Nr. 11/85 November
DM 6.50, sfr 6.50, öS 50, Lit. 5900, hfl 7.50

PEEKER

MAGAZIN FÜR APPLE-COMPUTER

Trickfilmgrafik
Plakatdruck
640K-ProDOS
Print-Using
Pascal-Compiler
Premium-Softcard

16 Seiten Sonderteil Z80-Programmierung

Hüthig PUBLIKATION

Hüthig-FACHBUCH-TIP

Apple Assembler
Tips und Tricks

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

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

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

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

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

BESTELLCOUPON

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

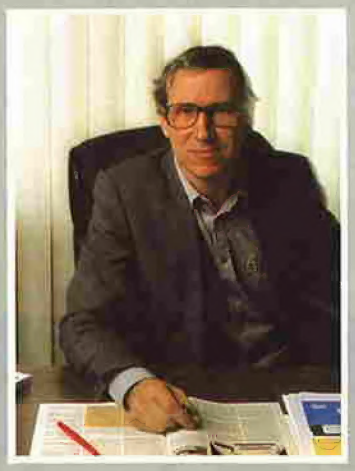
Buchtitel

Name

Straße

Ort

Unterschrift



EDITORIAL

Jobs' Next Apple

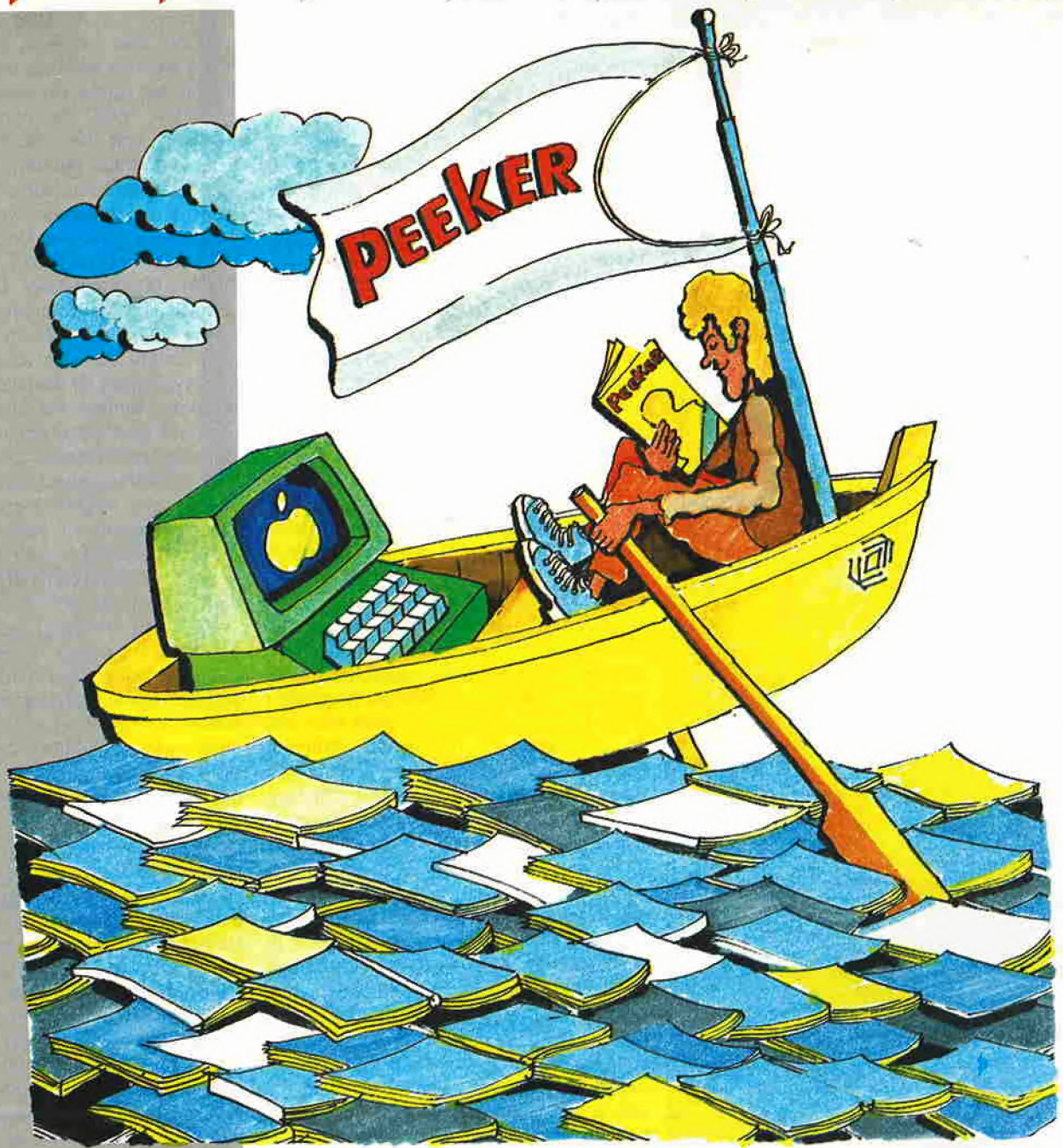
Das Jahr 1985 wird vielen Mikrocomputer-Herstellern in traumatischer Erinnerung bleiben. Bis Ende Januar herrschte eine trügerische Ruhe, denn die Läger des Einzelhandels waren vor Weihnachten mit Mikros vollgepumpt worden. Danach machte sich jedoch der „Pipeline-Effekt" bemerkbar: Der Weihnachtsberg war wider Erwarten nicht abgebaut worden, und Nachbestellungen fanden infolgedessen kaum statt. IBM erkannte als erste Firma die Zeichen der Zeit und trug ihren mit großem Werberummel lancierten PC-Junior bereits vor der europäischen Markteinführung zu Grabe. Danach überschlugen sich förmlich die Hiobsbotschaften in den Wirtschaftsblättern. Allenthalben wurden Gewinneinbußen und vielfach sogar Verluste gemeldet, wobei die branchenweite Flaute auch die Hersteller von Großanlagen in Mitleidenschaft zog (z.B. Wang, Control Data u.a.). Insofern stellen die wirtschaftlichen Entwicklungen bei Apple keinen Einzelfall dar. Bei Apple kommen jedoch neben den „exogenen" Ursachen (Marktflaute, EDV-Müdigkeit) die „endogenen" Einflüsse hinzu: Am 17. September verkündete Steve Jobs der verdutzten Presse, daß er die Firma Apple verlassen und eine eigene Firma namens „Next" gründen werde, die den „nächsten" Apple entwickeln werde. Um dies alles psychologisch erklären zu können, muß man bis zur „Garagenzeit" der beiden Gründer Wozniak und Jobs zurückgehen und mit einem irrigen Klischee aufräumen: Richtig ist, daß Steve Wozniak als das technische Genie gilt, das den Apple II konzipiert und in vielen Hard- und Software-Details selbst entwickelt hat. Falsch ist dagegen das Klischee, Steve Jobs sei das ökonomische Genie gewesen, das den Apple II zur Marktreife geführt habe. Vielmehr hat es Steve Jobs lediglich verstanden, die richtigen (?) Manager anzuheuern, zunächst Mike Markkula von Intel und später John Sculley von Pepsi-Cola.

