das Volume-Directory ist das erste Inhaltsverzeichnis; es steht auf der höchsten Stufe in der hierarchischen File-Struktur. Von ihm gehen alle tieferen Verzweigungen innerhalb der Directory-Hierarchie aus (Baumstruktur).
In den Pfadnamen werden die Namen der untergeordneten Subdirectories jeweils durch zusätzliche Schrägstriche abgetrennt. Die letzte Bezeichnung des Pfadnamens ist der Name des gesuchten Files.
Beispiel für einen Pfadnamen:
/KIX/BIN/CFG
Dieser Pfadname beschreibt den File CFG, der sich auf einer mit /KIX bezeichneten Diskette im Subdirectory BIN befindet.

3.5. Arbeits-Directory

Es ist nicht immer erforderlich, den gesamten Pfadnamen anzugeben, wenn man mit komplexen File-Strukturen arbeitet. Man kann den Pfad des (Sub-)Directory angeben, mit dem man gerade arbeitet, und diesen in einem Präfix oder Arbeits-Directory speichern. KIX stellt für diesen Zweck den „CD"-Befehl zur Verfügung, der für „Change Directory" steht.
Ein Beispiel: Sie arbeiten gerade mit Files, die im Directory /KIX/USER/MYDIR gespeichert sind. Um dieses Präfix festzulegen, geben Sie ein:
„CD /KIX/USER/MYDIR".
Von nun an können Sie auf die in diesem Directory gespeicherten Files direkt zugreifen, d.h. nur unter Angabe des Filenamens anstelle des vollständigen Pfadnamens.

3.6. BIN-Directory

Analog zu UNIX wurde auch in KIX ein BIN-Directory eingerichtet, das alle Utility-Befehle beinhaltet. Das bedeutet: Es wird immer dann versucht, einen Befehl aus dem BIN-Directory zu laden, wenn die KIX-Shell vom Benutzer ein Kommando erhält, welches nicht im Arbeits-Directory gefunden werden konnte und dem auch kein Pfadname vorangestellt wurde. Daraus folgt, daß man problemlos in einem beliebigen Directory arbeiten kann; trotzdem stehen alle Systembefehle ohne Angabe eines Pfadnamens jederzeit zur Verfügung.

3.7. KIX-Betriebssystem-Umgebung

KIX ist eine auf ProDOS basierende Betriebssystem-Umgebung für den Apple II. Es wurde der Benutzeroberfläche von UNIX nachempfunden und stellt dem Benutzer viele Utilities zur Verfügung, die man von größeren Computersystemen her kennt. KIX stellt eine Verbindung zwischen dem Benutzer und dem ProDOS-Kern dar. Diese beiden Elemente – die Benutzerschnittstelle (oder Betriebssystemoberfläche) und der Betriebssystemkern – repräsentieren die äußere und innere Welt eines Betriebssystems.

Der ProDOS-Kern ist der innere oder unsichtbare Teil des Betriebssystems. Er arbeitet automatisch und verwaltet Tastatur, Diskettenlaufwerke, Mikroprozessor, Drukker und andere Ein-/Ausgabegeräte. Der ProDOS-Kern führt Befehle aus, die er von der KIX System Shell bekommt.

Die KIX-Shell ist der äußere oder sichtbare Teil des Betriebssystems. Sie stellt die Verbindung zwischen dem Benutzer, den Benutzerfiles und der Computer-Hardware her. Wenn der Benutzer einen Befehl eingibt, interpretiert die KIX-Shell diesen Befehl, ruft die erforderlichen Anwendungs- und Hilfsprogramme auf und arbeitet zusammen mit dem ProDOS-Kern, um die Anweisungen des Benutzers wunschgemäß auszuführen.

Aufgrund der Tatsache, daß ProDOS permanent im Einsatz ist und alle grundlegenden Betriebssystem-Funktionen übernimmt, ist KIX kompatibel mit allen ProDOS-Funktionen und nahezu jeder auf ProDOS basierenden Software.

4. Arbeiten mit KIX

4.1. KIX System Shell

Die KIX-Shell und die Benutzerschnittstelle werden von dem File KIX.SYSTEM erzeugt, den man auf der KIX-Systemdiskette findet. Nach dem Booten der KIX-Diskette wird zuerst der ProDOS-Kern geladen, dann KIX.SYSTEM. Über das Prompt-Zeichen ist jederzeit leicht feststellbar, ob die KIX-Shell aktiv ist, denn KIX verwendet das Prozent-Zeichen (%) als System-Prompt. KIX wird mit dem Quit-Befehl verlassen. „Quit" lädt automatisch den ProDOS-Reboot-Befehl oder jenes Programm, das vor dem Start von KIX den Speicherbereich $D100-$D3FF belegte.

4.2. KIX-Befehlssyntax

Die KIX-Befehlssyntax hat die folgende Form:
Befehl -Optionen Argumente
Beim Befehl handelt es sich um den Kurznamen der Kommando-

Utility, die Sie aufrufen wollen (z.B. CP zum Kopieren). Diese Hilfsprogramme werden im nachfolgenden Kapitel einzeln beschrieben.
Optionen sind spezielle Anweisungen an die Utility, den Befehl in einer vom Standard abweichenden Weise auszuführen. Viele KIX-Befehle besitzen die Möglichkeit zu Optionen. Zum Beispiel bietet der „LS"-Befehl (List Directory) Optionen für eine erweiterte Directory-Anzeige, für ein alphabetisch sortiertes Directory usw. Werden keine Optionen angegeben, so verwendet KIX stets die Grundeinstellungen für den jeweiligen Befehl.
Der Argument-Teil der KIX-Befehlssyntax ist ein Pfadname. Das Argument gibt den oder die File(s) an, die mit dem Befehl zusammenhängen bzw. von der Utility bearbeitet werden sollen.
Beispiel: Ein Befehl soll Files auf Diskette löschen. Das Argument gibt hier die Pfadnamen der zu löschenden Files an.

4.3. KIX-Befehlsumfang

Die KIX-Utilities kann man in die folgenden sieben Kategorien einteilen:

4.3.1. Directory-Management
„CD Pfadname" wählt ein anderes Arbeits-Directory (Präfix wechseln);
„LS Pfadname" zeigt eine Liste von Directories und Files;
„PWD" zeigt den Pfadnamen des momentanen Arbeits-Directory;
„MKDIR Pfadname" erstellt ein Subdirectory;
„RMDIR Pfadname" löscht ein Subdirectory.

4.3.2. File-Management
„CAT Pfadname" zeigt den File-Inhalt;
„CHMOD Pfadname" ändert die Zugriffsbefugnis von Files und Directories;
„CP Quelle Ziel" kopiert File(s);
„LPR Pfadname" gibt einen File auf dem Drucker aus;
„MV Altname Neuname" überträgt einen File in ein anderes Directory oder benennt ihn neu;
„RM Pfadname" löscht File oder Directory.

4.3.3. Volume-Management
„CPV (Slot, Drive) (Slot, Drive)" kopiert eine Diskette;
„FORMAT (Slot, Drive) /Name" formatiert eine Diskette;
„MVV (Slot, Drive) /Name" benennt eine Diskette um.

4.3.4. Spezielle KIX-Befehle
„CMP Filename1 Filename2" vergleicht zwei Files;
„CMP (Slot, Drive) (Slot, Drive)" vergleicht zwei Disketten;
„FIND Directory -Filename" sucht einen File in der gesamten untergeordneten Directory-Struktur;
„GREP Zeichenkette Pfadname" sucht einen String in einem File;
„SDIFF Filename1 Filename2" vergleicht zwei Textfiles.

4.3.5. Abkürzungen und Wildcards
. (Abkürzung) beschreibt das Arbeits-Directory;
.. (Abkürzung) beschreibt das Directory, das in der Hierarchie an nächsthöherer Stelle liegt;
? (Wildcard) steht für einen beliebigen Buchstaben;
* (Wildcard) steht für eine beliebige Zeichenkette;
ECHO Pfadname zeigt alle Pfadnamen, die von der Wildcard-Verwendung betroffen sind.

4.3.6. Firmware-Aufrufe
„C40" schaltet die 80-Zeichen-Darstellung aus;
„C80" schaltet die 80-Zeichen-Darstellung ein;
„DATE jjmmttssmm" setzt System-Datum und -Uhrzeit;
„SD" druckt den aktuellen Bildschirminhalt aus.

4.3.7. KIX-Utility-Befehle
CFG verändert die Grundeinstellungen von KIX;
INSTALL installiert KIX auf einer 800K-Disk oder Festplatte;
KIX enthält eine zusammenfassende Beschreibung der KIX-Befehle;
QUIT verläßt KIX und startet ProDOS-Reboot-Programm.

5. Erfahrungen mit KIX

5.1. Ältere Versionen

In der Version 1.0 hatte KIX noch einige Macken. Zum Beispiel war es bei einigen Befehlen nicht möglich, die Ausgabe vom Bildschirm auf den Drucker oder auf Diskette umzuleiten. Ferner wurden die 80-Zeichenkarten der Firma Videx bei vielen Befehlen nicht korrekt angesteuert (der Bildschirm wurde nicht gelöscht). Mit der Bildschirmdarstellung gab es auf dem Original Apple II+ Probleme, da Kleinbuchstaben zum Teil nicht in Großbuchstaben umgewandelt wurden und einzelne Bildschirm-Menüs deshalb nicht lesbar waren. Weitere Probleme betrafen die alte ROM-Version des Apple IIe, denn bei vielen KIX-Utilties dient die Escape-Taste als Abbruchmöglichkeit. Aufgrund eines Fehlers in den alten ROMs bewirkt jedoch das Drücken der Escape-Taste etwas ganz anderes, nämlich ein Umschalten in den Escape-Modus. Eine Änderung in der KIX-Software, die zukünftig nicht mehr die genormten Tastatur-Routinen verwendet, löste auch dieses Problem.

5.2. Die neue Version

Inzwischen gibt es schon die Version 1.2, und KIX läßt sich jetzt mit einem Tastendruck aus dem integrierten Programm AppleWorks heraus starten. KIX wird inzwischen einzeln zu einem Preis von US$ 49.95 vertrieben und kann auch für US$ 99.95 zusammen mit Kyan-Pascal unter der Bezeichnung Kyan-Pascal Plus erworben werden. Diese Version enthält aber keine AppleWorks-Option. In diesem Zusammenhang möchte ich die Update-Politik der Firma Kyan loben. Gegen Einsendung einer beliebigen Kyan-Originaldiskette innerhalb von 90 Tagen nach Kaufdatum erhält man kostenlos per Luftpost die neueste Version dieser Software auf einer neuen Diskette. Nach Ablauf dieser Frist kostet ein Update US$ 10.00. Daneben gibt es für US$ 15.00 im Jahr ein alle zwei Monate erscheinendes Blatt „Update Kyan", in dem neue Produkte angekündigt und nützliche Pascal-Programme veröffentlicht werden.

5.3. Persönliche Erfahrungen

Das KIX-Betriebssystem verwende ich jetzt seit etwa vier Monaten. Eigentlich hatte ich ja damals ein Pascal-System bestellt, doch zu meinem Erstaunen erhielt ich damals von Kyan Software nicht nur ein Pascal-System, sondern auch unzählige Betriebssystem-Utilities gratis dazu. Den Grund dafür sollte ich erst später erfahren, es war nämlich so: Bevor KIX offiziell auf den Markt kam, wurde es in einer einfachen Version als kostenlose Ergänzung des Kyan-Pascal-Systems ausgeliefert, um vorab die Reaktion potentieller Käufer zu testen.

Nach intensiver Einarbeitung möchte ich diese weitgehende Erweiterung des ProDOS-Betriebssystems inzwischen nicht mehr missen. Anstatt mich mühsam durch primitive FILER-Menüs kämpfen zu müssen, um einen einzigen File zu kopieren, gebe ich jetzt nur noch eine kurze Kommandozeile ein und schon ist das Problem gelöst. Das Erlernen der Befehlssprache bis in alle Einzelheiten ist zwar am Anfang zeitaufwendig, aber man lernt es eben nur einmal und genießt danach den aus dem effektiveren Arbeiten resultierenden Zeitvorteil immer wieder.

Schon nach relativ kurzer Zeit wurde mir der Unterschied zwischen dem starren CATALOG-Befehl des BASIC.SYSTEMs und dem „LS"-Befehl von KIX bewußt. Obwohl ich stets versuche, so wenige Files wie möglich in einem Directory zu halten, kommt es doch immer wieder vor, daß es gelegentlich mehr werden, als es Zeilen auf dem Bildschirm gibt. In diesem Fall nehme ich jetzt einfach den LS-Befehl mit der kurzen Directory-Anzeige, und schon passen alle Einträge auf den Bildschirm. Dies ist natürlich nur ein kleines Beispiel, aber insgesamt gewinnt man den Eindruck, daß hier solide Denkarbeit vorliegt.

AppleWorks-KIX konnte ich leider nicht installieren und austesten, da ich noch eine uralte Version von AppleWorks verwende und darin außerdem eine ältere Version der RAMWorks-Patch-Software (Version 3.7.) installiert habe. (RAMWorks ist eine Speichererweiterung von Applied Engineering, die in den Auxiliary Slot gesteckt wird und gleichzeitig die 80-Zeichenkarte ersetzt.) Laut KIX-Manual soll der AppleWorks-Patch nur mit RAMWorks-Software ab Version 4.2 laufen, so daß ich weitere Versuche in dieser Richtung vorerst aufgegeben habe.

Fazit

KIX ist meiner Meinung nach eine gelungene und inzwischen auch relativ ausgereifte Erweiterung des ProDOS-Betriebssystems. Wünschenswert wären sicherlich noch weitere Befehle, doch bei der Diskettenkapazität eines typischen Apple-II-Systems wären diese kaum noch unterzubringen. KIX kann seine Vorteile aus diesem Grund besonders in Verbindung mit einer Festplatte ausspielen. Eine weitere Verfeinerung und Verbesserung dieses Betriebssystems ist möglich. Auch arbeitet Kyan Software inzwischen an einer 16-Bit-Version für den neuen Apple IIgs. Persönlich setze ich KIX sehr gerne in Verbindung mit Kyan-Pascal ein, denn in der Entwicklungsphase von Pascal- oder Assemblerprogrammen ist KIX für mich genau die richtige Betriebssystem-Umgebung.

Peeker-Sammeldisk #24

DOS-3.3-Diskette; Heft 12/1986
Einzelbezug DM 28,-
Fortsetzungsbezug DM 20,-
(1) Zweck; (2) Heft/Seite; (3) Gerätekonfiguration; (4) Betriebssystem; (5) Programmstart; (6) Sonstiges

T 062 T.FILECOPY
B 007 FILECOPY
(1) Dateikopierbefehl für 2- und 1-Drive-Besitzer zum Einbinden in das BASIC.SYSTEM; (2) Heft 12/86, S. 6; (3) Apple II+/e/c; (4) BASIC.SYSTEM 1.0, 1.1; (5) BRUN FILECOPY; (6) Die Dateien müssen zunächst mit CONVERT oder DOSTOPRO von der Sammeldisk auf Ihre ProDOS-Arbeitsdisk konvertiert werden.