Steve Jobs, den es gewurmt haben muß, daß er nicht den technischen Durchblick eines Wozniak oder den wirtschaftlichen Weitblick eines Markkula hatte, machte sich dann seit etwa 1983 zum Entwicklungschef der Macintosh-Abteilung, wobei er die denkbar größte Marketing-Todsünde begang, nämlich die völlige Ignorierung des Marktes und der Wünsche der potentiellen Käufer. Als dann Anfang dieses Jahres anläßlich einer Aktionärsversammlung der Apple II mit keinem Wort erwähnt wurde, obwohl der Löwenanteil der Erlöse vom Apple II und nicht vom Macintosh herrührte, kam es zum großen Eklat und kurz darauf zum Ausstieg von Steve Wozniak aus dem Konzern. Es wurde durch Steve Wozniak bekannt, daß viele Neuheiten für den Apple II in der Schublade vermoderten, weil die Apple-II-Weiterentwicklung von der Macintosh-Abteilung nicht zugelassen wurde. Im Laufe der ersten Jahreshälfte verschlechterte sich dann die Ertragslage dramatisch. Beispielsweise sank der Aktienkurs seit der Macintosh-Ära von $62.00 (Mitte 1983) bis auf $16.00 (September 1985). Nun zog John Sculley die Notbremse an. Zunächst wurde die Belegschaft um 20% reduziert, dann wurde Jobs im Juni die Entscheidungsvollmacht für die Macintosh-Division entzogen. Soeben hat Jobs Apple verlassen und eine Nachfolgefirma unter dem sinnträchtigen Namen „Next" in Sunnyvale, Kalifornien gegründet. Im Gegenzug strengt der Apple-Konzern jetzt einen Wettbewerbsprozeß gegen Jobs an, der sich wahrscheinlich über Monate hinziehen wird. Wird es damit neben dem „first" Jobs-Macintosh bald einen kompatiblen „next" Jobs-Macintosh geben? Immerhin hat Jobs mit einem Aktienanteil von 85 Millionen Dollar die Finanzkraft, um dem angeschlagenen „Mutterhaus" Paroli zu bieten.

Dem verunsicherten Apple-II-Besitzer stellt sich die Frage, wie dieses Drama, das die „FAZ" mit der Fernsehserie „Dallas" vergleicht, weitergehen wird. Bezüglich des Apple II kann man zumindest konstatieren, daß nach der Befreiung von der Jobs-Last fieberhaft an Neuentwicklungen gearbeitet wird, von denen noch in diesem Herbst mehrere erscheinen werden, z.B. neue ROMs (s. Peeker 10/85), Speichererweiterung bis 1 Megabyte, 800K-3,5-Zoll-Unidisk mit neuem ProDOS und Pascal, farbfähiger Imagewriter II, Farbmonitor für IIc u.a. All dies läßt uns hoffen. Nachdem jedoch das jugendlich-frische Apple-Image mit den schlacksigen Sonnyboys aus Southern California lädiert ist, werden in Zukunft nur Qualität und Preis überzeugen können.

Ulrich Stiehl

INHALT



Impressum

Peeker
Magazin für Apple-Computer
2. Jahrgang 1985
ISSN 0176–9200
© für den gesamten Inhalt einschließlich der Programme Dr. Alfred Hüthig Verlag, Heidelberg 1985

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

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

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

Hüthig PUBLIKATION

PEEKER

Heft 11/1985

Hinweis:

Der Pascal-Kurs ist auf 16 Seiten angewachsen und wird daher erst im Heft 12/85 veröffentlicht werden, weil das jetzige Heft bereits einen Z80-Sonderteil enthält.

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

Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement DM 72,–, einschließlich MwSt,
im Inland portofrei. Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089/319 10 67, Telex 0 522 656

Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postscheckkonten: BRD: Karlsruhe 485 45-753; Österreich: Wien 7555 8 88; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 47718; Belgien: Brüssel 723 0 26; Dänemark: Kopenhagen 349 69; Norwegen: Oslo 994 24; Schweden: Stockholm 547 7 76-5

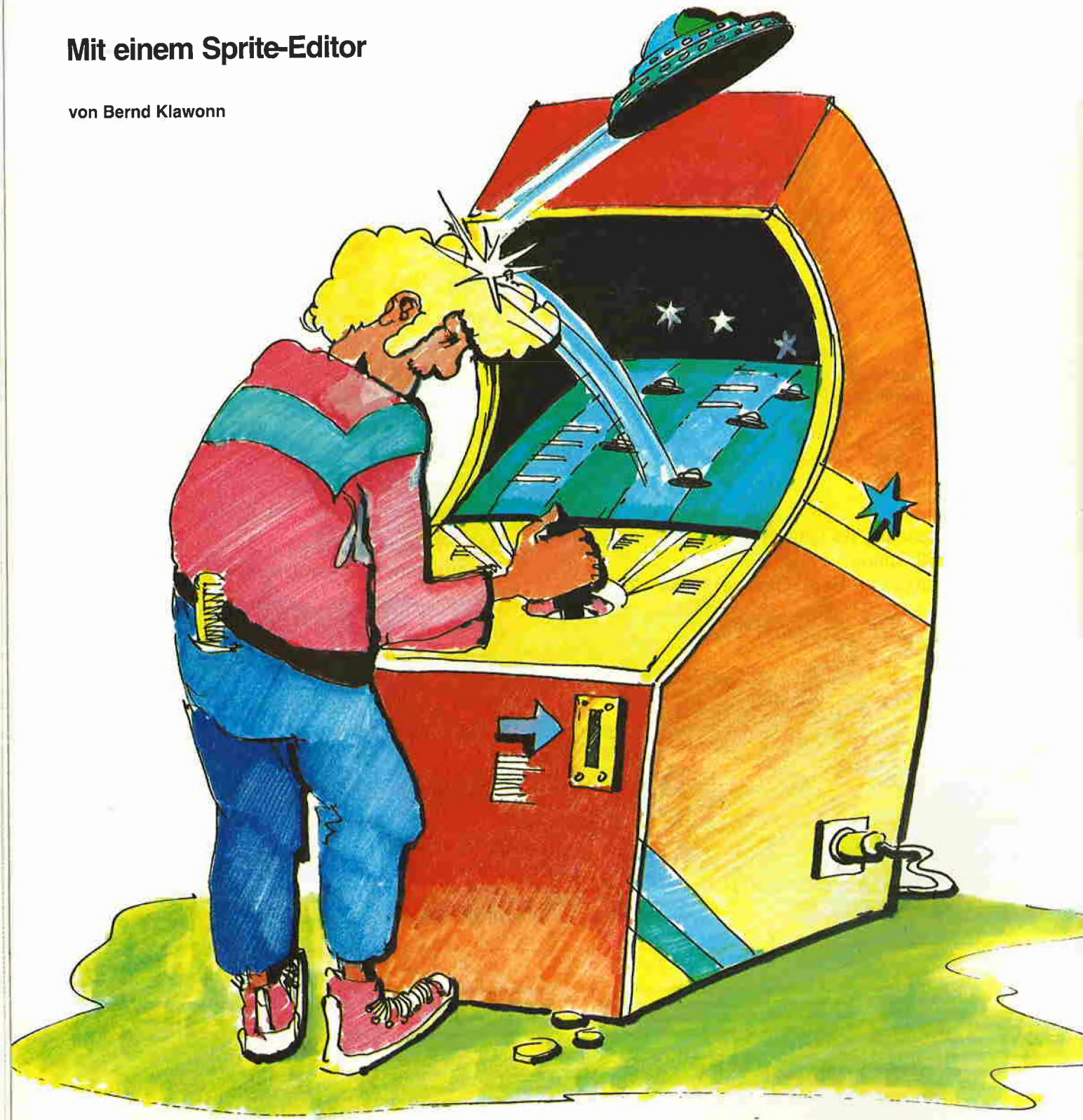
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 0 2165 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Trickfilmgrafik für Spielprogramme

Mit einem Sprite-Editor

von Bernd Klawonn



Das hier vorgestellte Programm wendet sich an den fortgeschrittenen Assemblerprogrammierer, der eigene Spiele oder Animationen mit professioneller Grafik erstellen möchte. Mit einem komfortablen Sprite-Editor können Objekte erstellt und in eigene Assemblerprogramme eingebunden werden. Die Darstellung der Sprites übernimmt das Modul ASTA.

Ziel war die Entwicklung einer Routine, die möglichst viele Figuren gleichzeitig flimmerfrei über den Bildschirm bewegt. Dabei sollte die Ansteuerung durch andere Programme so einfach wie die Verwendung der Shapes in Applesoft sein.
Shapes sind für diese Aufgabe viel zu langsam. Deshalb müssen die Figuren auf andere Weise abgespeichert werden. Als Lösung bietet sich die Verwendung von **Sprites** an, wie sie vom Commodore 64 her bekannt sind. Eine Figur wird bewegt, indem die alte Figur gelöscht und etwas versetzt neu gezeichnet wird. Dabei kommt es zu einem Flimmern auf dem Bildschirm, weil zeitweise die Figur nicht oder nur teilweise auf dem Bild erscheint. Deshalb ist es besser, ein Bild erst zu zeigen, wenn es fertig gezeichnet ist. Diese Trickfilmtechnik ist mit dem Apple leicht zu realisieren, da zwei unabhängige Grafikseiten (Pages) vorhanden sind. Während man eine Seite zeigt, wird in der anderen das nächste Bild fertig gezeichnet. Dann kann man auf die andere Seite umschalten und umgekehrt verfahren (sog. **Page-Flipping**).
Die hier vorgestellte Routine arbeitet mit drei Seiten. Page 1 (im Bereich $2000-$3FFF) und Page 2 ($4000-$5FFF) werden abwechselnd zur Darstellung benutzt, während in Page 3 ($6000-$7FFF) nur der Hintergrund gespeichert ist. Dies ist erforderlich, da die Objekte den Bildschirm überschreiben, und zu deren Löschung der Hintergrund wieder hergestellt werden muß.
Die Routine erlaubt bis zu 30 gleichzeitig bewegte Objekte, wobei jedoch 127 verschiedene Sprites verwaltet werden, und liefert dabei – je nach Anzahl und Größe der Objekte – 5-25 Bilder pro Sekunde. Auf einfache Weise können auch alle vorhandenen Farben benutzt werden. Der Name der Routine ist **ASTA** (Animation von Sprites mit Tabellenhilfe).

1. ASTA

1.1. Die Sprites

Die Abspeicherung der Figuren ist eine Abwandlung des in (1) beschriebenen Konzepts. Im Grafikspeicher sind jeweils sieben nebeneinanderliegende Punkte zu einem Byte zusammengefaßt. Wobei eine 1 in den Bits 1-7 (die Bits sind im folgenden von 1 bis 8 durchnumeriert) einem gesetzten Punkt entspricht. Das achte Bit beeinflußt die Farbe der Punkte. Bei einem Sprite werden alle Punkte der Figur direkt zu Bytes zusammengefaßt. Zum Zeichnen des Sprites braucht man nur noch die Bytes in den Speicher zu poken.

Wenn man ein solches Sprite horizontal verschieben möchte, tritt jedoch ein Problem auf: Beginnt man ein Byte später, hat man die Figur gleich um sieben Punkte verschoben. Wie kann man aber die Figur um einzelne Punkte verschieben? Dazu könnte man mit den Shift-Befehlen (ASL/LSR) alle Bytes in eine Richtung verschieben, jedoch bekäme man dann Schwierigkeiten mit dem 8. Bit und brauchte auch mehr Zeit für das Verschieben als für das Zeichnen des Sprites. Deshalb ist es einfacher und vor allem schneller, für jede der sieben Möglichkeiten ein entsprechend geshiftetes Sprite (sog. **Bitsprite**) abzuspeichern.
Damit die ASTA-Routine die Sprites richtig zeichnen kann, müssen ihnen noch einige Angaben hinzugefügt werden. Vor jedem Bitsprite werden mit zwei Bytes die Anzahl der Bytes je Zeile und die Anzahl der Zeilen angegeben. Vor den sieben Bitsprites muß noch eine **Bitsprite-Tabelle** mit den einzelnen Startadressen stehen. Schließlich braucht man noch eine **Sprite-Tabelle** (STAB) mit den Adressen aller Sprites; die ASTA-Routine benötigt dann nur noch eine Nummer und sucht sich mittels der beiden Tabellen und der angegebenen Bildschirmposition das richtige Bitsprite heraus.
Anhand des Beispiels in **Bild 1** können Sie das System noch einmal nachvollziehen. Um die Erstellung der Sprites zu vereinfachen, ist es naheliegend, die mühselige Arbeit mit einem Programm zu erledigen. Mit dem weiter unten beschriebenen Sprite-Editor können Sie ganz einfach die Sprites entwerfen und berechnen lassen.

1.2. Die ASTA-Positionstabelle

Ab Adresse $8000 steht die **Positionstabelle**, in der alle Angaben über die zu zeichnenden Objekte stehen. Zunächst fällt auf, daß sie in drei große Bereiche aufgeteilt ist. Die einzelnen Variablen haben dabei jeweils die gleiche Endung:
- ...TD – nächstes Bild (zu zeichnende Objekte),
- ...TZ – angezeigtes Bild (Zwischenspeicher),
- ...TL – letztes Bild (zu löschende Objekte).

Von dem aufrufenden Programm braucht nur der erstgenannte Bereich verändert zu werden. Die beiden anderen Bereiche werden von der ASTA-Routine erzeugt und dienen der Verwaltung. Jeder Variablen ist ein Feld von 30 Bytes zugeordnet, wobei das erste Byte für das erste Objekt, das zweite Byte für das zweite Objekt vorgesehen ist usw. Dadurch ist es in Assembler sehr einfach, einen Wert zu lesen oder zu schreiben. Man braucht nur die Nummer des Objekts als Index zu laden (im X- oder Y-Register) und kann die Werte dann mit der indizierten Adressierung (z.B. LDA SPRTD,Y) abrufen.

1.2.1. Die Bedeutung der einzelnen Variablen

SPRTD ($8000) – Sprite-Nummer (im Bereich 0-127): Es können bis zu 127 verschiedene Sprites verwendet werden. Sprite-Nummer 0 bedeutet, daß das entsprechende Objekt nicht gezeichnet werden soll.
XWRTD ($801E) – X-Koordinate (im Bereich 0-139): ASTA ist auf die Ausnutzung der Farbmöglichkeiten des Apples ausgelegt. Im normalen Raster (X im Bereich 0-279) hängt die Farbe von der X-Position ab. Alle Punkte mit geraden X-Koordinaten sind grün, alle Punkte mit ungeraden X-Koordinaten violett; bei gesetztem Bit 8 entsprechend orange und blau. Sind zwei benachbarte Punkte gesetzt, ergibt sich weiß. Daraus folgt, daß ein farbiges Sprite in X-Richtung nur in Zweierschritten bewegt werden darf, da sonst die Farben wechseln. Durch die Halbierung der Einteilung ist dieses Problem beseitigt. Wird XWRTD um eins erhöht, bewegt sich das Sprite um zwei Punkte nach rechts und behält die richtige Farbe.
XBYTD ($803C) – interne ASTA-Variable (X-Byte, 0-28): Diese Variable muß vom aufrufenden Programm nicht gesetzt werden.
XBITD ($805A) – interne ASTA-Variable (Bitsprite-Nummer): ASTA berechnet mit XWRTD das erste Byte des Sprites in der Zeile und die Nummer des Bitsprites (muß ebenfalls nicht gesetzt werden).
YWRTD ($8078) – Y-Koordinate (Bereich 0-191): Die Y-Koordinate entspricht der normalen Koordinate, d.h. 0 = oben, 191 = unten.
YMATD ($8096) – untere Bildgrenze (im Bereich 0-191): Die Y-Koordinate eines Objektes kann nach unten begrenzt werden (z.B. weil es hinter einer Mauer hervorkommt). Unterhalb von YMATD wird das Sprite nicht mehr weiter gezeichnet.
DRWTD ($80B4) – Zeichenmodus, Bit-8-Flag (0 = normal, 1 = überzeichen): Im Normalmodus wird beim Zeichnen der Hintergrund übernommen und das alte Sprite gelöscht, bevor das neue Sprite

gezeichnen wird. Bei besonders großen Sprites ist jedoch ein anderer Zeichenmodus zeitlich vorteilhafter. Wenn Bit 8 von DRWTD gesetzt ist, wird nur das neue Sprite gezeichnet, ohne den Hintergrund zu übernehmen und ohne das alte Sprite zu löschen.
MSKTD ($80D2) – Farbmaske ($80 = invertiert, $00 = normal): Ist Bit 8 von MSKTD gesetzt, werden die Farben des Sprites invertiert d.h. aus grün wird orange, aus violett wird blau und umgekehrt.