A 008 STARTUP
(1) Hello-Programm für BASIC.SYSTEM; (2) Heft 12/86, S. 15; (3) Apple II+/e/c; (4) ProDOS-BASIC.SYSTEM 1.1; (5) RUN START UP; (6) STARTUP muß zunächst mit CONVERT von der Sammeldisk auf Ihre ProDOS-Arbeitsdiskette konvertiert werden.

A 003 MACRO.BIN.MAKER
A 002 START.FW.MACRO
T 008 MACRO.TXT
B 002 MACRO.BIN
T 004 TESTTEXT
T 014 T.DRIVER
B 003 DRIVER
(1) Druckertreiber für Fast-Writer und ggf. andere Textverarbeitungsprogramme; (2) Heft 12/86, S. 16; (3) Apple II+/e/c; (4) DOS 3.3 (oder ProDOS); (5) Erst Makro-Tabelle mit Fast-Writer o.ä. anlegen, dann RUN MACRO.BIN.MAKER; (6) Wegen Speicherverteilung usw. siehe Aufsatz.

A 021 BASIC.MASKE
(1) Applesoft-Unterroutine für Eingabemasken; (2) Heft 12/86, S. 22; (3) Apple II+/e/c in 40 Z/Z; (4) DOS 3.3 oder ProDOS; (5) RUN BASIC.MASKE

A 005 FMULT.DEMO
T 006 T.FMULT.PATCH
B 002 FMULT.PATCH
(1) Demonstration und Behebung des Bugs in der Applesoft-Multiplikationsroutine; (2) Heft 12/86, S. 29; (3) Apple II+ mit LC oder e/c; (4) DOS 3.3 (nicht ProDOS); (5) RUN FMULT.DEMO

T 017 T.LINE.REFS
B 003 LINE.REFS
A 002 LINE.REFS.DEMO
(1) Auflistung aller Zeilenverweise in Applesoft-Programmen zur systematischen Programmentwicklung; (2) Heft 12/86, S. 32; (3) Apple II+/e/c; (4) DOS 3.3 oder ProDOS; (5) RUN LINE.REFS.DEMO; (6) Kollidiert speichermäßig mit MACROEDITOR.

T 098 T.DHGR.IIPLUS
B 009 DHGR.IIPLUS
B 009 DHGR.IIPLUS.PRO
B 009 DHGR.IIPLUS.48K
A 004 DHGR.GENAU
A 004 DHGR.EPSON
(1) Double-Hires-Ampersand-Routinen für Apple II+; (2) Heft 12/86, S. 34; (3) Apple II+ (natürlich auch IIe/c); (4) DOS 3.3 oder ProDOS; (4) RUN DHGR.GENAU; (5) DHGR-Anzeige muß jeweils mit ESC abgebrochen werden.

T 067 LIBRARY1.TEXT
T 060 LIBRARY2.TEXT
(1) Erweitertes UCSD-Library-Programm, das weitgehend der SYSTEM.LIBRARY entspricht; (2) Heft 10/86, 45 (nicht gelistet!); (3) Apple II+/e/c; (4) Apple-Pascal 1.1/1.2; (5) Dateien müssen zunächst mit GETDOS (von Disk #18) auf Ihre Pascal-Arbeitsdisk konvertiert werden.

T 006 AUFSPALTEN.TEXT
(1) Utility zum Aufspalten großer Pascal-Quelltexte, die mit anderen Editoren erstellt wurden (z.B. LIBRARY1.TEXT usw.); (2) Heft 12/86, S. 31; (3) Apple II+/e/c; (4) Apple-Pascal 1.1/1.2; (5) Datei muß zunächst mit GETDOS (von Disk #18) auf Ihre Pascal-Arbeitsdisk konvertiert werden.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg 1

Programming Toolkits für Kyan-Pascal 2.0

Toolkit I: System Utilities: Clubpreis DM 118,–, Normalpreis DM 148,– (lieferbar)

Toolkit II: Mouse Text: Clubpreis DM 118,–, Normalpreis DM 148,– (lieferbar)

Toolkit VI: Turtle Graphics: Clubpreis DM 68,–, Normalpreis DM 88,– (lieferbar)

Toolkit III: Advanced Graphics: Clubpreis DM 118,–, Normalpreis DM 148,– (lieferbar)

Toolkit V: Mouse Graphics: Clubpreis DM 158,–, Normalpreis DM 198,– (in Vorbereitung)

Toolkit VI: Code Optimizer: Clubpreis DM 298,–, Normalpreis DM 348,– (lieferbar)

KIX: Clubpreis DM 98,–, Normalpreis DM 118,– (lieferbar)

Kyan-Ordner: Leerer Ordner für Utility-Anleitungen, DM 19,–

Alle Utilities werden als teils beidseitig bespielte Disketten geliefert, die neben den Include-Files (meist Quelltexte) diverse Demos enhalten. Die Anleitungen selbst sind Loseblattlieferungen (z. B. bei den System Utilities 52 Druckseiten), die für den grauen Ordner von Kyan 2.0 bestimmt sind. Zum Club-Preis werden nur Mitglieder des Kyan-Clubs beliefert.

Toolkit I: System Utilites
Diese Utilities decken verschiedene Bereiche ab:
Routinen für ProDOS-Funktionen, 18 Treiberroutinen für Maus und Joystick, diverse Routinen zur Bildschirmsteuerung (Scrollen, Tab, Inverse etc.), Routinen zur Erzeugung von Zufallszahlen, Zahlenkonvertierungsroutinen: Real-String, String-Real, Integer-String, String-Integer, Routinen zum alphabetischen und numerischen Sortieren und Mischen von bis zu 5 Dateien, Line Parsing Routine.

Toolkit II: Mouse Text
Diese Utilities umfassen mehrere Dutzend Befehle für Fenstertechnik, die Ihre Kyan-Pascal- und Assemblerprogramme um Macintosh-ähnliche Features erweitern. Im einzelnen bietet Mouse Text Cursor-Befehle, Interrupts, Menü-Befehle, Kontrollbefehle und spezielle Befehle zur Erstellung und zum Arbeiten mit Bildschirmfenstern.

Toolkit III: Advanced Graphics
Dieses Toolkit besteht aus den beiden Modulen Graphics Primitives und Advanced Graphics. Graphic Primitives enthält Assemblerprozeduren und -funktionen für Initialisierungsbefehle, Graph-Port-Befehle, Grundfunktionen zur Erzeugung grafischer Darstellungen (Linien, Rechtecke, Flächen zeichnen usw.) und Textbefehle.
Advanced Graphics bietet Befehle zur Erstellung dreidimensionaler Grafik. Sie unterstützen ein- und mehrfarbige Grafiken. Ein Objekt kann aus einem beliebigen Winkel und aus beliebiger Entfernung betrachtet und im Raum gedreht werden.

Toolkit IV: Turtle Graphics
Diese Diskette enthält diverse Hires- und Ton-Routinen.
Verschiedene Turtle-Befehle nutzen die Möglichkeiten der Turtle-Grafik, 4 Prozeduren erzeugen unterschiedliche Geräuscheffekte. Erstellung von Balkendiagrammen dienen die Prozeduren Bar-Chart, Pie-Chart und Plot-x-y.

Toolkit VI: Code Optimizer
Der Code Optimizer macht den vom Pascal-Compiler erzeugten Code wesentlich schneller und kürzer. Er optimiert den compilierten Assembler-Quellcode so, daß nur die im Programm verwendeten Segmente der Runtime-Library in das Quellprogramm eingebunden werden; zahlreiche Macroaufrufe werden zusammengefaßt. Das so entstandene optimierte Assemblerprogramm wird zu einem ausführbaren Maschinencode assembliert.
Interessant für fortgeschrittene Programmierer: Der Code Optimizer enthält den Quelltext der gesamten Kyan-Pascal Runtime-Library.

KIX
KIX erweitert das ProDOS-Betriebssystem um eine RAM-residente, standardisierte Benutzeroberfläche, die dem UNIX-Betriebssystems ähnelt.
KIX enthält eine Sammlung externer Befehle zur Verwaltung von Directories, Files, Disketten und weitere spezielle Befehle. KIX ist kompatibel mit allen ProDOS-Funktionen und nahezu jeder auf ProDOS basierenden Software. Die KIX-Shell stellt die Verbindung zwischen dem Benutzer, den Benutzerfiles und der Hardware her. Sie interpretiert Befehle, ruft die erforderlichen Arbeits- und Hilfsprogramme auf und arbeitet mit dem ProDOS-Kern, dem inneren Teil des Betriebssystems, zusammen.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg

Der Matrixdrucker Epson EX-800

Ein Erfahrungsbericht

von Ulrich Tönnies

Der Epson EX-800 ist ein schneller 9-Nadel-Matrixdrucker, der zusätzlich schönschriftfähig ist. Er wird für ca. DM 2100.– angeboten und könnte in Zukunft die FX-Drucker des gleichen Herstellers ersetzen. Erste Erfahrungen mit dem EX-800 werden im folgenden Text beschrieben.

1. Technische Ausstattung

Der 9-Nadel-Matrixdrucker EX-800 wird als besonders schneller Drukker angepriesen. Laut Herstellerangaben druckt er mit bis zu 300 Z/s in der Schriftart Elite, in Pica noch mit immerhin 250 Z/s. Der Drucker ist zudem NLQ-fähig; in diesem Druckmodus soll er noch 50 Z/s schaffen. Die Ergebnisse meiner eigenen Messungen zur Druckgeschwindigkeit sind im Abschnitt 7 wiedergegeben.
Der Drucker ist serienmäßig mit Centronics-Parallelschnittstelle sowie mit der seriellen RS-232-Schnittstelle (6polige DIN-Buchse) ausgestattet, weitere Interfaces können zusätzlich eingebaut werden (z.B. IEEE 488).

Der EX-800 ist IBM-kompatibel, d.h. er hat neben dem normalen Epson-Modus noch die Fähigkeit, IBM-Matrixdrucker zu emulieren, dann fallen jedoch einige der im Epson-Modus vorhandenen Fähigkeiten weg. Mit einer zusätzlichen Vorrichtung kann der neue Epson auch farbig drucken. Ein Einzelblatt-Aufsatz ist als Sonderzubehör zu haben (Preis um DM 500.–). Serienmäßig besitzt der Drucker einen Schiebetraktor (Stachelwalze) sowie eine umsteckbare Papierführung, mittels derer man leicht einzelne Papierbögen einziehen lassen kann. Dabei muß man jedes Blatt einzeln an der Führung in den Drucker gleiten lassen. Den eigentlichen Einzug übernimmt dann der Drucker, nachdem ein spezieller Hebel betätigt wurde. Dieser Vorgang läuft zuverlässig ab, auch bei einem Ausdruck, der über mehrere Druckseiten geht: Wenn eine Seite beschrieben ist, stellt sich der Drucker „OFFLINE", man läßt das nächste Blatt einziehen und stellt den Drucker wieder „ONLINE", worauf der Ausdruck fortgesetzt wird.
Der EX-800 besitzt einen internen Datenpuffer von 8K Größe, es sind auch 32K-Puffer-Schnittstellen lieferbar. Dies kostet aber etwa DM 400,– Aufpreis. Wenn man soviel zahlt, wird man ggfs. einen noch größeren externen Puffer bevorzugen.

2. Steuerbefehle und Kompatibilität

Laut Handbuch ist die Menge der Steuerbefehle des EX-800 eine Obermenge der Steuerbefehle des FX-85, d.h. man sollte einen FX-85 ohne irgendwelche Softwareanpassungen durch den EX-800 ersetzen können. Dies konnte nicht geprüft werden, da der EX-800 als Ersatz für einen (zu langsam gewordenen) FX-80 des Autors eingesetzt wurde. Zum FX-80 ist der neue EX-800 aber nicht hundertprozentig kompatibel. Insbesondere die beim FX-80 und auch beim EX-800 gegebenen Schriftwahl-Steuersequenzen über „ESC ! nnn" sind nicht kompatibel. Wer in bestehender Software mittels dieser Sequenz zwischen verschiedenen Schriftarten wechseln konnte, muß wahrscheinlich Anpassungen an der Software vornehmen. Dagegen funktionieren die übrigen Sequenzen auf beiden genannten Druckern nahezu gleich (Randeinstellung, eigenen Zeichensatz per „download"-Sequenz in den Drucker bringen, Grafiksequenzen etc., aber auch die nicht mittels „ESC !" eingeleiteten Schriftwahlsequenzen). Hier dürften keine Probleme zu erwarten sein.

3. Schriftarten

Der EX-800 verfügt über die bekannten Schriftvarianten Pica und Elite in den Formen schmal, normal und breit. Er kann darüber hinaus zusätzlich alle Schriftarten kursiv wiedergeben, die Schriftarten Pica und Elite auch im Proportionaldruck.
Gewisse Einschränkungen sind hinsichtlich der Kombinationsmöglichkeiten gegeben, jedoch hält der EX-800 eine Vielzahl von Varianten bereit. (**Abb. 1**)
Daneben bietet er zwei Schönschriften (NLQ-Druck): Roman und Sans Serif. Die Roman-Schrift erinnert sehr an die Schreibmaschinenschrift einer alten Maschine. Dagegen ist Sans Serif eine glattere, moderne Schriftvariante. Beide Schriften werden im Doppeldruck erzeugt, dadurch wird der EX-800 während des Druckens doch wesentlich langsamer. Zwar können beide Schönschriften ihre Herkunft von einem Matrixdrucker nicht verleugnen, andererseits bieten sie – v.a. bei frischem Farbband – doch ein schönes Schriftbild (**Abb. 2**). Die NLQ-Schriften sind ebenfalls in Normal-, Schmal-, Proportionalschrift etc. möglich.

4. Handbuch

Das mitgelieferte, ca. 250 Seiten starke deutschsprachige Handbuch ist – verglichen mit dem Handbuch zum alten FX-80 – eine Enttäuschung. Besonders die im FX-80-Handbuch reichlich vorhandenen kurzen Demo-Listings mit Demo-Ausdruck fehlen bis auf we-

EX-800 Druckvariante "ESC ! 12"
EX-800 Druckvariante "ESC ! 13"
EX-800 Druckvariante "ESC ! 14"
EX-800 Druckvariante "ESC ! 15"
EX-800 Druckvariante "ESC ! 20"
EX-800 Druckvariante "ESC ! 21"
EX-800 Druckvariante "ESC ! 22"
EX-800 Druckvariante "ESC ! 23"
EX-800 Druckvariante "ESC ! 24"
EX-800 Druckvariante "ESC ! 25"
EX-800 Druckvariante "ESC ! 90"
EX-800 Druckvariante "ESC ! 91"
EX-800 Druckvariante "ESC ! 92"
EX-800 Druckvariante "ESC ! 93"

Abb. 1: Auswahl aus den Druckvarianten

nige Ausnahmen fast völlig. Allerdings kann man auch mit dem EX-800-Handbuch nach kurzer Zeit einigermaßen zurechtkommen. Dabei muß man aber in Kauf nehmen, gewisse Steuersequenzen erst einmal durchzuprobieren, um deren Ergebnis einschätzen zu können.