Die X,Y-Koordinaten geben die obere, linke Ecke des zu zeichnenden Objekts an.

Normalerweise brauchen Sie alle Variablen nur einmal einzusetzen. Danach reicht meist die Veränderung einzelner Werte. ASTA läßt alle Daten in der TD-Tabelle unverändert stehen (außer XBITD/XBYTD). Um die POSTABelle ganz zu löschen, können Sie die **POSINIT**-Routine aufrufen.

1.3. Der Aufruf von ASTA

Wie bringen Sie ASTA in Ihrem eigenen Programm unter? Sehen Sie sich dazu das nachfolgende allgemeine Beispiel für ein Spiel an (siehe hierzu auch **ASTA-.DEMO**):

```
        JSR POSINIT
        JSR INIT

LOOP    LDA $C054
        LDA #$40
        STA HPAG
        JSR PROGRAMM
        JSR ASTA
        JSR GRAFIK

        LDA $C055
        LDA #$20
        STA HPAG
        JSR PROGRAMM
        JSR ASTA
        JSR GRAFIK

        CLC
        BCC LOOP
```

Zunächst müssen alle Variablen gelöscht werden. Zu diesem Zweck wird POSINIT ($85BB) aufgerufen, um die ASTA-Positionstabelle zu löschen. Danach folgt die Initialisierungsroutine des Programms. Dort wird z.B. der Hintergrund eingeladen, der in alle Seiten kopiert wird, der Punktezähler wird auf Null gesetzt, die Anfangskoordinaten der einzelen Objekte werden in die Positionstabelle geschrieben und die Grafik wird eingeschaltet. In der Hauptschleife wird nacheinander das Hauptprogramm (PROGRAMM), ASTA ($8400) und GRAFIK aufgerufen: Das Hauptprogramm berechnet die neuen Positionen der Objekte und ermittelt z.B. Punktgewinne. ASTA zeichnet die Objekte, und mit GRAFIK kann der Punktestand angezeigt oder das Bild zusätzlich verändert werden. In der Schleife erfolgt auch die Umschaltung der zu zeigenden und neu zu zeichnenden Seiten. Dabei gibt HPAG ($00E6) die zu zeichnende Seite an ($20 = Page 1, $40 = Page 2). Diese Umschaltung ist bewußt nicht in ASTA hineingenommen worden, um noch eine Veränderung des Bildes durch weitere Routinen nach dem Aufruf von ASTA zu ermöglichen.

2. Der Sprite-Editor

2.1. Programmaufbau

Das Programm besteht aus zwei Teilen: das Hauptprogramm **SPRITE.EDIT** in Applesoft und einer kleinen Assemblerroutine **SPRITE.EDIT.ASS** mit einigen Shapes. Für die Abspeicherung des Sprites wird nicht ein Array, sondern ein abgegrenzter Speicherbereich benutzt, in dem mit PEEK und POKE die Punkte abgespeichert werden. Dadurch wird weniger Speicherplatz benötigt als durch einen entsprechend großen Array. Außerdem ist es auch einfacher möglich, einige Berechnungen durch ein Assemblerprogramm erledigen zu lassen. Beim Abbruch des Programms bleibt das eingegebene Sprite auch erhalten, wenn das Programm mit RUN neu gestartet wird.
Um die Ausführungszeiten herabzusetzen, kann das Programm einfach compiliert werden. Für den TASC-Compiler muß nur der HIMEM-Befehl gelöscht und bei der Speicherverteilung die Startadresse für Variabeln auf 29000 gesetzt werden. Bei anderen Compilern ist darauf zu achten, daß der Bereich von 16384-29000 für die Grafik und das Datenfeld frei bleibt.

2.2. Die Erstellung eines Sprites

Wählen Sie mit „1" den Editor an. Es steht Ihnen ein 42 * 42 großes Feld zur Verfügung, innerhalb dessen Sie Ihr Sprite gestalten können. Den Cursor können Sie wie im Textmodus mit I, J, K, M bewegen. Wenn Sie gleichzeitig die Ctrl-Taste drükken, wird die Position, an der sich der Cursor befindet, invertiert. Mit „F" können Sie das Farbbit in einer Zeile invertieren. Durch Eingabe von „L" wird das Sprite gelöscht, und mit „V" kann es um ein Bit in jede Richtung verschoben werden. Als Hilfe können Sie durch Eingabe von „?" eine Übersicht aller Editor-Kommandos abrufen.
Neben dem großen Feld wird das Sprite in Orginalgröße gleich viermal dargestellt, damit alle möglichen Farbkombinationen gleichzeitig zu sehen sind. Links oben sehen Sie das Sprite so, wie es normalerweise mit ASTA dargestellt wird. Wenn das Farbbit invertiert wird (MSKTD = $80), erscheint es wie links unten dargestellt. Die beiden rechts abgebildeten Sprites können Sie durch Verschieben um eine Stelle in X-Richtung erzeugen.
Mit „ESC" verläßt man den Editor.
Das Programm bietet auch die Möglichkeit, ein so erstelltes Sprite abzuspeichern oder ein anderes Sprite von Diskette zu laden.

2.3. Die Berechnung der Bitsprites

Hauptaufgabe des Programms ist die Berechnung der Bitsprites. Es erzeugt jedoch nicht direkt die Bytes der Sprites, sondern liefert einen Assembler-Textfile, der erst von einem Assembler übersetzt werden muß. Diese auf den ersten Blick etwas umständliche Methode hat einige Vorteile. In den Adreßtabellen werden absolute Startadressen verlangt, deshalb ist das Verschieben des Sprites in den gewünschten Speicherbereich mit einen Assembler leichter zu ereichen. Außerdem kann der Speicherplatz besser ausgefüllt werden, wenn man die Sprite-Tabellen zwischen Programmteilen unterbringen will. Durch kleine Änderungen kann das Programm an das Format der verschiedenen Assembler leicht angepaßt werden.
Bei der Berechnung der Sprite-Tabelle können Sie zwischen drei Möglichkeiten wählen:

Normal – Es wird ein Sprite erzeugt, das aus 7 Bitsprites besteht.
Normal (alle Farben) – Hier werden zwei Sprites erzeugt, wobei das zweite Sprite gegenüber dem ersten um einen Punkt verschoben ist. Jedoch werden dazu nur 8 Bitsprites benötigt und nicht 2 * 7 Bitsprites wie bei der Benutzung von zwei normalen Sprites. Beim zweiten Sprite ist dem Label ein „S" hinzugefügt.
Einzel-Sprite – Wenn im voraus feststeht, daß ein Objekt nur in Y-Richtung bewegt wird, brauchen Sie nur *ein* Bitsprite. Welches Bitsprite Sie benötigen, können Sie mit folgender Formel berechnen (X im Bereich 0-139!):
Bitsprite-Nummer = (X * 2 / 7 - INT (X * 2 / 7)) * 7.
Nachdem Sie die Art der Berechnung gewählt haben, folgen Fragen nach dem Filenamen und den im Assemblerfile zu verwendenden Labels. Die Größe des Sprites wird automatisch festgelegt. Beim Einzel-Sprite können Sie sie selbst wählen.

Achtung! Zusatzprogramm für *AppleWorks*

DIESE ANZEIGE

wurde vollständig mit *AppleWorks* gedruckt. Dazu benötigen Sie nur noch den *DMP Charger*, der außerdem noch 20 weitere Zeichensätze für Sie bereit hält.

CAPITALS, *Italic*, Schmasch, Schreib,
АВДЕФГНЬКЛМП, ОПЕРПФХУZ

ATHENS LONDON ZEIT

Auch vom *WordStar* und anderen Programmen können diese Zeichen benutzt werden. Sie können die Zeichen verändern, oder auch neue gestalten. Das Programm arbeitet mit dem //e (128K) oder //c und mit dem Imagewriter, auch FX 80, Panasonic, Okidata.
Preis: 198,- incl. MwSt. und ausführlichem Handbuch. *Infoblatt kostenlos.* Versand gegen offene Rechnung möglich

Norbert Hunstig
Nottulner Landweg 81
D-4400 Münster
Tel.: 02534/ 7036 Telex: 892 496

auch bei Pandasoft und Intus erhältlich.

ccp datentechnik

640 KByte-Drives für den Apple //c!!

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

Festplatten für Apple II (//e)

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

640 KByte-Drives für Apple II (//e)

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

80 Zeichen + 64 K für Apple //e

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

HERDERSTR. 12 2000 HAMBURG 76
☎ 040/ 225676

ZUSATZ-KARTEN:

V-24-Schnittstelle	199,-	**Z-80-Karte**	98,-
80-Zeichen-Karte m.Softswitch	236,-	**16 K-Language-Karte**	98,-
Centronics-Karte von Epson für Graphik	210,-	für Text	145,-
Centronics-Schnittstelle für 2 Drucker gleichzeitig			129,-
Eprommer incl. Software			198,-

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

Floppy-Controller

FDC 4 für alle Laufwerke	169,-	Bausatz wie links	159,-
Leerplatine wie oben incl. Prom u. Eprom			98,-
Druck Spooler mit 16, 32 oder 64 KB			**Preis auf Anfrage**

Erphi-Controller **298,-**

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

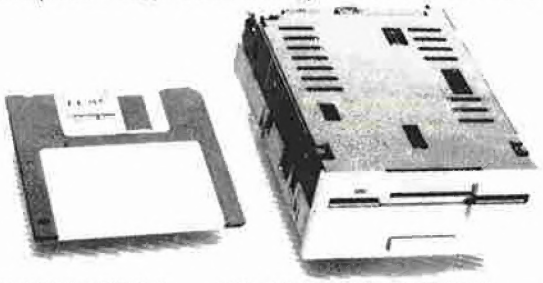
Joy Stick De Luxe	59,-	**Netzteil 5A**	149,-
Gehäuse für 1 5¼" Slimline Laufwerk			39,-
Gehäuse für 2 5¼" Slimline Laufwerke mit Platz für ein Netzteil			159,-
Gehäuse für 2 3½" Slimline Laufwerke mit Platz für ein Netzteil			79,-
IBM®-Gehäuse			229,-
Floppy-Kabel 34pol. für 2 Laufwerke mit Shugart-Bus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

Preh Commander Keyboards

Wir bieten Ihnen die **Preh-Qualität** auch für Apple. AK 88 Spez. mit Gehäuse, Anschlußkabel, Zehner-Tastenfeld, dt. Zeichensatz, Sondertasten für Ctrl-Codes und Rechenfunktionen **339,-**
Preh Commander Keyboard, frei programmierbar bis zu 10 Ebenen, pro Taste bis zu 250 Zeichen . . . . . nur **599,-**

TEAC 3½" Laufwerk FD 35 F 515,-

Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur . . . . .



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

SONY 3½" Laufwerk . . . . . nur **799,-**
Apple®-kompatibles Laufwerk incl. Gehäuse + Kabel **599,-**
320 KB Laufwerk für IIc **948,-**
640 KB Laufwerk für IIc **1088,-**

EPSON DRUCKER

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

Die Microfloppy mit Zukunft:

Speicherkapazität: 2 x 1 MByte formatiert: 2 x 640 kByte. Anschlußfertig mit PROM-residenter Patchsoftware für CP/M 2.2, Apple DOS 3.3, DiversiDOS 2-C, 4-C (DD MOVER), Apple Pascal 1.1, Pascal 1.2, Pro-DOS 1.0.1, 1.1, 1.1.1 zum Preis von **1598,-**
Low Power Version 1740,-

10 MB Winchester

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

Sonderangebot

Distar Laufwerk für II + IIe, incl. Kabel u. Gehäuse . . . . . jetzt nur **398,-**

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

UEDING electronics

Holtewiese 2
5750 Menden 1

DFÜ 02373/66877
Tel. 02373/63159

2.4 Beispiel

Wenn alle Sprites berechnet sind, wird nach folgendem Schema der Sourcecode zusammengefügt (Merlin-Assembler):

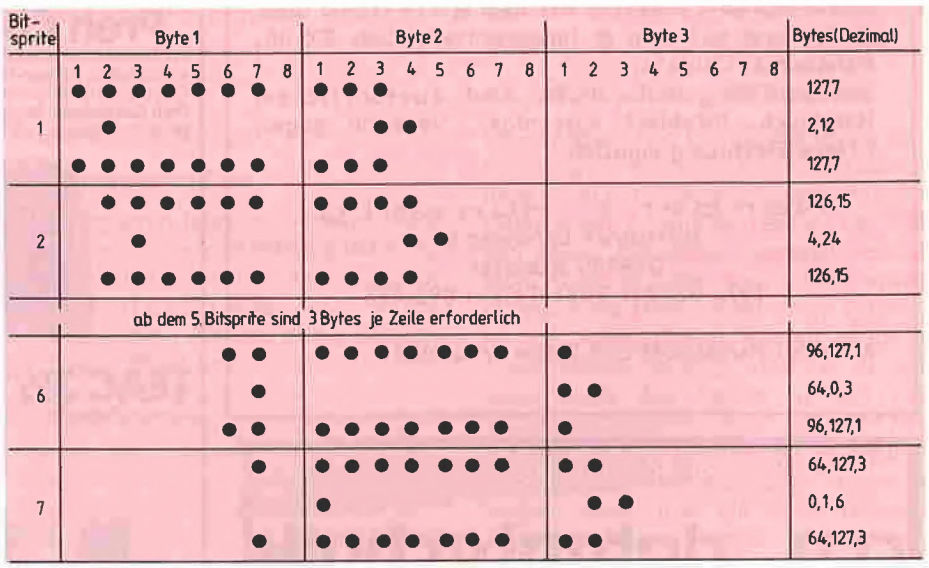
```
ORG $8300
STAB DA $0000
DA SPRITE1
DA SPRITE2
DA SPRITE3

SAV STAB

ORG $9000 ;(Beispiel)
PUT SPRITE1
PUT SPRITE2
PUT SPRITE3

SAV SPRITES
```

Zunächst werden in der Sprite-Tabelle die Startadressen der einzelnen Sprites angegeben, dann folgen erst die eigentlichen Sprites, die in jeden freien Bereich gelegt werden können. Wenn der PUT-Befehl nicht in Ihrem Assembler vorhanden ist, müssen Sie alle Files nacheinander in den Speicher laden und dann assemblieren.

Quellenverzeichnis

(1) Sprite-Designer für Apple II, Homecomputer, 12/83, S.34.
(2) Apple-II-Grafik-Organisation, c't, 6/84, S.40.
(3) Grafik-Tuning, Teil 3, c't, 8/84, S.77.


```
Aufbau des Sprites TEST

TEST        DA   BITSPRITE1
             .
             .
             .
            DA   BITSPRITE7
BITSPRITE1  DFB  2,3          ;Länge (X), Breite (Y)
            DFB  127,7
            DFB  2,12
            DFB  127,7
             *
             *
             *
BITSPRITE7  DFB  3,3
            DFB  64,127,3
            DFB  Ø,1,6
            DFB  64,127,3
```



Beispiel eines Sprites mit entsprechenden Bitsprites.