5. Bedienung

Die Bedienung des EX-800 wird durch ein Bedienfeld mit Folientasten erledigt. Man wählt dort die Schriftqualität zwischen Entwurf, NLQ Sans Serif und NLQ Roman, die Schriftvariante zwischen Pica, Elite und Proportional und die Schriftgröße zwischen normal und schmal aus. Unmögliche Kombinationen werden vom EX-800 durch mehrmaliges Piepsen angezeigt, erfolgreiche Umstellungen in der Regel mit einem einfachen „Piep" quittiert.

Für den Benutzer zunächst äußerst undurchsichtig ist das Verhalten des Druckers auf Steuersequenzen (nach dem Motto: „mal macht er das, dann macht er jenes"). Z.B. können einige der obengenannten Schriftwechsel mittels der „ESC ! nnn"-Sequenz die am Bedienfeld eingeschalteten Schriftvarianten nicht so ohne weiteres überschreiben. Nach längerem Lesen im Handbuch findet man zwar auch heraus, warum das so ist; verwirrend bleibt es aber allemal.

Die Bedienung über die Folientastatur ist für einen unerfahrenen Benutzer sicher ideal, ein Programmierer sollte sich aber das Handbuch konsequent aneignen, wenn er die Steuerung des EX-800, die im übrigen auch vom Programm aus vollständig möglich ist, richtig bewältigen will. (So gibt es zwar z.B. einen „Esc ! nnn"-Befehl für NLQ, hatte man aber vorher „Draft" eingeschaltet, muß man trotzdem noch einen anderen Befehl für NLQ senden).
Über DIP-Schalter (2 x 8) auf der Rückseite des EX-800 werden der beim Einschalten serienmäßige Zeichensatz (ASCII, Deutsch,...), das CR-LF-Verhalten, die Baudrate und Eigenschaften der RS-232 etc. eingestellt. Da diese Einstellungen sicherlich nicht allzu oft notwendig sind, ist die doch recht schlechte Zugänglichkeit nicht allzu nachteilig zu bewerten.

6. Sondereigenschaften

6.1. Dump-Data-Modus

Werden während des Einschaltens des Druckers die FORMFEED- und LINEFEED-Tasten gleichzeitig gedrückt gehalten, geht der Drucker in den Data-Dump-Modus. Er druckt jetzt jedes ankommende Zeichen als HEX-Angabe (16 pro Zeile) und am Ende der Zeile noch die druckbaren ASCII-Äquivalente. Das ist ganz nützlich, falls man sich Klarheit über auftretende Unregelmäßigkeiten verschaffen will.

6.2. Textverarbeitung

Der EX-800 bietet einen Steuerbefehl „Textverarbeitung" (ESC a nn) mit den Druckoptionen „linksbündig, zentriert, rechtsbündig" und „Blocksatz". Diese Eigenschaft hebt ihn von vielen Drukkern ab. Man kann z.B. den entsprechenden Steuerbefehl an den Drucker senden und dann Texte (bei Blocksatz) ohne CR/LF an den Drucker schicken: Der EX-800 formatiert sie automatisch (CR/LF muß nur z.B. für neue Absätze angegeben werden).

Da dies bei allen Schriftvarianten funktioniert (auch bei Proportionalschrift), bietet sich der EX-800 als echte Alternative für alle Appleworks-Benutzer an, die sich aus irgendwelchen Gründen keinen Imagewriter zulegen wollen (meines Wissens funktioniert bei Appleworks der proportionale Ausdruck im Blocksatz nur mit dem Imagewriter). Wenn auch vielleicht nicht die ‚normale' Vorgehensweise (Text eintippen und ausdrucken lassen) möglich ist, so sollte doch immer das Abspeichern von Texten als ASCII-Datei funktionieren; von dort kann der Text dann ja durch ein einfaches Basicprogramm an den Drucker übertragen werden. (Hinweis: Dabei muß die Druckerschnittstelle oder die Basic-Version so konfiguriert werden, daß nicht automatisch – z. B. alle 80 Zeichen – ein CR/LF zusätzlich eingefügt wird, sonst klappt es nicht.) Die Formatierung erfolgt dann erst durch den Drucker. Auch Wordstar-Benutzer haben unter Apple-CP/M offenbar nicht die Möglichkeit, Texte in Proportionalschrift im Blocksatz zu drucken.
Kurz und gut: All diesen kann der EX-800 helfen. Wegen der dann immer noch fehlenden Umbruchfunktion wird man evtl. auf Kopf- und Fußzeilen etc. verzichten müssen, aber die Druckergebnisse sind doch schon recht zufriedenstellend.

7. Druckgeschwindigkeit

Seit langem ist bekannt, daß die Hersteller von Matrixdruckern bei der Angabe der Druckgeschwindigkeit einen Hang zur Übertreibung besitzen. Über ihre Meßmethoden zur Ermittlung der Druckgeschwindigkeit ist nicht viel bekannt, so daß man sich in dieser Hinsicht nur auf den subjektiven Eindruck oder Praxistests verlassen kann.

```
Entwurf Pica

Der Epson EX-800 ist ein schneller
9-Nadel-Matrixdrucker,          der
zusätzlich schönschriftfähig ist. Er

NLQ Sans Serif Pica

Der Epson EX-800 ist ein schneller
9-Nadel-Matrixdrucker,          der
zusätzlich schönschriftfähig ist. Er

NLQ Roman proportional
Der Epson EX-800 ist ein schneller
9-Nadel-Matrixdrucker, der zusätzlich
schönschriftfähig ist. Er wird für ca. DM

NLQ Roman Elite
Der Epson EX-800 ist ein schneller
9-Nadel-Matrixdrucker,          der
zusätzlich schönschriftfähig ist. Er
```

Abb. 2: Schriftarten des EX-800

Subjektiv erscheint der EX-800 im Vergleich zu einem FX-80 zunächst einmal ‚sehr schnell'. Das fällt besonders dann auf, wenn man z.B. den internen Buffer im OFFLINE-Zustand voll werden läßt, dann auf ONLINE schaltet und den Ausdruck beobachtet. Auch beim Grafikdump, der normalerweise bidirektional, per Steuerbefehl auch unidirektional abläuft, ist der Unterschied auffällig.

Am besten läßt sich die wirkliche Druckgeschwindigkeit mit Praxistests bestimmen. Gibt man beispielsweise eine Reihe von Zeilen mit jeweils 80 „A" aus, so erzielt man in Elite-Entwurfschrift eine Druckgeschwindigkeit von ca. 175 Z/s, in Pica ca. 150 Z/s. Das sind zwar nicht die angepriesenen 300 bzw. 250 Z/s, aber immerhin deutlich mehr als die ca. 90 Z/s, die ein FX-80 unter ähnlichen Umständen ausgibt.
Im NLQ-Modus erreicht man mit obigem Test unter Elite knapp 50 Z/s, unter Pica gut 40 Z/s.

Ein anderes Beispiel aus der Praxis: Für den während einer technischen Berechnung entstehenden „Zahlenfriedhof" samt 8 Grafikdumps mit einer Auflösung von 512 x 390 Pixeln brauchte der FX-80 18 Minuten reine Druckzeit. Der EX-800 schafft es in 7 Minuten, also rund 2,5mal so schnell. (Anmerkung: Der Rechner benötigte ca. 2 Minuten für Berechnungen und Dump; in den obigen Zeitangaben ist die Rechenzeit nicht enthalten, da die Ausgabe über einen Druckerpuffer erfolgte.)

8. Lautstärke

Der EX-800 ist kein leiser Drucker. Zwar erscheint das „Nadeln" subjektiv nicht lauter als z.B. beim FX-80, jedoch kommen andere Geräusche hinzu: Der Druckkopfschlitten wird mit dem darauf aufsitzenden (übrigens nicht FX-kompatiblen) Farbband wegen der enormen Zeilenwechselzeiten stark beschleunigt und wieder gebremst. Dies läßt auch die Umgebung mitschwingen (die Stärke der Schwingungen hängt von der Stabilität des Tisches ab, auf dem der Drukker steht) und bewirkt eine ziemlich störende „rappelnde" Geräuschkulisse.

Man sollte den Drucker am besten auf eine dämpfende Unterlage stellen, das ist recht nützlich und verringert die Vibrationen in der Umgebung. Dann kann man auch den „Kaffeetasse-Rappeltest" (vgl. Peeker 8/86, S.57) riskieren, bei dem gleiche Zeilen mit einem „." am Anfang und 79 folgenden Spaces ausgegeben werden. Dabei bringt der EX-800 ca. 500 Z/s

aufs Papier. Anstelle des „Nadelns" hört man dann nur noch ein „Rappeln".

9. Gesamteindruck

Der EX-800 ist das Richtige für die Leute, die bisher häufig auf ihren Drucker gewartet haben. Man vergleiche obiges Praxisbeispiel: 2 Minuten Rechenzeit und 18 Minuten Druckzeit sind eigentlich kein tragbares Verhältnis mehr, denn es bedeutet letztlich, daß 90 Prozent der nutzbaren Rechenkapazität eines Computers nur mit „Warten" verschwendet werden.
Wer solche Probleme kennt, darf bei der vorgelegten Geschwindigkeit erleben, daß nicht mehr unbedingt der Rechner auf den Drucker wartet, sondern umgekehrt gelegentlich auch der Drucker auf den Rechner. Der EX-800 ist auch in der Lage, einen sauberen Brief zu schreiben, kann dabei aber seinen 9-Nadel-Druckkopf nicht ganz verleugnen. Die Domäne des EX-800 ist das schnelle Schreiben. Einige weitere Features (z.B. „Textverarbeitung") sind ebenfalls Argumente für den Kauf eines EX-800.
Man darf annehmen, daß in absehbarer Zeit die Druckgeschwindigkeit eines EX-800 Standard wird. Es gibt ja bereits Drucker, die zu vergleichbaren oder sogar niedrigeren Preisen ähnliche Leistungen bieten (Imagewriter II, speziell wohl für die Apple-Welt oder einige japanische Modelle).

Ausschlaggebend für einen Kauf dürfte der gewünschte Verwendungszweck sein: Für Apple-Computer dürfte der Imagewriter II (vgl. Test in Peeker 8/86) sicher der geeignetere Drucker sein, da er in aller Regel besser zu den auf Apple-Computern laufenden Programmen paßt. Bei anderen Computern, z.B. in der HP- oder IBM-Welt, wird man möglicherweise den EX-800 vorziehen.

Neben Überlegungen zum reinen Kaufpreis sollte man dabei nicht die Kompatibilitätsfrage übersehen: ein „billigerer" Drucker braucht nicht der „preiswertere" zu sein: In das Umschreiben von Programmen muß man schnell mehr investieren, als man beim Kaufpreis eingespart hat.
In dieses Konzept scheint der EX-800 genau hineinzupassen: Bisher schon vorhandene (langsamere) Epson-Drucker kann er dank seiner weitgehenden Kompatibilität ohne allzu große Probleme ersetzen, durch seine IBM-Emulation paßt er aber auch gut in die MS-DOS-Welt.

Statistik in Basic

Ein Kurztest

von Dagmar Berberich

1. Lieferumfang

Als „PC-Combo" bezeichnet der Hofacker-Verlag in Holzkirchen seine Kombinationsangebote von Computerbüchern mit Begleitdiskette. In dieser Form ist zum Preis von DM 79,– auch das Paket „Statistik in Basic – Einführung, Praktische Anwendungen, Programmbeispiele" erschienen.
Zu diesem Paket gehört eine einseitig bespielte, nicht kopiergeschützte Diskette und ein 214 Seiten starkes Buch, das auch ohne Diskette zum Preis von DM 39,– zu beziehen ist. Buch und Diskette sind in einer Plastikbox verpackt, die dem Paket offenbar ein etwas „profihafteres" Aussehen verleihen soll.

2. Hardware

„Statistik in Basic" ist für den Apple IIc und IIe bestimmt. Schon im Vorwort des Buches wird darauf hingewiesen, daß alle Programme in Microsoft-Basic auf dem Apple II und dem IBM-PC geschrieben und über den Apple II ausgedruckt wurden. Mit Ausnahme der Grafikprogramme (Balken-, Strich-, Torten- und Polygonzugdiagramme) wurde rechnerunabhängig programmiert, so daß die Programme auf allen Basic-Rechnern laufen sollten. Welche Änderungen dabei evtl. nötig sind und wie umfangreich diese wären, konnte nicht nachgeprüft werden.

3. Die Begleitdiskette

Beim Booten der Begleitdiskette erscheint zunächst nur die lapidare Mitteilung „STATISTIK DISK". Nach dem CATALOG-Befehl erhält man eine Übersicht über die abgespeicherten Programme, die man mit dem üblichen „RUN *Dateiname*" startet.
Die Diskette enthält die im Buch abgedruckten Programme und die Sortierverfahren, die im Anhang aufgeführt sind, in Form von Applesoft-Dateien. Die Dateinamen entsprechen den Bezeichnungen der Programme im Buch.

4. Das Buch

Das Buch enthält eine Sammlung der wichtigsten statistischen Verfahren und die zugehörigen Basicprogramme mit vollständigen Listings.
Es versteht sich *nicht* als Lehrbuch der Statistik; die begleitende Theorie zu den Programmbeispielen ist sehr kurz gehalten und beschränkt sich auf die Wiedergabe der mathematischen Grundlagen.
Der Autor will in 16 Kapiteln zeigen, wie man auf einfache Weise statistische Berechnungen in Basic programmieren kann. Jedes Programm wird von einem Anwendungsbeispiel begleitet.
In den Anfangskapiteln werden einige grundlegende statistische Elemente wie Häufigkeiten und Klassenbreiten berechnet und verschiedene Diagrammtypen der beschreibenden Statistik in Form von Grafiken erstellt. Die folgenden Abschnitte beschäftigen sich mit statistischen Kennzahlen, Wahrscheinlichkeitsrechnung, Markov-Analyse, diskreter statistischer und Normalverteilung (Gauß-Verteilung). Es folgen statistische und parameterfreie Testverfahren, Berechnungen zur Varianzanalyse, zur linearen, nichtlinearen und mehrfach linearen Regression und Autokorrelation. Zwei Kapitel über ‚Vorhersagen und Trendberechnungen' und ‚Statistik und Glücksspiel' schließen den Hauptteil des Buches ab. Im Anhang werden sechs verschiedene Sortierverfahren und die Gammafunktion aufgelistet und kurze Erläuterungen zur Datenspeicherung auf Diskette und zum Zeichnen von 2 Polygondiagrammen (auf IBM-PC) gegeben.