ASTA

```
BSAVE ASTA, A$84ØØ, L$Ø47A
```

```
1    **********************************
2    *                                *
3    * ASTA - Animation von Sprites *
4    *        mit Tabellenhilfe       *
5    *                                *
6    *       Bernd Klawonn            *
7    *                                *
8    *                                *
9    **********************************
1Ø
11              ORG  $84ØØ
12   ANZAHL     =    3Ø          ;Tab. Sprite-Anzahl
13   POS        EQU  $19         ;Pos. in der Tab.
14   XBYTE      EQU  $1A
15   YWERT      EQU  $1C
16   STLO       EQU  $1D
17   STHI       EQU  STLO+1
18   XLAE       EQU  $1F         ;Länge des Sprites
19   XCOUNT     EQU  $Ø8         ;X-Zähler
2Ø   YCOUNT     EQU  $Ø9         ;Y-Zähler
21   HPAG       EQU  $E6         ;Grafik-Page (Hi)
22   SCRLO      EQU  $Ø6         ;Grafik-Page-Adr.
23   SCRHI      EQU  SCRLO+1
24   PSEUDO     EQU  $FFFF
25   BUFFLO     EQU  $EØ         ;Adresse d. Puffers
26   BUFFHI     EQU  BUFFLO+1    ;(ab $6ØØØ)
27   FMASK      EQU  $E2         ;Farbmaske (Bit 8)
28   YMAX       EQU  $E3
29   *
```

```
3Ø   **********************************
31   *                                *
32   *           A S T A              *
33   * HPAG - Hi-Byte, HIRES-Page     *
34   * PTAB - Positionstabelle        *
35   *                                *
36   **********************************
37   *
38   *     Sprites löschen
39   *
8ØØØ: A9 1E     4Ø   ASTA     LDA  #ANZAHL
8ØØ2: 85 19     41           STA  POS
8ØØ4: C6 19     42   ASTA1    DEC  POS         ;Pos.-Zähler - 1
8ØØ6: A4 19     43           LDY  POS         ;POS als Ind. laden
8ØØ8: B9 EØ 81  44           LDA  SPRTL,Y     ;Sprite = Ø?
8ØØB: FØ Ø3     45           BEQ  ASTA2       ;wenn ja, weiter
8ØØD: 2Ø Bl 84  46           JSR  SLOESCH     ;Sprite löschen
8Ø1Ø: A5 19     47   ASTA2    LDA  POS         ;POS <> Ø?
8Ø12: DØ FØ     48           BNE  ASTA1       ;wenn ja, weiter
                49   *
                5Ø   *     Sprites zeichnen
                51   *
8Ø14: AØ ØØ     52           LDY  #$ØØ
8Ø16: 84 19     53           STY  POS         ;POS auf Ø setzen
8Ø18: B9 FØ 8Ø  54   ASTA3    LDA  SPRTZ,Y     ;POS-Tabelle
8Ø1B: 99 EØ 81  55           STA  SPRTL,Y     ;aktualisieren
8Ø1E: B9 ØØ 8Ø  56           LDA  SPRTD,Y
8Ø21: 99 FØ 8Ø  57           STA  SPRTZ,Y
8Ø24: B9 ØE 81  58           LDA  XWRTZ,Y
8Ø27: 99 FE 81  59           STA  XWRTL,Y
8Ø2A: B9 1E 8Ø  6Ø           LDA  XWRTD,Y
8Ø2D: 99 ØE 81  61           STA  XWRTZ,Y
8Ø3Ø: B9 2C 81  62           LDA  XBYTZ,Y
8Ø33: 99 1C 82  63           STA  XBYTL,Y
8Ø36: B9 3C 8Ø  64           LDA  XBYTD,Y
```

UNIVERSAL KEYBOARDS

Modell AN95FE... DM 448,– ohne MwSt. (DM 510,72 incl. MwSt.)
Die KEYBOARDS SPEZIELL angepaßt für den APPLE IIe
Händleranfragen erwünscht

- FLEXIBEL — Jede Taste frei im EPROM programmierbar in bis zu 8 Ebenen im mitgelieferten EPROM
- PROFESSIONELL — Für Anwender mit gehobenen Ansprüchen
- ERGONOMISCH — Nach DIN ULTRAFLACH gestaltetes stabiles Gehäuse
- KOMPLETT — Tastatur, Gehäuse und Kabel fertig montiert und getestet. Durch Spezial-Kabel und Spezial-EPROM sofort einsteckfertig.
- KOMPAKT + FLACH — Durch Einsatz von „SIEMENS" Flachtasten-modulen

gesellschaft für computersteuerungen und datentechnik mbh
D-4930 Detmold ★ Alter Mühlenweg 5
Telefon 0 52 31/41 76 ★ Telex 9 35 660 acs d

MICROMINT

MICROMINT VOLLTREFFER

MICROMINT 16
XT-IBM comp.

256 K, 8 Slots, Tastatur, Laufwerk 320 K, Contr, Graphic-Color-Card, 15 A Netzteil nur

2.111;

Prospekte anfordern!
Händlerpreise erfragen.

Unsere aktuellen Preise sind meist niedriger. Über Anrufbeantworter Ruf 0 21 04/3 94 71 erfahren Sie's vom Tiefstpreisgarant...

IBM-PREISHAMMER bei 1a QUALITÄT
incl. 14 Tage Rückgaberecht

Fertigplatine XT-256 K/Bootrom	**529,–**
dito Megabord 640	**688,–**
Grafic-Color-Card	**199,–**
Controller 2 LW	**158,–**
384 K Multifunktionscard (256 K on Board)	**459,–**
Multifunctionscard I/O	**300,–**
Winchester-Controller 0–140 MB 1 AX	**799,–**
Winchster-Harddisc 20 MB formatiert	**1.888,–**
Tastatur 1a, kapazitiv, programmierbar	**229,–**
Mehrzweckklappgehäuse IBM-Design	**132,–**
Netzteil 20 A	**229,–**
Monitorfüße schwenkbar 1 a	**29.–**
APPLE 64 K/2 x CPU kompl. lt. Abb.	**988,–**

APPLE-SUPER-SONDER-PREISE – erfragen
TROPHY OF QUALITY = MICROMINT BOOTROM

Generalimporteur MICROMINT Computer GmbH
Hochdahler Straße 151, 4006 Erkrath 2

☎ **02104/33024**

MEGA-BOARD schafft die Verbindung zu Megabytes unter vier Betriebssystemen

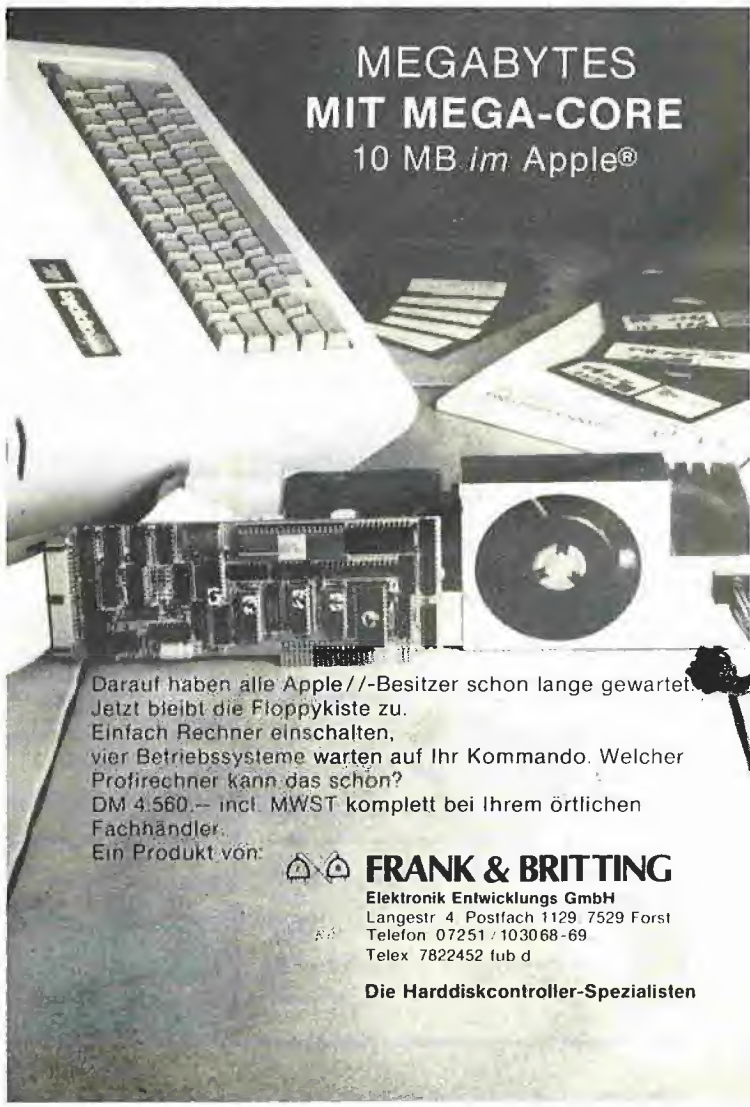
Leistungsumfang:
- Alle Diskparameter einstellbar.
- Betriebssystembereiche frei wählbar.
- Booten von der Harddisk.
- Betriebssysteme DOS, CP/M®, PASCAL, ProDOS® menuegesteuert im Zugriff.

Lieferumfang: Controller, Kabel, Software und Manual.

Ein Produkt von:

FRANK & BRITTING
Elektronik Entwicklungs GmbH
Langestr. 4, Postfach 1129, 7529 Forst
Telefon: 07251 / 103068-69
Telex: 7822452 fub d

Die Harddiskcontroller-Spezialisten



Darauf haben alle Apple//-Besitzer schon lange gewartet.
Jetzt bleibt die Floppykiste zu.
Einfach Rechner einschalten,
vier Betriebssysteme warten auf Ihr Kommando. Welcher Profirechner kann das schon?
DM 4.560.– incl. MWST komplett bei Ihrem örtlichen Fachhändler.

Ein Produkt von:

FRANK & BRITTING
Elektronik Entwicklungs GmbH
Langestr. 4, Postfach 1129, 7529 Forst
Telefon 07251 / 103068-69
Telex 7822452 fub d

Die Harddiskcontroller-Spezialisten

```
8439: 99 2C 81  65            STA  XBYTZ,Y
843C: B9 4A 81  66            LDA  XBITZ,Y
843F: 99 3A 82  67            STA  XBITL,Y
8442: B9 5A 8Ø  68            LDA  XBITD,Y
8445: 99 4A 81  69            STA  XBITZ,Y
8448: B9 68 81  7Ø            LDA  YWRTZ,Y
844B: 99 58 82  71            STA  YWRTL,Y
844E: B9 78 8Ø  72            LDA  YWRTD,Y
8451: 99 68 81  73            STA  YWRTZ,Y
8454: B9 86 81  74            LDA  YMATZ,Y
8457: 99 76 82  75            STA  YMATL,Y
845A: B9 96 8Ø  76            LDA  YMATD,Y
845D: 99 86 81  77            STA  YMATZ,Y
8460: B9 A4 81  78            LDA  DRWTZ,Y
8463: 99 94 82  79            STA  DRWTL,Y
8466: B9 B4 8Ø  8Ø            LDA  DRWTD,Y
8469: 99 A4 81  81            STA  DRWTZ,Y
846C: B9 C2 81  82            LDA  MSKTZ,Y
846F: 99 B2 82  83            STA  MSKTL,Y
8472: B9 D2 8Ø  84            LDA  MSKTD,Y
8475: 99 C2 81  85            STA  MSKTZ,Y
                86  *
8478: B9 ØØ 8Ø  87            LDA  SPRTD,Y    ;Sprite = Ø?
847B: FØ 28     88            BEQ  ASTA5      ;wenn ja, weiter
847D: B9 1E 8Ø  89            LDA  XWRTD,Y    ;Lade XWerRt
848Ø: AA        9Ø            TAX
8481: BD E7 87  91            LDA  BYTAB,X    ;ermittel XBYTE
8484: 99 3C 8Ø  92            STA  XBYTD,Y
8487: 99 2C 81  93            STA  XBYTZ,Y
848A: BD 5B 87  94            LDA  BITAB,X    ;und XBIT
848D: 99 5A 8Ø  95            STA  XBITD,Y
849Ø: 99 4A 81  96            STA  XBITZ,Y
                97  *
8493: B9 B4 8Ø  98            LDA  DRWTD,Y    ;8 Bits gesetzt?
8496: 1Ø ØA     99            BPL  ASTA4      ;nein, dann weiter
8498: A9 ØØ     1ØØ           LDA  #$ØØ       ;Sprite später
849A: 99 FØ 8Ø  1Ø1           STA  SPRTZ,Y    ;nicht löschen
849D: 2Ø A5 85  1Ø2           JSR  UEBERZ     ;Sprite zeichnen
84AØ: DØ Ø3     1Ø3           BNE  ASTA5
                1Ø4  *
84A2: 2Ø 24 85  1Ø5  ASTA4    JSR  SZEICH     ;Sprite zeichnen
84A5: E6 19     1Ø6  ASTA5    INC  POS        ;POS-Zähler erhöhen
84A7: A4 19     1Ø7           LDY  POS
84A9: CØ 1E     1Ø8           CPY  #ANZAHL    ;Letzte Position?
84AB: FØ Ø3     1Ø9           BEQ  ASTA6      ;wenn ja, -> fertig
84AD: 4C 18 84  11Ø           JMP  ASTA3
84BØ: 6Ø        111  ASTA6    RTS
                112  *
                113  *-------------------------------
                114  *       Unterprogramme
                115  *
84B1: B9 EØ 81  116  SLOESCH  LDA  SPRTL,Y    ;SPR-Nummer laden
84B4: ØA        117           ASL             ;A = Sprite-Nummer
84B5: A8        118           TAY
84B6: B9 ØØ 83  119           LDA  STAB,Y     ;Sprite-Ad. laden
84B9: 85 1D     12Ø           STA  STLO       ;Lo-Byte
84BB: C8        121           INY
84BC: B9 ØØ 83  122           LDA  STAB,Y
84BF: 85 1E     123           STA  STHI       ;Hi-Byte
                124  *
84C1: A4 19     125           LDY  POS        ;Tab.-Pos. laden
84C3: B9 3A 82  126           LDA  XBITL,Y    ;XBI,Y - Bitsprite
84C6: A8        127           TAY
84C7: B1 1D     128           LDA  (STLO),Y   ;Bitsprite-Adresse
84C9: AA        129           TAX             ;Lo-Byte
84CA: C8        13Ø           INY
84CB: B1 1D     131           LDA  (STLO),Y   ;Hi-Byte
84CD: 85 1E     132           STA  STHI
84CF: 86 1D     133           STX  STLO
                134  *
84D1: AØ ØØ     135           LDY  #$ØØ
84D3: B1 1D     136           LDA  (STLO),Y   ;Länge des Sprites
84D5: 85 1F     137           STA  XLAE
84D7: C8        138           INY
84D8: B1 1D     139           LDA  (STLO),Y   ;Breite des Sprites
84DA: 85 Ø9     14Ø           STA  YCOUNT
                141  *
84DC: A4 19     142           LDY  POS
84DE: B9 1C 82  143           LDA  XBYTL,Y    ;XBY,Y - X-Byte
84E1: 85 1A     144           STA  XBYTE
84E3: B9 58 82  145           LDA  YWRTL,Y    ;YWR,Y - Y-Wert
84E6: 85 1C     146           STA  YWERT
84E8: B9 76 82  147           LDA  YMATL,Y    ;YMAX des Sprites
84EB: 85 E3     148           STA  YMAX
                149  *
84ED: A5 1F     15Ø  SLOESCH1 LDA  XLAE       ;Schleife für Y
84EF: 85 Ø8     151           STA  XCOUNT
84F1: A4 1C     152           LDY  YWERT
84F3: C4 E3     153           CPY  YMAX       ;untere Bildgrenze
84F5: 9Ø Ø4     154           BCC  SLOESCH2
84F7: AØ 29     155           LDY  #$29
84F9: BØ 1C     156           BCS  SLOESCH4
```

```
84FB: B9 9B 86  157  SLOESCH2 LDA  GADRHI,Y
84FE: Ø5 E6     158           ORA  HPAG       ;HGR-Page (Hi)
85ØØ: 85 Ø7     159           STA  SCRHI      ;abspeichern
85Ø2: Ø9 6Ø     16Ø           ORA  #%Ø11ØØØØØ ;Hintergrund (Hi)
85Ø4: 85 E1     161           STA  BUFFHI     ;abspeich. ($6ØØØ)