Fazit

Wer als Basicprogrammierer Datenmaterial statistisch aufbereiten möchte, ist mit dem Paket sicher gut bedient. Ihm stehen mit Buch und Diskette eine breite Auswahl der gängigen statistischen Verfahren für eigene Anwendungen zur Verfügung.
Wer sich dagegen mit den theoretischen Grundlagen der Statistik näher befassen möchte, sollte zu einem Lehrbuch greifen. Sicher ist es auch dann motivierend, theoriebegleitend Anwendungsberechnungen durchführen zu können. Mit DM 79,– ist „Statistik in Basic" aber nicht gerade preiswert.
Wer sich DM 40,– sparen möchte und gewissenhaft Programme eingeben kann, sollte sich vielleicht nur das Buch kaufen und dann die Programme, die er wirklich benötigt, von Hand eingeben.

Bücher

Hardware-Erweiterungen für den Apple

Zusatzschaltungen für den Apple II, Apple IIe und kompatible Computer
von O. Merker
1986, 280 S., kart., DM 48,–
Franzis-Verlag, München
Gliederung
Wie funktioniert ein Computer? – Grundplatinen – Digitale Ein/Ausgabekarten – Analoge Ein/Ausgabekarten – Zeit-Erfasssung – Tonerzeugung – Universal-Funktionskarten – Nichtflüchtige Datenspeicher – Anwendungen – Literatur
Bemerkungen
Dieses Buch eröffnet dem Programmierer die Welt der Meß-, Steuer- und Regeltechnik. Es werden verschiedene Hardware-Zusätze vorgestellt, die Meß-, Regel- und Steueraufgaben und der hardwaremäßigen Tonerzeugung dienen. Eine Centronics-Schnittstelle zum Anschluß eines Druckers und eine Language Card werden ebenfalls beschrieben. In allen Fällen sind genaue Bauanleitungen mit Platinenvorlagen angegeben. So wird gleichzeitig die Funktion der einzelnen Schaltungen erklärt. Mit den beschriebenen Erweiterungen wird aus dem Apple II ein professionell einsetzbares System.

Textverarbeitung mit Home und Personal Computern

Systeme, Vergleiche, Anwendungen
von A. Görgens
1986, 125 S., kart., DM 29,80
Falken-Verlag, Niedernhausen
Gliederung
Einsatzanalyse – Hardware – Software – Peripherie
Bemerkungen
Wer nach einem geeigneten Textverarbeitungsprogramm sucht oder wissen möchte, ob sich Textverarbeitung mit dem Computer für ihn überhaupt lohnt, findet in diesem Buch praxisgerechte Tips und aktuelle Vergleichsanalysen. Die wichtigsten Rechner am Markt, von Commodore 64 bis IBM AT, ihre Betriebssysteme und Textverarbeitungssoftware werden vorgestellt. Das Buch mit seinen leichtverständlichen Definitionen versteht sich als praktischer Ratgeber in Sachen Textverarbeitung.

CP/M ständig im Griff

von R. Weiss
1986, 91 S., Spiralheftung, DM 21,80
Dr. Alfred Hüthig Verlag, Heidelberg
Gliederung
Das Betriebssystem CP/M – Die residenten Kommandos – Die wichtigsten transienten Kommandos
Bemerkungen
Sinn und Zweck dieses Taschenbuches ist die ständige Verfügbarkeit von Beschreibungen der am häufigsten verwendeten CP/M-Kommandos. Es bietet sich daher zur Schnellinformation als „Nachschlagefibel" an, die neben der Tastatur ihren Platz findet. Bei den Erläuterungen zu den Befehlen wird auf eventuelle Abweichungen zwischen den Betriebssystemen CP/M-80, CP/M-86 und CP/M-Plus (CP/M 3.0) eingegangen.

Das UNIX System

von S. R. Bourne
1985, 467 S., kart., DM 58,–
Addison-Wesley Verlag (Deutschland) GmbH, Bonn
Gliederung
Einführung in UNIX – Der Einstieg – Das Editieren von Dateien – Die Shell – Die Programmiersprache C – UNIX-Systemprogrammierung – Dokumentaufbereitung – Werkzeuge zur Datenbearbeitung – Kommandos – Systemaufrufe – C-Funktionen – adb-, ed-, sh-, troff-, vi-Anweisungen – Makrobücherei – Allgemeiner Anhang
Bemerkungen
Dieses Buch ist die deutsche Erstausgabe des amerikanischen UNIX-„Klassikers" von S. R. Bourne. Es stellt sowohl für Einsteiger als auch für Experten ein umfangreiches Lehrbuch und Nachschlagewerk dar, in dem Erläuterungen und praktische Hinweise zum Einsatz des Betriebssystems UNIX enthalten sind. Das Buch beinhaltet die Einführung in die wichtigsten Komponenten von UNIX, das Datensystem, die Texteditoren und die Kommandosprache Shell. Die weiteren Kapitel über die Programmiersprache C, die UNIX-Systemprogrammierung und die Dokumentaufbereitung vertiefen das Wissen. Beispiele im Text erläutern praxisnahe Techniken und demonstrieren den Einsatz von UNIX. Die ausführlichen Anhänge enthalten das Wichtigste für den tägliche Gebrauch des UNIX-Systems.

Grundwissen Informationsverarbeitung

von H. Schiro
1986, 311 S., geb., DM 58,–
Falken-Verlag, Niedernhausen
Gliederung
Entwicklung und Bedeutung der Datenverarbeitung – Grundbegriffe der Datenverarbeitung – Struktur eines Datenverarbeitungssystems – Hardware – Software – Von der Aufgabe zum Programm – Bedeutung der Betriebssysteme – Betriebsarten der Datenverarbeitung – Berufe in der Datenverarbeitung – Informationssysteme – Bürokommunikation – Neue Wege der Kommunikation: Bildschirmtext – Vermittlungs- und Speichersysteme – Datenkommunikation – Die Informationsverarbeitung der Gegenwart und Zukunft – Fachwortverzeichnis
Bemerkungen
Von der Entstehung der EDV bis hin zu den modernsten Kommunikationsmitteln wie Bildschirmtext, Datennetzstrukturen und Telekommunikation behandelt der Autor auf anschauliche und verständliche Weise alle wesentlichen Aspekte der elektronischen Datenverarbeitung. Komplizierte Sachverhalte werden für jedermann verständlich erklärt, eine reiche Bebilderung veranschaulicht Sachzusammenhänge. Das Buch verschafft seinen Lesern ein solides Basiswissen über das Gebiet der Informationsverarbeitung.

Macintosh-Programmierhandbuch

von D. A. Lien
1986, 444 S., kart., DM 59,–
te-wi Verlag, München
Gliederung
Teil A (Lehrbuch für MS-BASIC 2.0): Wir fangen an – Einfache BASIC-Operationen – Zeichenketten (Strings) – Rechengenauigkeit und Mathematik – Grafiken – Formatierung der Ausgabe – Bereiche (Arrays) – Dateien, Grafiken und vieles anderes mehr – Programmiertechniken – Teil B: Lösungen zu den Übungen – Teil C: ROM-Routinen – Teil D: Anhänge – Teil E: Stichwortverzeichnis
Bemerkungen
Dieses Buch bringt die leicht erlernbare Programmiersprache MS-BASIC mit dem benutzerfreundlichen, komfortablen Apple-Macintosh zusammen. Die besonderen Stärken des Macintosh wie Maus, Fenstertechnik, Jalousie, Grafikroutinen etc. nimmt MS-BASIC in sein Befehlsrepertoire auf. Der Hauptteil des Buches stellt ein Lehrbuch für MS-BASIC dar: In 51 Kapiteln werden die Fähigkeiten der Sprache – oft mit Übungen und Programmlistings – auf dem Macintosh erklärt. Dazu gehören die Programmsynthese aus Modulen, das Einbinden von Mac-Funktionen, Gleitkommaarithmetik, Fremddateizugriffe, Programmablaufsteuerung, Peripheriebefehle, strukturierte BASIC-Programmierung ohne Zeilennummern etc.
Im zweiten Teil folgen die Lösungen zu einem Teil der Übungen. Der nächste Abschnitt behandelt Maschinenspracheroutinen zur Steuerung der Maus, zur Auswahl von Schrifttypen und -arten und zum Erstellen von Grafiken aller Art.
Der Anhang umfaßt neben den üblichen ASCII-Tabellen, Fehlermeldungen und den Listen reservierter BASIC-Wörter auch ein Kapitel zu Unterprogrammen und ein kurzes deutsch-englisches MS-BASIC-Wörterbuch.

Heimcomputer-Bastelkiste

Messen, Steuern, Regeln mit C64-, Apple II-, MSX-, TANDY-, MC-, Atari- und Sinclair-Computern
von G. A. Karl
1986, 255 S., kart., DM 39,–
Falken-Verlag, Niedernhausen
Gliederung
So ist ein Heimcomputer aufgebaut – Computer steuern und regeln – Verbindungen nach innen und außen – Bauelemente – Anwendungsbeispiele – Anhang
Bemerkungen
Dieses Buch ist für all diejenigen gedacht, die über etwas Bastelgeschick und elektronisches Verständnis verfügen und mit ihrem Heimcomputer mehr anfangen wollen. Zahlreiche Anwendungen verschiedenen Schwierigkeitsgrades mit Funktionserklärungen, Bauteilbeschreibugnen und Verdrahtungslisten werden erläutert.

Neben den Schnittstellen werden auch zahlreiche Bauelemente wie Transistoren, LEDs, D/A-Wandler, Komparatoren, Optokoppler u.a. berücksichtigt. Programmbeschreibungen, Listings und eine Einführung in die Maschinensprache der Mikroprozessoren Z80 und 6502 runden den Band ab.

Apple-Assembler lernen

Band 2: Nutzung besonderer Apple-Eigenschaften
von Dr. J. Kehrel
1986, 281 S., kart., DM 38,–
Dr. Alfred Hüthig Verlag, Heidelberg
Gliederung
Eine Scheibe bleibt nicht matt – Blockmalereien – Der kleinkarierte Apfel – Da steckt MUSIC drin – Der mobile Punkt – Und immer schön variabel bleiben – Kleinvieh macht auch Mist – Speicher-Recycling = Overlay – Blitz-Leser – Mehr PRO als CONTRA – Schnell wie der Wind – Anhänge
Bemerkungen
Das zweibändige Werk „Apple-Assembler lernen" ist ein kompletter Kurs über die Assemblerprogrammierung des 6502 und 65C02 auf dem Apple II. Der zweite Band stellt Programme und Unterroutinen vor, um fast alle grundlegenden Probleme auf dem Apple in Maschinensprache zu lösen. Dazu gehören Ein- und Ausgabeoperationen wie die formatierte Bildschirmausgabe in 40 oder 80 Z/Z, Lores- und Hires-Grafik, Hires-Schrift durch Bit-Grafik, Hires-Fenster incl. Fensterscroll. Die Diskettenlaufwerke werden unter DOS oder ProDOS oder Nibble für Nibble am Beispiel eines schnellen Kopierprogrammes besprochen, das in einem Durchgang Disketten formatiert und Daten schreibt. Zugriffe auf die Stringverwaltung und die Fließkommaarithmetik von Applesoft und elegante Ton- und Musikeffekte werden behandelt. Dabei wird von ROM-Routinen Gebrauch gemacht, die jeweils im Zusammenhang erklärt werden. Mit „Apple-Assembler lernen" lernen Sie viel über das Innenleben des Apple und die Verbesserung von BASIC-Programmen mit Maschinensprache.
Die „Begleitdiskette zu Apple-Assembler lernen, Bd. 2" ist für DM 44,– vom Verlag zu beziehen.

Knobeleien mit dem Micro

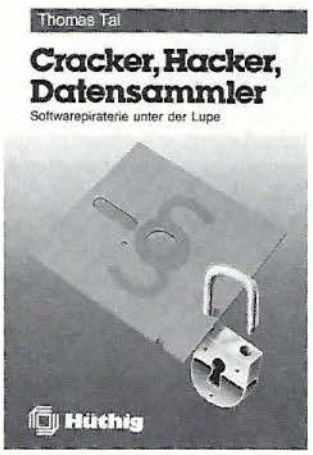
von H. Schumny (Hrsg.)
1985, 250 S., kart., DM 38,–
Vieweg- Verlag, Braunschweig
Gliederung
Münzenkombination – Gleichung – Potenzsummen-Gleichungen – Diophatische Gleichungen – Primzahlanalyse – Schachproblem – Zwei Kreise mit Schnittpunkten – Dreiecke mit Rundungen – Gittertangramme – 5 Weitere Aufgaben
Bemerkungen
Das Buch entstand aus Vorschlägen und Lösungseinsendungen zu einer Knobelecke in einem Computer-Taschenbuch des Verlags. Der erste, umfangreichste Teil umfaßt in 8 Kapiteln acht Knobelaufgaben mit 57 Lösungen, die auf 15 verschiedenen Computern erarbeitet wurden. Im Teil 2 werden in der Knobelei „Gittertangramme" 8 Fragen gestellt, von denen nur die erste schon bearbeitet ist. Der dritte Teil enthält 5 ungelöste Aufgaben, die von den Lesern erst noch herausgeknobelt werden sollen.

Aufbau und Struktur einer Datenbank mit dBase II

Anleitung zum System-Design
von R. Freshman
1985, 134 S., kart., DM 36,–
Vieweg Verlag, Braunschweig
Gliederung
Einführung – Grundbegriffe – Auswahl und Definition eines Problems – System-Design – Verzeichnis privater und geschäftlicher Adressen, Version I und II – Wie geht es weiter? – Anhänge
Bemerkungen
Dieses Buch dient als elementare Einführung in den Aufbau und die Pflege einer Datenbank mit dBASE II. Vom ersten Konzept bis zur endgültigen Programmierung wird schrittweise erklärt, welche Möglichkeiten der Benutzer hat. Der Autor beschreibt zuerst die theoretischen Konzepte einer Datenbankverwaltung, danach stellt er die Anwendung ausführlich vor. Dieser Aufbau erleichtert dem Leser das Verständnis und die Arbeit mit dBASE II.

Professioneller Datenbankeinsatz

Management und Organisation für Rechtsanwälte, Ärzte und Apotheker
von B. Lewis
1986, 164 S., kart., DM 48,–
Vieweg Verlag, Braunschweig
Gliederung
Was ist Datenmanagement? – Analyse der Büro- und Verwaltungsaufgaben – Ärzte – Zahnärzte – Rechtsanwälte – Apotheker – Anschaffungskosten für Mikrocomputersysteme
Bemerkungen
Dieses Buch ist als Einführung für künftige Benutzer eines Datenverwaltungssystems gedacht. Es erläutert Aufbau und Funktion von Datenverwaltungssystemen und die allgemeinen und branchenspezifischen Einsatzmöglichkeiten. Nach der Analyse einzelner Berufsbereiche sollte der Leser beurteilen können, ob der Einsatz eines Computers in seinem Büro sinnvoll ist. Das letzte Kapitel gibt Aufschluß über das Verhältnis von technischer Leistungsfähigkeit zu Kostenaufwand bei Mikrocomputern und über grundsätzliche Fragen, die bei einer solchen Investition zu berücksichtigen sind.



Cracker, Hacker, Datensammler

Softwarepiraterie unter der Lupe
von T. Tai
1986, 92 S., kart., DM 24,–
Dr. Alfred Hüthig Verlag, Heidelberg
Gliederung
Kopieren als Hobby – Die Gegenseite – Endstation Kadi – Hinweise für Ertappte – Die rechtliche Situation
Bemerkungen
Der zunehmenden Softwarepiraterie bei Home und Personal Computern begegnen immer mehr Softwarefirmen mit drastischen Methoden. Das Buch von T. Tai gibt keine Tips für Kopierer, es ist auch keine Werbeschrift eines Softwarehauses. Vielmehr stellt es die unterschiedlichen Standpunkte beider Seiten dar, liefert deren Argumente und vergleicht sie. Ein Kapitel über die Kampfmethoden der Firmen gegen Raubkopierer rundet die Bestandsaufnahme über den Software(schwarz)markt ab.
Der Leser erfährt in dem Buch alles über die rechtliche Situation in bezug auf den Urheberrechtsschutz von Software und über bisherige Gerichtsurteile zu dieser Problematik. Für den Fall einer Abmahnung oder Hausdurchsuchung werden Verhaltensregeln gegeben.

Applesoft BASIC Programmer's Reference Manual

The Official Publication from Apple Computer, Inc.
1985, 373 S., Spiralheftung, US$ 22,95
von S. Kamins
Addison-Wesley Verlag (USA), Reading
Gliederung
Allgemeine Information – Variablen und Arithmetik – Kontrollanweisungen – Arrays und Strings – Input/Output – Grafik – Utility-Anweisungen – Programmierung – Anhänge, Tabellen, Glossar
Bemerkungen
Das englischsprachige Buch ist das offizielle Apple-Handbuch zur Programmiersprache Applesoft-BASIC, wie sie auf den Computern der Apple-II-Reihe implementiert ist, und ist kompatibel zum Apple IIc, IIe und dem 64K II Plus. Es wendet sich an Leser, die schon Erfahrung in der Programmierung von BASIC oder einer anderen Programmiersprache haben und das Applesoft Tutorial oder ihr Computer-Handbuch bereits gelesen haben. Das Handbuch beschreibt und erläutert das gesamte Applesoft-BASIC für den Apple IIe und IIc und führt in Zusatzkapiteln in Aufbau, Planung und Effizienz der Programmierung ein. Peeks, Pokes, Speicheraufbau und Unterroutinen sind im Anhang erklärt.

Handbuch der BASIC-Dialekte

von P. Fritzsche und G. Seeßlen
1986, 478 S., geb., DM 39,80
Franz Schneider Verlag, München
Bemerkungen
Dieses Buch beschreibt 400 alphabetisch geordnete Befehle zu den BASIC-Versionen der gebräuchlichsten Mikrocomputer und erläutert anhand von kurzen Programm-

beispielen ihre Anwendungsmöglichkeiten. Zudem erhält der Leser Tips, wie er bestimmte Befehle auf Rechnern simulieren kann, in deren BASIC-Dialekt sie nicht vorgesehen sind. Eine Kompatibilitätsliste bei jedem Stichwort zeigt, auf welchen Rechnern der Befehl vorhanden ist und welche Syntax jeweils verwendet werden muß.

Professionelles Arbeiten mit dem Apple II–IIe–IIc

von G. Keeler
1986, 319 S., kart., DM 48,–
Addison-Wesley Verlag (Deutschland), Bonn
Gliederung
Teil A: Einführung – Erster Kontakt mit dem Computer – Betriebsart Direktanweisung – Variablen – Der Programmier-Modus – Die wichtigsten Elemente der Programmiersprache BASIC – Das Systemprogramm DOS – Das Editieren von Programmen – Weiterführende Programmtechniken – Strukturiertes Programmieren – Teil B: Texte (Strings) – Fehlermeldungen – Mathematische Funktionen – DOS für Fortgeschrittene – Arbeiten mit Dateien – Der Speicheraufbau des Apple – Fortgeschrittene Programmiertechniken – Graphik-Anwendungen – Die Vektortabelle – Anhänge
Bemerkungen
Dieses Handbuch beschreibt die Eigenschaften und Möglichkeiten aller Geräte der Apple-II-Familie. Teil A bietet dem Anfänger ohne Computererfahrung eine Einführung in die Nutzung des Apple und führt den Leser schrittweise zum Schreiben und Anwenden von BASIC-Programmen. Zahlreiche Übungsaufgaben machen das Buch gleichzeitig zum Lehrbuch für die Praxis. Teil B enthält eine Fülle von Informationen über Programmiertechniken und die Möglichkeiten des Apple für den fortgeschrittenen Programmierer.

Applesoft Tutorial

The Official Publication from Apple Computer, Inc.
1985, 287 S., Spiralheftung, US$ 29,95
Addison-Wesley Verlag (USA), Reading
Gliederung
Programmieren lernen – Einführung in Applesoft – Elementares Programmieren – Veränderungen – Viel über Grafik – Strings und Arrays – Zusammenfassung der Anweisungen und Befehle – Anhänge, Subroutinen, Glossar
Bemerkungen
Dieses englischsprachige Buch stellt zusammen mit der Begleitdiskette die klassische Einführung in die Programmiersprache Applesoft-BASIC dar. Das Tutorial ist eine leichtverständliche, unterhaltsame Einführung für jeden, der BASIC Schritt für Schritt kennenlernen möchte. Die Begleitdiskette enthält Beispiele für Programmiertechniken, Spiele und hilfreiche Programmierwerkzeuge. Das Applesoft Tutorial vermittelt die Grundlagen von Applesoft BASIC, ausgefeilte Techniken zur Grafikanwendung und die Prinzipien guten Programmierstils. Notwendige Gerätekonfiguration: Apple IIe oder IIc oder Apple II mit 48K RAM und Language Card.

BASIC Programmieren mit ProDOS

Das offizielle Handbuch von Apple Computer, Inc.
1986, 333 S., Spiralheftung, DM 89,50
Addison-Wesley Verlag (Deutschland), Bonn
Gliederung
Einführung – Dateien und Kommandos – Die Benutzung von Dateien – BASIC-Programmdateien – Die Programmierung mit ProDOS – Textdateien – Random-Textdateien – EXEC: Steuerung über eine Textdatei – Binärdateien – Zusammenfassung von ProDOS – DOS, ProDOS und Applesoft – Fehlermeldungen und -nummern – Extras
Bemerkungen
Dieses deutschsprachige Buch enthält eine schrittweise Anleitung zur Bedienung und Programmierung mit ProDOS, dem neuen Betriebssystem für die Computer der Apple-II-Familie, und Applesoft – von der Ausgabe einfacher Befehle wie CATALOG bis zur Behandlung von Random-Dateien. Programmierer, die mit DOS 3.3 gearbeitet haben, können sich mit den erweiterten Fähigkeiten von ProDOS vertraut machen; Neulingen bietet das Buch ein komplettes Kompendium zum Erlernen der Dateibehandlung mit einem Disketten-Betriebssystem.
Sämtliche Programmierbeispiele finden sich auf der mitgelieferten Begleitdiskette, darüber hinaus einige Dienstprogramme wie die ProDOS-Version des „Applesoft Programmer's Assistant". Das Buch setzt einen Apple IIc, IIe oder 64K II Plus voraus.

Drucker und Plotter

Text und Grafik für Ihren Computer von K.-H. Koch
1986, 192 S., kart., DM 39,–
Falken-Verlag, Niedernhausen
Gliederung
Kaufentscheidung – Die Sprache der Drucker – Nadeldrucker – Tintenstrahldrucker – Typenraddrucker – Plotter – Laserdrucker
Bemerkungen
Dieses Buch wendet sich an alle, die wissen möchten, welche verschiedenen Druckersysteme verfügbar sind, was sie leisten und welches System für welchen Anwender das richtige ist. Wer bereits einen Drucker besitzt, erfährt, wie man mit ihm arbeitet und dabei alle Möglichkeiten des Gerätes ausnutzt.

Speicheraufteilung und Tastaturbelegung beim Atari ST

von Jürgen Geiß

1. Speicheraufteilung

Abb. 1 zeigt die grobe Aufteilung des Speichers bei einem Atari 520 ST+ oder einem 1040 ST mit eingebauten ROMs. Dabei wird vorausgesetzt, daß keine RAM-Disk installiert wurde. Zu bemerken wäre noch, daß die ROM-Adressen $FC0000 bis $FC0007 in die Adressen $000000 bis $000007 gemappt werden, da sich 68000-Rechner dort den Programmzähler und den Supervisor-Stack-Pointer holen. Die genaue Belegung der I/O-Adressen ab Speicherstelle $FF8000 sowie der untere Bereich des Speichers von Adresse $000000 bis $0007FF werden in einem späteren Artikel genau untersucht.

2. Tastaturbelegung

Die Tastatur des Atari ST erzeugt nicht nur den üblichen ASCII-Code, also „65" bei einem „A", sondern auch noch einen Tastatur-Code, der *Scan-Code* genannt wird. Damit ist jeder Taste eine eindeutige Nummer zugeordnet. Es ist also möglich, Tasten abzufragen, die unabhängig von der Belegung der Tastatur sind. Z.B. könnte ein Spiel die „amerikanische" Z-Taste abfragen, die aber bei der deutschen Atari-Version ganz woanders liegt. Auf deutschen Tastaturen könnten Fingerverrenkungen die Folge sein, wenn der ASCII-Code abgefragt wird. Das kann umgangen werden, wenn der Scan-Code benutzt wird. Denn da hat die amerikanische Z-Taste den gleichen Code wie die deutsche Y-Taste ($2C). Außerdem hat nicht jede Taste einen ASCII-Code, z.B. die Funktionstasten. Um sie abzufragen, muß der Scan-Code benutzt werden.
Die folgende Tabelle ist nach dem ASCII-Code sortiert, um ein schnelles Auffinden von Zeichen mit zugehörigen Scan-Codes zu ermöglichen. ASCII-Code und Scan-Code werden in hexadezimaler (in Klam-

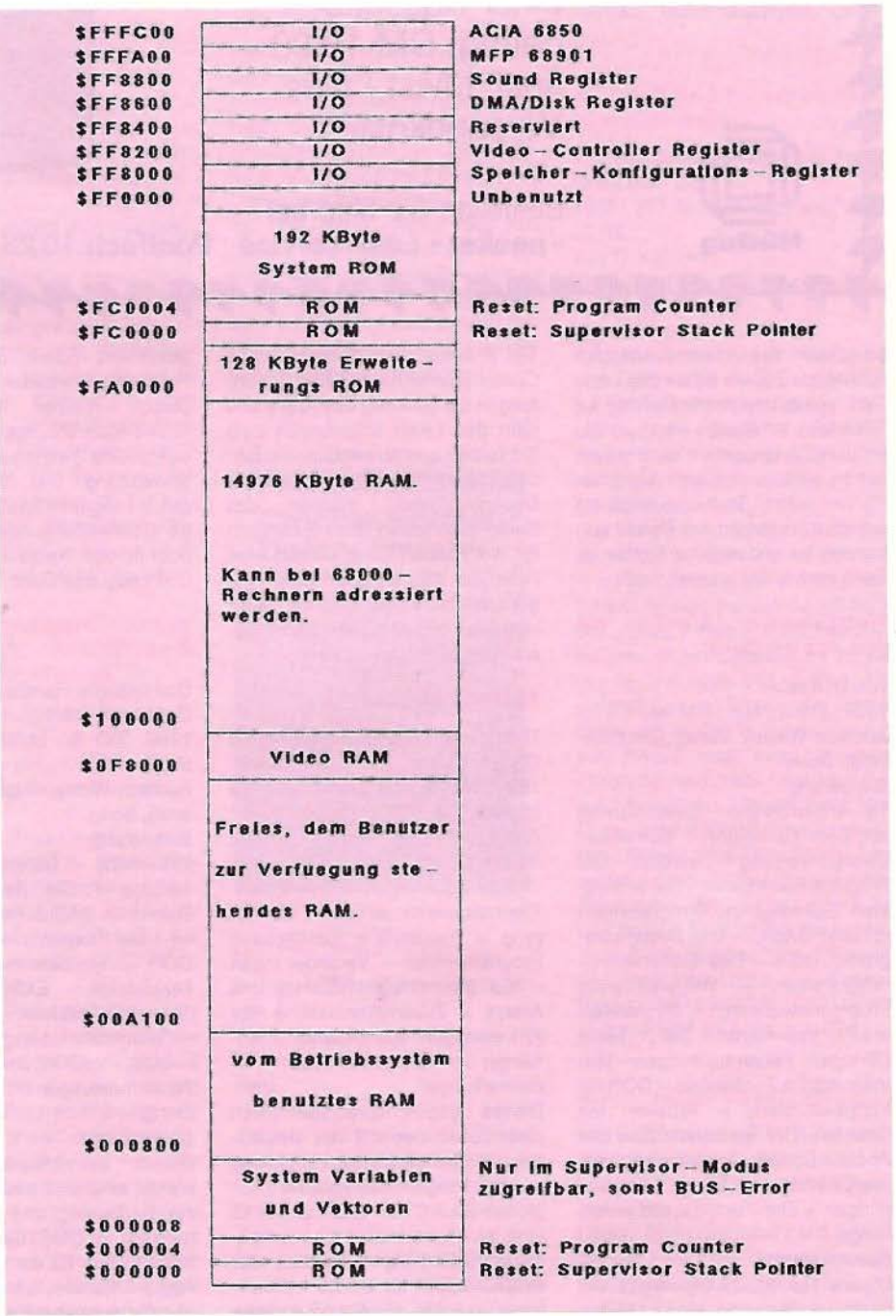


Abb. 1. Speicheraufteilung

Tabelle 1: Sortiert nach ASCII-Code

Zeichen	ASCII-Code	Scan-Code	Taste(n)
NUL	$00 (0)		
SOH	$01 (1)	$1E (30)	<Control-A>
STX	$02 (2)	$30 (48)	<Control-B>
ETX	$03 (3)	$2E (46)	<Control-C>
EOT	$04 (4)	$20 (32)	<Control-D>
ENQ	$05 (5)	$12 (18)	<Control-E>
ACK	$06 (6)	$21 (33)	<Control-F>
BEL	$07 (7)	$22 (34)	<Control-G>
BS	$08 (8)	$23 (35)	<Control-H>
HT	$09 (9)	$17 (23)	<Control-I>
LF	$0A (10)	$24 (36)	<Control-J>
VT	$0B (11)	$25 (37)	<Control-K>
FF	$0C (12)	$26 (38)	<Control-L>
CR	$0D (13)	$32 (50)	<Control-M>
SO	$0E (14)	$31 (49)	<Control-N>
SI	$0F (15)	$18 (24)	<Control-O>
DLE	$10 (16)	$19 (25)	<Control-P>
DC1	$11 (17)	$10 (16)	<Control-Q>
DC2	$12 (18)	$13 (19)	<Control-R>
DC3	$13 (19)	$1F (31)	<Control-S>
DC4	$14 (20)	$14 (20)	<Control-T>
NAK	$15 (21)	$16 (22)	<Control-U>
SYN	$16 (22)	$2F (47)	<Control-V>
ETB	$17 (23)	$11 (17)	<Control-W>
CAN	$18 (24)	$2D (45)	<Control-X>
EM	$19 (25)	$2C (44)	<Control-Y>
SUB	$1A (26)	$15 (21)	<Control-Z>
ESC	$1B (27)	$01 (1)	<Esc>
FS	$1C (28)	$	<Control-<>
GS	$1D (29)	$	<Control-§>
RS	$1E (30)	$29 (41)	<Control-↑>
US	$1F (31)	$35 (53)	<Control-_>
	$20 (32)	$39 (57)	
!	$21 (33)	$02 (2)	
"	$22 (34)	$03 (3)	
#	$23 (35)	$29 (41)	
$	$24 (36)	$05 (5)	
%	$25 (37)	$06 (6)	
&	$26 (38)	$07 (7)	
'	$27 (39)	$0D (13)	
(	$28 (40)	$09 (9)	
)	$29 (41)	$0A (10)	
*	$2A (42)	$1B (27)	
+	$2B (43)	$1B (27)	
,	$2C (44)	$33 (51)	
-	$2D (45)	$35 (53)	
.	$2E (46)	$34 (52)	
/	$2F (47)	$08 (8)	
0	$30 (48)	$0B (11)	
1	$31 (49)	$02 (2)	
2	$32 (50)	$03 (3)	
3	$33 (51)	$04 (4)	
4	$34 (52)	$05 (5)	
5	$35 (53)	$06 (6)	
6	$36 (54)	$07 (7)	
7	$37 (55)	$08 (8)	
8	$38 (56)	$09 (9)	
9	$39 (57)	$0A (10)	
:	$3A (58)	$34 (52)	
;	$3B (59)	$33 (51)	
<	$3C (60)	$60 (96)	
=	$3D (61)	$0B (11)	
>	$3E (62)	$60 (96)	
?	$3F (63)	$0C (12)	
@	$40 (64)	$1A (26)	<Altern.-@>
A	$41 (65)	$1E (30)	
B	$42 (66)	$30 (48)	
C	$43 (67)	$2E (46)	
D	$44 (68)	$20 (32)	
E	$45 (69)	$12 (18)	
F	$46 (70)	$21 (33)	
G	$47 (71)	$22 (34)	
H	$48 (72)	$23 (35)	
I	$49 (73)	$17 (23)	
J	$4A (74)	$24 (36)	
K	$4B (75)	$25 (37)	
L	$4C (76)	$26 (38)	
M	$4D (77)	$32 (50)	
N	$4E (78)	$31 (49)	
O	$4F (79)	$18 (24)	
P	$50 (80)	$19 (25)	
Q	$51 (81)	$10 (16)	
R	$52 (82)	$13 (19)	
S	$53 (83)	$1F (31)	
T	$54 (84)	$14 (20)	
U	$55 (85)	$16 (22)	
V	$56 (86)	$2F (47)	
W	$57 (87)	$11 (17)	
X	$58 (88)	$2D (45)	
Y	$59 (89)	$2C (44)	
Z	$5A (90)	$15 (21)	
[	$5B (91)	$27 (39)	<Alternate-[>
\	$5C (92)	$1A (26)	<Alternate-535>
]	$5D (93)	$28 (40)	<Alternate-]>
↑	$5E (94)	$29 (41)	
_	$5F (95)	$35 (53)	
`	$60 (96)	$0D (13)	
a	$61 (97)	$1E (30)	
b	$62 (98)	$30 (48)	
c	$63 (99)	$2E (46)	
d	$64 (100)	$20 (32)	
e	$65 (101)	$12 (18)	
f	$66 (102)	$21 (33)	
g	$67 (103)	$22 (34)	
h	$68 (104)	$23 (35)	
i	$69 (105)	$17 (23)	
j	$6A (106)	$24 (36)	
k	$6B (107)	$25 (37)	
l	$6C (108)	$26 (38)	
m	$6D (109)	$32 (50)	
n	$6E (110)	$31 (49)	
o	$6F (111)	$18 (24)	
p	$70 (112)	$19 (25)	
q	$71 (113)	$10 (16)	
r	$72 (114)	$13 (19)	
s	$73 (115)	$1F (31)	
t	$74 (116)	$14 (20)	
u	$75 (117)	$16 (22)	
v	$76 (118)	$2F (47)	
w	$77 (119)	$11 (17)	
x	$78 (120)	$2D (45)	
y	$79 (121)	$2C (44)	
z	$7A (122)	$15 (21)	
{	$7B (123)	$27 (39)	<Alternate-{>
|	$7C (124)	$2B (43)	
}	$7D (125)	$28 (40)	<Alternate-}>
~	$7E (126)	$2B (43)	
DEL	$7F (127)	$53 (83)	<Delete>
§	$DD (221)	$04 (4)	
ä	$84 (132)	$28 (40)	
ö	$94 (148)	$27 (39)	
ü	$81 (129)	$1A (26)	
ß	$9E (158)	$0C (12)	
Ä	$8E (142)	$28 (40)	
Ö	$99 (153)	$27 (39)	
Ü	$9A (154)	$1A (26)	

Man vergleiche hierzu dei ASCII-Tabelle auf Seite 18 in diesem Heft

mer in dezimaler) Schreibweise aufgeführt. Wo kein druckbares Zeichen, keine Taste oder kein Scan-Code existiert, ist das entsprechende Feld leer.

In **Abb. 2** kann umgekehrt zu jeder Taste der entsprechende Scan-Code abgelesen werden.

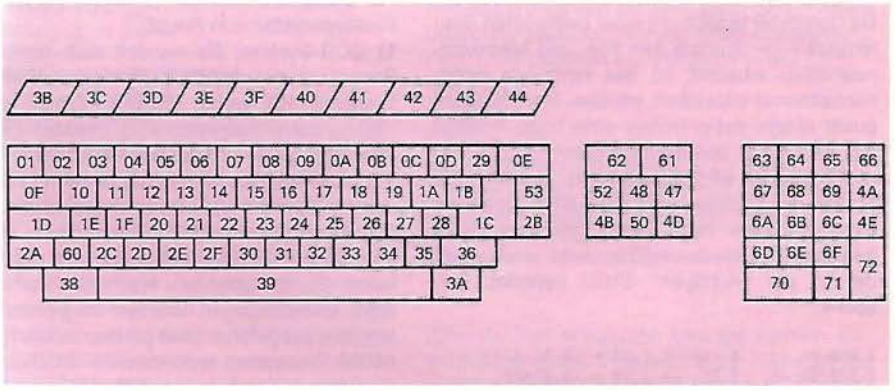


Abb. 2. Die deutsche ST-Tastatur mit ihren Scan-Codes

Binärmathematische Rundungsfehler anschaulich demonstriert

Mit GFA-BASIC-Beispielen

von Ulrich Stiehl

In diesem Beitrag soll in anschaulicher Form gezeigt werden, wie bei Computern Fließkommazahlen binär dargestellt werden und wie sich Rundungsfehler bei mathematischen Berechnungen sowie Zahlenkonvertierungen erklären lassen. Zu Demonstrationszwecken wird das GFA-BASIC-Zahlenformat herangezogen, doch gelten die Ausführungen mutatis mutandis auch für andere Fließkommaformate.

1. Normal- und Exponentialformat

Dezimalzahlen werden üblicherweise im sog. Normalformat dargestellt, welches den Dezimalpunkt an der „richtigen" Stelle beläßt, z.B. 0.1234, 1.234, 12.34, 123.4 usw. Bei kleinen oder großen Zahlen entstehen dadurch vorne oder hinten entsprechend viele Nullen, z.B. 0.00001234, 12340000.0 usw. Dies ist jedoch nur möglich, weil beim Normalformat die Anzahl der Vor- und Nachkommastellen schwanken darf.

Da Computer jedoch mit einer begrenzten Stellenanzahl (= Summe aus Vor- und Nachkommastellen) arbeiten, ist das normierte Exponentialformat entwickelt worden. Der Dezimalpunkt nimmt dabei immer eine feste Position ein, und durch den hinzugefügten Exponenten zur Basis 10 wird ausgedrückt, um wieviele Stellen der Dezimalpunkt nach links (negativer Exponent) oder nach rechts (positiver Exponent) gerückt werden müßte, damit er sich wieder an der „richtigen" Stelle befindet. Beispiele:

```
1.23E-Ø1 ->   Ø.123 -> 1.23 * 1Ø.Ø↑-Ø1
1.23E+ØØ ->   1.23  -> 1.23 * 1Ø.Ø↑+ØØ
1.23E+Ø1 ->  12.3   -> 1.23 * 1Ø.Ø↑+Ø1
1.23E+Ø2 -> 123.Ø   -> 1.23 * 1Ø.Ø↑+Ø2
```

Das Exponentialformat setzt sich aus dem Mantissenvorzeichen, der eigentlichen Mantisse, dem Exponentenvorzeichen und dem eigentlichen Exponenten zusammen. Mit Ausnahme der Zahl Null (Sonderfall, s.u.) ist die *erste* Ziffer der normierten Mantisse stets eine *geltende Ziffer* (= Nicht-Null-Ziffer). Dabei kann sich der Dezimalpunkt der normierten Mantisse rechts nach der ersten (m.mmm) oder links vor der ersten (.mmmm) Nicht-Null-Ziffer befinden.

In GFA-BASIC beträgt die Mantisse maximal 11 Stellen, und der maximal 3stellige Exponent liegt im Bereich E-153 bis E+153 (= Potenzwert 2↑512). Das Exponentialformat wird bei der Zahlenausgabe in der Regel nur verwendet, wenn der Exponent die Mantissenstellenanzahl von 11 überschreitet. Die Anzahl der Nachkommastellen kann deshalb größer als 11 sein. Beispiel:

Print Pi/100000 → 0.000031415926536.

2. BCD- und Binärsystem

Für Computer kommen im wesentlichen zwei Positionssysteme in Frage:

1) BCD-System: Es handelt sich hierbei im Prinzip um das dezimale Positionssystem (Potenzbasis 10), wobei die BCD-Zahlen jedoch rechnerintern halbbytemäßig gepackt gespeichert werden (= 2 Mantissenstellen pro Byte). Vorteil: Bei der Konvertierung der normalen Dezimalzahlen, d.h. der beispielsweise am Bildschirm sichtbaren ASCII-Ziffernfolgen, in das interne BCD-Format und umgekehrt entstehen keine Rundungsfehler. Nachteil: Mathematische Berechnungen werden vergleichsweise langsam ausgeführt. Dies gilt besonders für den 68000-Prozessor, weil dessen BCD-Befehle nur Bytes (und keine Wörter) verarbeiten können.

2) Binärsystem: Es handelt sich hierbei um das binäre oder duale Positionssystem (Potenzbasis 2), wobei die Binärzahlen intern bitmäßig gepackt gespeichert werden (8 Mantissenstellen pro Byte). Nachteil: Bei der Konvertierung der normalen Dezimalzahlen in das interne Binärformat und umgekehrt entstehen Rundungsfehler. Vorteil: Mathematische Berechnungen werden vergleichsweise schnell ausgeführt.

3. Binärumwandlung

In GFA-BASIC belegen Fließkommazahlen 6 Bytes oder 48 Bits. Die Mantisse umfaßt 38 und der Exponent 10 Bits. Das Programm **Grundrechenarten** enthält ein allgemeines Unterprogramm zur Umwandlung der GFA-BASIC-internen Fließkommazahlen in vier verschiedene Formen von sichtbaren Binärzahlen:

1. Rohform als 6 Blöcke zu je 8 Bits
2. 38-Bit-Mantisse und 10-Bit-Exponent
3. Normiertes Exponentialformat
4. Normales Nicht-Exponentialformat

Um Binär- in Dezimalzahlen und umgekehrt zu konvertieren, verwende man eine Tabelle der Zweierpotenzen (s. Listing). Wie man dann beispielsweise die Binärzahl 11101.011 in die entsprechende Dezimalzahl 13.375 umwandelt, zeigt das folgende Beispiel.

```
3 2 1 Ø . 1 2 3 Positionen
1 1 Ø 1 . Ø 1 1 Ziffern

  8.ØØØ = 1 * 2↑+Ø3
 +4.ØØØ = 1 * 2↑+Ø2
 +Ø.ØØØ = Ø * 2↑+Ø1
 +1.ØØØ = 1 * 2↑+ØØ
 +Ø.ØØØ = Ø * 2↑-Ø1
 +Ø.25Ø = 1 * 2↑-Ø2
 +Ø.125 = 1 * 2↑-Ø3
-----------------------
 13.375 = 11Ø1.Ø11
```

Wir wollen nun Schritt für Schritt die Umwandlung der Dezimalzahl 13.375 in das binäre Normalformat demonstrieren (vgl. **Abb. 1**).

Schritt 1: 01010110 ...10 00000011

Zunächst verwandeln wir die 6 Bytes der Fließkommazahl 13.375 in 6 Blöcke zu je 8 Bits (s. Abb. 1, 1. Zeile des 1. Dreierblocks). Mit dieser kryptischen Darstellung können wir natürlich noch nichts anfangen.

Schritt 2: 01010110... 1000000011

Deshalb splitten wir die Bitfolge in die 38 Bits der Mantisse sowie die 10 Bits des Exponenten auf (s. Abb. 1, 1. Zeile des 2. Dreierblocks).

Schritt 3: +1.1010110...+000000011

Jetzt wird es komplizierter, denn wir müssen nun die Vorzeichen und den Binärpunkt ermitteln (s. Abb. 1, 1. Zeile des 3. Dreierblocks):

– Die erste Stelle der normierten Mantisse ist immer eine Nicht-Null-Ziffer, wie oben bereits dargelegt wurde. Da Binärzahlen jedoch nur aus den Ziffern 0 und 1 bestehen, beginnt die genormte Mantisse von GFA-BASIC-Zahlen regelmäßig mit einer „1", sofern die Binärzahl nicht den Wert Null hat (s.u.). Und weil dies immer so ist, wird das erste Bit der Mantisse zur Verschlüsselung des Vorzeichens verwendet:
Wenn 1. Bit = „0", dann „+".
Wenn 1. Bit = „1", dann „-".
Wir gelangen damit von „01010110" über „11010110" zu „+11010110".

– Der Binärpunkt steht bei GFA-BASIC-Zahlen immer *nach der ersten* Nicht-Null-Ziffer. Es genügt deshalb, daß wir uns den Binärpunkt denken, anstatt ihn als Bit o.ä. physisch zu markieren. Wir gelangen damit von „+11010110" zu „+1.1010110".

– Nun müssen wir noch den 10stelligen Exponenten aufteilen. Für das Vorzeichen des Exponenten gilt:
Wenn 1. Bit „1", dann „+".
Wenn 1. Bit „0", dann „-".
Aus „1000000011" wird dann „+000000011", also Vorzeichenbit plus 9stelliger Exponent.
(Negative Exponenten werden komplementär dargestellt, z.B. „0111111111" → „-111111111" → „-000000001" → dezimal „-1".)

Schritt 4: 1101.0110 ...

Nun brauchen wir nur noch den Binärpunkt gemäß dem Wert des Exponenten zu verrücken (hier um +3 Stellen nach rechts), und wir sind bei der binären Normaldarstellung angelangt (s. Abb. 1, 1. Zeile des 4. Dreierblocks). Man beachte, daß man die Nullen am Ende der Nachkommastellen nicht ohne weiteres streichen kann, weil ein periodischer Binärbruch (s.u.) vorliegen könnte.

Null als Sonderfall

Bei Binärzahlen muß man die Null stets als Sonderfall behandeln, weil sonst keine Unterscheidung zwischen „+0", „+1" und „-1" möglich wäre. In GFA-BASIC ist eine Binärzahl dann Null, wenn alle 48 Bits auf Null gesetzt sind (vgl. **Abb. 2**).

Andere Zahlenformate

Das obige Umwandlungsverfahren ist GFA-BASIC-spezifisch. Interessierte finden am Ende der Listings einen beispielmäßigen Vergleich des GFA-Formats mit zwei weiteren Formaten: DRI = Digital-Research-BASIC (auf Atari-Systemdiskette mitgeliefert) und AS = Applesoft-BASIC (identisch mit CP/M-MBASIC 5.27, wenn man von der Anzahl und der Reihenfolge der Bytes absieht).

– Bei DRI (3 Mantissenbytes und 1 Exponentenbyte) und AS (4 Mantissenbytes und 1 Exponentenbyte) steht der Binärpunkt *vor* der ersten geltenden Ziffer und nicht wie bei GFA-BASIC danach, so daß der positive/negative Exponent um 1 erhöht/erniedrigt ist.

– Bei DRI befindet sich das Mantissenvorzeichen als erstes Bit im Exponentenbyte, so daß das erste DRI-Mantissenbit im Gegensatz zu GFA-BASIC und AS stets den korrekten Wert 1 hat. Damit wird der DRI-Exponent jedoch um 1 Bit gekürzt: DRI: 6 Bits = 2 ↑ 64, AS: 7 Bits = 2 ↑ 128, GFA-BASIC: 9 Bits = 2 ↑ 512.

4. Rundungsfehler

Wenn wir zwei 2stellige Zahlen multiplizieren, kann sich ein 4stelliges Produkt (99 * 99 = 9801) ergeben. Arbeitet man jedoch mit einer konstanten Mantissenlänge für Faktoren und Produkt, dann fällt die zweite Hälfte der Produktmantisse regelmäßig unter den Tisch, was durch den Exponenten abgefangen wird (99 * 99 = 98E+02). Mit derartigen Rundungsfehlern müssen wir bei der Exponentialdarstellung leben, denn dafür erhalten wir im Gegensatz zu einfachen Taschenrechnern keine Fehlermeldung, wenn die Mantisse überläuft. Es gibt jedoch noch diffizilere Probleme:

4.1. Periodische Brüche

Dezimale Ganzzahlen (z.B. 987.0) lassen sich ohne Rundungsfehler in Binärzahlen konvertieren (s. **Abb. 3**). Das gleiche gilt für Dezimalbrüche, die sich aus Zweierpotenzen zusammensetzen, z.B. 128 + 8 oder 0.5 + 0.25 usw. So läßt sich etwa die Dezimalzahl 0.5 exakt durch die Binärzahl 0.1 darstellen. Vgl. auch **Abb. 4** (echter Bruch) und Abb. 1 (unechter Bruch). In der Regel führt jedoch die Konvertierung einer Dezimalzahl zu einem periodischen Binärbruch (analog zu 1/3 → 0.333...). So muß beispielsweise die Dezimalzahl 0.1 durch den unendlichen Binärbruch 0.0_0011_0011... mit der Vorperiode „0" und der Periode „0011" dargestellt werden:

```
                Ø.5ØØØØØØØØ E-Ø1 zu groß
                Ø.25ØØØØØØ E-Ø2 zu groß
                Ø.125ØØØØØ E-Ø3 zu groß
+Ø.Ø625ØØØØ E-Ø4 geht
+Ø.Ø3125ØØØ E-Ø5 geht
                Ø.Ø15625ØØ E-Ø6 zu groß
                Ø.ØØ7812ØØ E-Ø7 zu groß
+Ø.ØØ3ØØ625Ø E-Ø8 geht
+Ø.ØØ1953125 E-Ø9 geht
-------------------------------
 Ø.Ø99ØØ9375 Zwischensumme usw.
```

Abb. 8 („10.0 + 0.1") führt deshalb nur „zufällig" zum korrekten Ergebnis („10.1"), denn in der 2. Zeile des 4. Dreierblocks sehen wir deutlich, daß die letzten 7 Mantissenbits der Dezimalzahl „0.1" beim Addieren unter den Tisch fallen. Es wird zwar noch intern das 39. Bit erfaßt, das jedoch 0 ist und somit keine Aufrundung bewirkt. **Abb. 9** („10.1 - 10.0") führt deshalb prompt zu einem Rundungsfehler („0.099999999977"), weil es beim ersten Summanden nichts mehr zum Aufrunden gibt und deshalb die letzten 7 Bits der Summe auf 0 gesetzt werden. Erst die Aufrundung durch Print Using (s. Listing) führt wieder zur korrekten Summe („0.1").

Während bei der Binärzahl „0.0_0011_0011..." die Periode mehrfach im Mantissenfeld untergebracht werden kann, ist dies bei den meisten Binärbrüchen nicht der Fall, denn die Perioden sind in der Regel erschreckend lang. So hat beispielsweise die Schnapszahl „0.12345" eine exakt 500stellige Periode! Von daher wird klar, daß man sich normalerweise mit abgehackten Perioden zufriedengeben muß.

Fazit: Mißtraue der vollen Mantisse und runde mittels Print Using auf die vorletzte Stelle auf.

4.2. Exponentendifferenz

Der Fehler mit der berüchtigten „10.1" (sinngemäß „100.1", „1000.1" usw.) ist jedoch harmlos im Vergleich zu Fehlrundungen, die entstehen, wenn die Mantissen im Normalformat zu stark auseinanderklaffen (= zu große Exponentendifferenz), was auch bei nicht-periodischen Binärbrüchen vorkommen kann. Beispiel:

```
16777216.Ø  +  Ø.ØØØ9765625
12345678          9Ø12345678
mehr als 11 Stellen Abstand
```

Abb. 5 liefert als Ergebnis die 11stellige Dezimalzahl „16777216.001", wobei „0.0009765625" gerade noch auf „0.001" gerundet werden kann. Intern ist das binäre Ergebnis hingegen völlig exakt, wie man der 3. Zeile des 4. Dreierblocks unschwer entnehmen kann.

Abb. 6 liefert kein sichtbar korrektes Ergebnis mehr, aber intern ist die Zahl noch korrekt.

Abb. 7 zeigt schließlich, daß auch intern das letzte Bit nicht mehr vorhanden ist.

Fazit: Schreibe Formelausdrücke (z.B. oX*X*X/1/2/3") dergestalt um, daß sich die Exponentendifferenz bei Zwischenergebnissen in Grenzen hält (z.B. „X/1*X/2*X/3").

4.3. Vergleichsfehler

Die Exponentendifferenz enthüllt ein Problem, das uns bei Vergleichsbefehlen zu schaffen macht, denn zwei Variablen X und Y, die nach Print am Bildschirm dieselbe Anzeige produzieren, müssen intern noch lange nicht identisch sein. Als Beispiel berechnen wir die Quadratwurzel nach dem Newton-Verfahren (s. Listing **Newton-Wurzel** sowie **Abb. 10-12**). Als Abbruchkriterium vergleichen wir den momentanen Näherungswert mit dem zuvor berechneten (mit/ohne Berücksichtigung einer Konstanten). Fatal wäre dagegen ein Vergleich in der Form

```
R = Sqr(X)
S = R * R
If S = X then...,
```

denn dieser würde meistens zu einer Endlosschleife führen, obwohl S und X am Bildschirm die gleiche Anzeige produzieren würden.

Fazit: Traue nie dem Spruch: What you see is what you get!

(Die im Text erwähnten Listings werden einschließlich eines Programms zur Berechnung beliebig langer Binärbruch-Perioden im nächsten Peeker veröffentlicht.)

```
13.375 + 3.25 = 16.625        16.625000000

 01010110 00000000 00000000 00000000 00000010 00000011    13.375
 01010000 00000000 00000000 00000000 00000010 00000001    3.25
 00000101 00000000 00000000 00000000 00000010 00000100    16.625

 01010110000000000000000000000000000000000000000000011    13.375
 01010000000000000000000000000000000000000000000000001    3.25
 00000101000000000000000000000000000000000000000000100    16.625

+1.0101011000000000000000000000000000000000000 +000000011 +3    13.375
+1.0100000000000000000000000000000000000000000 +000000001 +1    3.25
+1.0000010100000000000000000000000000000000000 +000000100 +4    16.625

 +1101.0110000000000000000000000000000000000
   +11.0100000000000000000000000000000000000000
+10000.0101000000000000000000000000000000000
```

Abb. 1

```
-1 + 1 = 0        0.000000000

 10000000 00000000 00000000 00000000 00000010 00000000    -1
 00000000 00000000 00000000 00000000 00000010 00000000    1
 00000000 00000000 00000000 00000000 00000000 00000000    0

 10000000000000000000000000000000000000000000000000000    -1
 00000000000000000000000000000000000000000000000000000    1
 00000000000000000000000000000000000000000000000000000    0

-1.0000000000000000000000000000000000000000 +000000000 +0    -1
+1.0000000000000000000000000000000000000000 +000000000 +0    1
+0.0000000000000000000000000000000000000000 +000000000 +0    0

-1.00000000000000000000000000000000000000000
+1.00000000000000000000000000000000000000000
 0.00000000000000000000000000000000000000000
```

Abb. 2

```
992 + 1024 = 2016      2016.000000000

 01111000 00000000 00000000 00000000 00000010 00001001    992
 00000000 00000000 00000000 00000000 00000010 00001010    1024
 01111100 00000000 00000000 00000000 00000010 00001010    2016

 01111000000000000000000000000000000000000000000001001    992
 00000000000000000000000000000000000000000000000001010    1024
 01111100000000000000000000000000000000000000000001010    2016

+1.1111000000000000000000000000000000000000000 +000001001 +9    992
+1.0000000000000000000000000000000000000000000 +000001010 +10    1024
+1.1111100000000000000000000000000000000000000 +000001010 +10    2016

 +1111100000.000000000000000000000000000000
+10000000000.000000000000000000000000000000
+11111100000.000000000000000000000000000000
```

Abb. 3

```
0.9990234375 - 0.875 = 0.1240234375        0.124023438

 01111111 11000000 00000000 00000000 00000001 11111111    0.9990234375
 01100000 00000000 00000000 00000000 00000001 11111111    0.875
 01111110 00000000 00000000 00000000 00000001 11111100    0.1240234375

 01111111111000000000000000000000000000000001111111111    0.9990234375
 01100000000000000000000000000000000000000001111111111    0.875
 01111110000000000000000000000000000000000001111111100    0.1240234375

+1.1111111111000000000000000000000000000000 -000000001 -1    0.9990234375
+1.1100000000000000000000000000000000000000 -000000001 -1    0.875
+1.1111110000000000000000000000000000000000 -000000100 -4    0.1240234375

+0.1111111111000000000000000000000000000000
+0.1110000000000000000000000000000000000000
+0.0001111111100000000000000000000000000000
```

Abb. 4

```
16777216 + 0.0009765625 = 16777216.001    16777216.001000000

 00000000 00000000 00000000 00000000 00000010 00011000    16777216
 00000000 00000000 00000000 00000000 00000001 11110110    0.0009765625
 00000000 00000000 00000000 00000000 00100010 00011000    16777216.001

 00000000000000000000000000000000000000000000000011000    16777216
 00000000000000000000000000000000000000000000000001111    0.0009765625
 00000000000000000000000000000000000000000001000011000    16777216.001

+1.0000000000000000000000000000000000000000000 +000011000 +24    16777216
+1.0000000000000000000000000000000000000000000 -000001010 -10    0.0009765625
+1.0000000000000000000000000000000000000001000 +000011000 +24    16777216.001

+10000000000000000000000.000000000000000
                        +0.000000000100000000000000000000000000000000000000000000000000
+10000000000000000000000.000000000001000
```

Abb. 5

```
10 + 0.1 = 10.1        10.100000000

 00100000 00000000 00000000 00000000 00000010 00000011    10
 01001100 11001100 11001100 11001100 11001101 11111100    0.1
 00100001 00011001 00011001 00011001 00011010 00000011    10.1

 00100000000000000000000000000000000000000000000000011    10
 01001100110011001100110011001100110011001100111111100    0.1
 00100001100110011001100110011001100110011001100111111    10.1

+1.0100000000000000000000000000000000000000000000000 +000000011 +3    10
+1.0011001100110011001100110011001100110011001100111 -000000100 -4    0.1
+1.0100001100110011001100110011001100110011001100111 +000000011 +3    10.1

+1010.000000000000000000000000000000000000000
   +0.000110011001100110011001100110011001100110011
+1010.000110011001100110011001100110011001100110
```

Abb. 6

```
536870912 + 0.0009765625 = 536870912    536870912.000000000

 00000000 00000000 00000000 00000000 00000010 00011101    536870912
 00000000 00000000 00000000 00000000 00000001 11110110    0.0009765625
 00000000 00000000 00000000 00000000 00000010 00011101    536870912

 00000000000000000000000000000000000000000000000011101    536870912
 00000000000000000000000000000000000000000000000001111    0.0009765625
 00000000000000000000000000000000000000000001000011101    536870912

+1.0000000000000000000000000000000000000000000 +000011101 +29    536870912
+1.0000000000000000000000000000000000000000000 -000001010 -10    0.0009765625
+1.0000000000000000000000000000000000000001000 +000011101 +29    536870912

+10000000000000000000000000000.000000000
                        +0.000000000100000000000000000000000000000000000000000000000000
+10000000000000000000000000000.000000000
```

Abb. 7

```
134217728 + 0.0009765625 = 134217728    134217728.000000000

 00000000 00000000 00000000 00000000 00000010 00011011    134217728
 00000000 00000000 00000000 00000000 00000001 11110110    0.0009765625
 00000000 00000000 00000000 00000000 00000010 00011011    134217728

 00000000000000000000000000000000000000000000000011011    134217728
 00000000000000000000000000000000000000000000000001111    0.0009765625
 00000000000000000000000000000000000000000001000011011    134217728

+1.0000000000000000000000000000000000000000000 +000011011 +27    134217728
+1.0000000000000000000000000000000000000000000 -000001010 -10    0.0009765625
+1.0000000000000000000000000000000000000001000 +000011011 +27    134217728

+100000000000000000000000000.000000000
                        +0.000000000100000000000000000000000000000000000000000000000000
+100000000000000000000000000.000000001
```

Abb. 8

```
10.1 - 10 = 0.099999999977        0.100000000

 00100001 10011001 10011001 10011001 10011010 00000011    10.1
 00100000 00000000 00000000 00000000 00000010 00000011    10
 01001100 11001100 11001100 11001100 00001001 11111100    0.099999999977

 00100001100110011001100110011001100110011001100111100    10.1
 00100000000000000000000000000000000000000000000000100    10
 01001100110011001100110011001100110011000000000011100    0.099999999977

+1.0100001100110011001100110011001100110011001100111 +000000011 +3    10.1
+1.0100000000000000000000000000000000000000000000000 +000000011 +3    10
+1.0011001100110011001100110011001100110011000000000 -000000100 -4    0.099999999977

+1010.000110011001100110011001100110011001100111
+1010.000000000000000000000000000000000000000000
   +0.000110011001100110011001100110011001100000000
```

Abb. 9

```
S = SQR(2) = 1.4142135624
Abbruch durch Differenz zwischen
Vorher-Wurzel und Nachher-Wurzel
Zähler:  6
Square: +1.0110101000000100111100110011001111111111 -> 1.4142135624
Newton: +1.0110101000000100111100110011001111111110 -> 1.4142135624
S * S : +10.000000000000000000000000000000000000000000000000001 -> 2
N * N : +1.111111111111111111111111111111111111111 -> 2
stimmt nicht
Abbruch durch Differenz zwischen
Vor/Nachher-Wurzel und Konstante
Zähler:  5
Square: +1.0110101000000100111100110011001111111111 -> 1.4142135624
Newton: +1.0110101000000100111100110011001111111110 -> 1.4142135624
S * S : +10.000000000000000000000000000000000000000000000000001 -> 2
N * N : +1.111111111111111111111111111111111111111 -> 2
stimmt nicht
```

Abb.10

```
S = SQR(3) = 1.7320508076
Abbruch durch Differenz zwischen
Vorher-Wurzel und Nachher-Wurzel
Zähler:  6
Square: +1.1011101101100111101011101000010110001 -> 1.7320508076
Newton: +1.1011101101100111101011101000010110001 -> 1.7320508076
S * S : +11.000000000000000000000000000000000000000000000000001 -> 3
N * N : +11.000000000000000000000000000000000000000000000000001 -> 3
stimmt
Abbruch durch Differenz zwischen
Vor/Nachher-Wurzel und Konstante
Zähler:  6
Square: +1.1011101101100111101011101000010110001 -> 1.7320508076
Newton: +1.1011101101100111101011101000010110001 -> 1.7320508076
S * S : +11.000000000000000000000000000000000000000000000000001 -> 3
N * N : +11.000000000000000000000000000000000000000000000000001 -> 3
stimmt
```

Abb.11

```
S = SQR(4) = 2
Abbruch durch Differenz zwischen
Vorher-Wurzel und Nachher-Wurzel
Zähler:  6
Square: +10.000000000000000000000000000000000000000000000000000 -> 2
Newton: +10.000000000000000000000000000000000000000000000000000 -> 2
S * S : +100.000000000000000000000000000000000000000000000000000 -> 4
N * N : +100.000000000000000000000000000000000000000000000000000 -> 4
stimmt
Abbruch durch Differenz zwischen
Vor/Nachher-Wurzel und Konstante
Zähler:  6
Square: +10.000000000000000000000000000000000000000000000000000 -> 2
Newton: +10.000000000000000000000000000000000000000000000000000 -> 2
S * S : +100.000000000000000000000000000000000000000000000000000 -> 4
N * N : +100.000000000000000000000000000000000000000000000000000 -> 4
stimmt
```

Abb.12

3 starke Bücher für Ihren

Atari ST

Dieter und Jürgen Geiß

Logo auf dem Atari ST

1986, 146 S., DM 35, –,
ISBN 3-7785-1262-5

LOGO ist die erste Sprache auf dem Atari ST. Hier treffen sich die hervorragenden grafischen Fähigkeiten einer Programmiersprache und die überlegenen Leistungen des neuen Rechners. Das Atari-LOGO unter der Benutzeroberfläche des GEM verfügt über den zur Zeit größten LOGO-Sprachumfang und macht vollen Gebrauch von Fenstern, der Maus als Eingabegerät und den sogenannten Drop-Down-Menüs. Dieses Buch zeigt das Planen und Schreiben von faszinierenden und nützlichen Programmen. Das gesamte LOGO-System – nämlich Bedienung und Sprache – wird vorgestellt. Hier stehen die Antworten auf Fragen, die im Original-Handbuch offen geblieben sind. Der Leser lernt die gesamte LOGO-Sprache mit strukturierter Top-Down-Programmierung, Prozeduren, Rekursionen usw. Einige beispielhafte Projekte zeigen, daß LOGO weit mehr ist als eine anschauliche Lernsprache für Kinder.

Hajo Lemcke, Volker Dittmar und Michael Sommer

Programmierlexikon für Atari ST

1986, ca. 500 S., DM 48, –,
ISBN 3-7785-1412-1

Wie jedes Lexikon ist auch dieses vollständig nach Stichworten sortiert. Im Gegensatz zu einem normalen Lexikon findet der Leser hier jedoch nicht nur eine Beschreibung, sondern gleich eine Programmieranleitung. Ebenso sind mehrere Tabellen enthalten, die das Auffinden weiterer Stichworte erleichtern und zusätzliche Informationen beinhalten. Die meisten anderen Bücher enthalten entweder eine Dokumentation über die GEM-Fähigkeiten des Rechners oder die Beschreibung der einfachen Betriebssystemroutinen. Hier jedoch findet der Leser alles. Es gibt nicht nur Hinweise zur Programmierung von Dialogboxen, Fenstern oder Kommandointerpreten, sondern es werden auch alle systeminternen Fragen beantwortet. Dies umfaßt sowohl die Programmierung der im Rechner benutzten Chips, als auch eine Beschreibung der Schnittstellen und deren Benutzung. Es wird auf alle grafischen Möglichkeiten des ST eingegangen. Gleichgültig, ob nach den deutschen oder nach den englischen Begriffen gesucht wird, es sind alle vorhanden und verweisen gegebenenfalls aufeinander.

Software-Entwicklung auf dem Atari ST

1986, 388 S., DM 54, –, ISBN 3-7785-1339-7

In diesem Buch findet der ernsthafte Programmierer alles was er braucht, um gute und professionelle Software auf dem Atari ST zu entwickeln.
Der vollständige Arbeitsablauf sowohl in einer C- als auch in einer PASCAL-Umgebung wird beschrieben, die notwendigen BATCH-Programme zum Compilieren, Assemblieren und Linken sind gelistet.
Das Kapitel 3 beschäftigt sich mit der Entwicklung von reinen TOS-Programmen. An dieser Stelle werden sowohl das Betriebssystem und der Aufruf aller GEMDOS-, BIOS- und XBIOS-Funktionen als auch die Bedeutung der System-Variablen erklärt.
Kapitel 4 ist das Herzstück des Buches: die GEM-Programmierung. Alle Funktionen der beiden großen GEM-Bibliotheken (VDI, AES) werden behandelt.
Zwei komplette Sitzungen mit dem Recourse-Construction-Set werden in Kapitel 5 dargestellt.
Im Kapitel 6 werden dem Leser an zwei vollständigen, sehr sauber programmierten und kommentierten Beispielen mit einigen hundert Zeilen fast alle Probleme vor Augen geführt und gelöst, die bei der Fensterprogrammierung auftreten. Diese Programme können als Muster für eigene Applikationen oder Desk-Accessories benutzt werden.

☐ D. u. J. Geiß, **Logo auf dem Atari ST,**
ISBN 3-7785-1262-5, DM 35, –

☐ Lemcke, Dittmar und Sommer, **Programmierlexikon für den Atari ST,**
ISBN 3-7785-1412-1, DM 48, –

☐ D. u. J. Geiß, **Software-Entwicklung auf dem Atari ST,**
ISBN 3-7785-1339-7, DM 54, –

BESTELLCOUPON

Gewünschte Bücher bitte ankreuzen und an Dr. Alfred Hüthig Verlag, Postfach 102869, 6900 Heidelberg, schicken.

Name _____

Straße _____

Ort _____

Datum _____ Unterschrift _____

Hüthig

Warum wollen Sie ein teures Textverarbeitungsprogramm kaufen, wenn es ein billiges und besseres gibt?

Fast-Writer

von Harald Grumser
Programmdiskette und Handbuch
Gerätevoraussetzung: Apple IIe oder IIc (nicht II+)
DOS-3.3-Version.
Normalpreis DM 128,– (ISBN 3-7785-1419-9)
Sonderpreis für Peeker-Abonnenten DM 98,–

ProDOS-Version.
Normalpreis DM 128,– (ISBN 3-7785-1421-0)
Sonderpreis für Peeker-Abonnenten DM 98,–
Kombinationspreis für Bezieher der DOS-3.3-Version DM 28,–

Fast-Writer läuft auch auf neuem Apple GS!

Der Fast-Writer von Harald Grumser ist in zahlreichen Funktionen wie Scrollen, Suchen und Ersetzen mit Abstand das schnellste und damit angenehmste Textverarbeitungsprogramm für den Apple IIe oder IIc.

Flexibilität

Viele Textverarbeitungsprogramme sind geschützt und laufen deshalb nur in Verbindung mit normalen Disk-II-Laufwerken. Nicht so der Fast-Writer!

– Der Fast-Writer modifiziert weder DOS 3.3 (oder Diversi-DOS) noch ProDOS und kann deshalb mit BRUN FAST.WRITER gestartet werden. Unter Diversi-DOS ist der Fast-Writer dann in 3 Sekunden im Speicher. Vergleichen Sie einmal, wie lange es dauert, bis andere Textprogramme im Speicher sind!

– Da der DOS-3.3-Fast-Writer in den oberen 16K (= Language Card) liegt, kann man ihn vorübergehend verlassen und mit einem einfachen Befehl wieder starten. Mit anderen Worten: Der Fast-Writer ist permanent verfügbar, auch wenn Sie zwischendurch beispielsweise mit FID Dateien kopiert haben.

– Der Fast-Writer läuft mit allen externen Datenspeichern, die für DOS 3.3 oder ProDOS gedacht sind, z.B. mit dem Erphi-160-Spur-Subsystem, mit der Megaboard-MDB-Festplatte, mit RAM-Karten usw. Spezielle Anpassungen sind nicht erforderlich. Suchen Sie einmal ein Textprogramm, das mit diesen Datenspeichern auf Anhieb funktioniert!

– Der Fast-Writer kann mühelos über ein Menü für Ihre speziellen Aufgaben konfiguriert werden. Sie können z.B. per Knopfdruck die Zeilenbreite (normal 80 Zeichen) am Bildschirm einstellen, wobei ab einer Breite von weniger als 41 Zeichen automatisch auf die größere Bildschirmschrift umgestellt wird. Ferner können Sie die Größe des Arbeitsspeichers (insgesamt ca. 35 500 Zeichen) beliebig in Textspeicher und Hilfspuffer (für Löschen und Blockverschieben) aufteilen. Wenn Sie z.B. große Textblöcke im Speicher zu verschieben haben, so können Sie einen entsprechend großen Hilfspuffer von z.B. 10 000 Zeichen einrichten. Damit entfällt das zeitraubende Zwischenspeichern und Einlesen von Diskette.

Befehlsvorrat

Der Fast-Writer verfügt über eine große Zahl von Befehlen, von denen Sie in der Praxis jedoch nur wenige benötigen. Fünf Befehlsübersichten sind durch eingebaute Hilfsübersichten immer abrufbar, so daß Sie schon nach einer mehrstündigen Einarbeitung auf das Handbuch verzichten können. Eine Auswahl der wichtigsten Befehle:

– Freie Cursorbewegung in allen vier Richtungen mit eingebauter Schnell-Scroll-Routine.

– Diverse, per Knopfdruck ein- und ausschaltbare Optionen, z.B. Wortumbruch/kein Wortumbruch, Return sichtbar/unsichtbar, Kopfzeile (Statuszeile) mit Speicherbelegung, Cursorposition usw. eingeblendet/ausgeblendet, Bildschirm geteilt/ungeteilt, Tabulatorleiste sichtbar/unsichtbar, überschreibmodus (statt normalen Einfügmodus) ein/aus usw.

– Eingabe von Kontrollbuchstaben (einschließlich Ctrl-V!) möglich. Automatische Konvertierung in Groß- oder Kleinschreibung (unter Berücksichtigung der Umlaute und ß!)

– Extrem schnelles Suchen und Ersetzen von Zeichenketten (vorwärts und rückwärts).

– Makros frei definierbar und per Knopfdruck abrufbar. Makros können nicht nur stereotype Wortfolgen sein (z.B. „Sehr geehrte Herren"), sondern auch alle Befehlsfolgen, die man beim Fast-Writer sonst über die Tastatur eingeben würde. So läßt sich beispielsweise ein Text automatisch von Laufwerk 1 laden und auf Laufwerk 2 speichern.

– DOS-Kommandos wie Catalog, Delete, Rename usw. immer verfügbar (bei DOS 3.3 zusätzlich Init, bei ProDOS zusätzlich Online und Datum)

– Ausdruck auf Matrixdrucker (normal endlos), Schreibmaschine (normal mit Einzelblatt) und zu Kontrollzwekken auf Bildschirm; links- und rechtsbündig, zentriert und Blocksatz; einstellbarer linker, rechter und oberer Rand (im Text änderbar), bei Bedarf mit Kopfzeile und Paginierung usw. Der Ausdruck kann über eigene Druckertreiber umgelenkt werden, um z.B. Probleme mit Typenrädern, Steuerzeichen usw. zu beheben.

– Makros, Druckparameter, Druckertreiber und Tabulatoren können auf Diskette gespeichert werden.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg 1