85Ø6: B9 DB 85  162           LDA  GADRLO,Y   ;Lo-Byte holen
85Ø9: 85 Ø6     163           STA  SCRLO      ;und
85ØB: 85 EØ     164           STA  BUFFLO     ;abspeichern
                165  *
85ØD: A4 1A     166           LDY  XBYTE
85ØF: CØ 28     167  SLOESCH3 CPY  #$28
8511: BØ Ø4     168           BCS  SLOESCH4
                169  *
8513: B1 EØ     17Ø           LDA  (BUFFLO),Y ;Hintergrundpuffer
8515: 91 Ø6     171           STA  (SCRLO),Y  ;Bildbyte
                172  *
8517: E8        173  SLOESCH4 INX
8518: C8        174           INY
8519: C6 Ø8     175           DEC  XCOUNT     ;Spaltenzähler
851B: DØ F2     176           BNE  SLOESCH3   ;Zeile fertig?
851D: E6 1C     177           INC  YWERT
851F: C6 Ø9     178           DEC  YCOUNT     ;Zeilenzähler
8521: DØ CA     179           BNE  SLOESCH1   ;Sprite fertig?
8523: 6Ø        18Ø           RTS
                181  *
                182  *-------------------------------
8524: B9 ØØ 8Ø  183  SZEICH   LDA  SPRTD,Y    ;SPR-Nummer laden
8527: ØA        184           ASL             ;A = Spritenummer
8528: A8        185           TAY
8529: B9 ØØ 83  186           LDA  STAB,Y     ;Sprite-Adresse
852C: 85 1D     187           STA  STLO       ;Lo-Byte
852E: C8        188           INY
852F: B9 ØØ 83  189           LDA  STAB,Y
8532: 85 1E     19Ø           STA  STHI       ;Hi-Byte
                191  *
8534: A4 19     192           LDY  POS        ;Tabellen-Position
8536: B9 5A 8Ø  193           LDA  XBITD,Y    ;XBI,Y - Bitsprite
8539: A8        194           TAY
853A: B1 1D     195           LDA  (STLO),Y   ;Bitsprite-Adresse
853C: AA        196           TAX             ;Lo-Byte
853D: C8        197           INY
853E: B1 1D     198           LDA  (STLO),Y   ;Hi-Byte
8540: 85 1E     199           STA  STHI
8542: 86 1D     2ØØ           STX  STLO
                2Ø1  *
8544: 8D 8F 85  2Ø2           STA  MOD+2      ;PSEUDO durch
8547: 8E 8E 85  2Ø3           STX  MOD+1      ;Sprite-Ad. ersetz.
                2Ø4  *
854A: AØ ØØ     2Ø5           LDY  #$ØØ
854C: B1 1D     2Ø6           LDA  (STLO),Y   ;Länge des Sprites
854E: 85 1F     2Ø7           STA  XLAE
8550: C8        2Ø8           INY
8551: B1 1D     2Ø9           LDA  (STLO),Y   ;Breite des Sprites
8553: 85 Ø9     21Ø           STA  YCOUNT
                211  *
8555: A4 19     212           LDY  POS
8557: B9 3C 8Ø  213           LDA  XBYTD,Y    ;XBY,Y - X-Byte
855A: 85 1A     214           STA  XBYTE
855C: B9 78 8Ø  215           LDA  YWRTD,Y    ;YWR,Y - Y-Wert
855F: 85 1C     216           STA  YWERT
8561: B9 96 8Ø  217           LDA  YMATD,Y    ;YMAX des Sprites
8564: 85 E3     218           STA  YMAX
                219  *
8566: B9 D2 8Ø  22Ø           LDA  MSKTD,Y    ;Bitmaske für Farbe
8569: 85 E2     221           STA  FMASK      ;abspeichern
856B: A2 Ø2     222           LDX  #$Ø2       ;Index für Sprite
                223  *                       ;auf 1. Byte
856D: A5 1F     224  SZEICH1  LDA  XLAE       ;Schleife für Y
856F: 85 Ø8     225           STA  XCOUNT
8571: A4 1C     226           LDY  YWERT
8573: C4 E3     227           CPY  YMAX       ;untere Bildgrenze
8575: 9Ø Ø4     228           BCC  SZEICH2
8577: AØ 29     229           LDY  #$29
8579: BØ 1D     23Ø           BCS  SZEICH4
857B: B9 9B 86  231  SZEICH2  LDA  GADRHI,Y
857E: Ø5 E6     232           ORA  HPAG       ;Bildschirmadr.
8580: 85 Ø7     233           STA  SCRHI      ;wird ermittelt
                234  *
8582: B9 DB 85  235           LDA  GADRLO,Y
8585: 85 Ø6     236           STA  SCRLO
                237  *
8587: A4 1A     238           LDY  XBYTE
8589: CØ 28     239  SZEICH3  CPY  #$28
858B: BØ ØB     24Ø           BCS  SZEICH4
                241  *
858D: BD FF FF  242  MOD      LDA  PSEUDO,X   ;Bitsprite-Adresse
8590: 11 Ø6     243  PATCH    ORA  (SCRLO),Y  ;Hintergrund lesen
8592: 29 7F     244           AND  #$7F       ;Farbbit löschen
8594: 45 E2     245           EOR  FMASK      ;Farbmaske f. Bit 7
8596: 91 Ø6     246           STA  (SCRLO),Y  ;Bildbyte zurück
                247  *
8598: E8        248  SZEICH4  INX
```

APPLE -- DISKETTEN LAUFWERKE -- APPLE

Original Disk II m. Controller + DOS 3.3 + Hdbuch	DM	995,—
Original Disk II (2. Laufwerk)	DM	825,—
Duo-Disk Station Slimline, 2 x 143KB Chinon	DM	995,—
Siemens Disk-Laufw., 143KB, m. Kabel + Gehause	DM	498,—
Abor-Disk-Laufw., Slimline, 143KB, m. Kab. + Geh.	DM	398,—
Chinon Slimline, 143KB, Superleise, m. Kab. + Geh.	DM	495,—
Teac 55F, 1 MB und. Kapazität, Shugartbus	DM	488,—
Teac 55F, kpl. im Gehause, 40/80 Track, 1MByte anschlußfertig, mit Umschaltung 40/80 + Kabel	DM	548,—
Zweitlaufwerk für Apple II C, 143 KB, mit Kabel	DM	448,—

APPLE -- INTERFACES + MAINBOARDS -- APPLE

Disk Controller f. 2 Original o. kompat. Drives	*DM	79,—
Super-Controller f. 2x Teac 55F, mit Software	**DM	188,—
80 Zeich-Karte mit 64K RAM + Softsw. f. IIe	DM	99,—
80 Zeich-Karte II, m. Softswitch, 2 Zeichensatze	*DM	159,—
16K RAM Erweiterung für II und kompatible	*DM	98,—
Z 80A Interface Karte fur CPM 2.2	*DM	79,—
Z 80B Interface Karte fur CPM 3.0, m. 64K RAM	*DM	595,—
Printer-Grafik Interface, Epson kompatible	*DM	79,—
Centronic-Parallel Text-Interface Karte	*DM	79,—
Printer-Grafik Interface, NEC/ITOH kompatibel	*DM	169,—
Anschluß Kabel f. Parallel + Grafik Interface	DM	34,—
Buffer Grafik-Interface, benutzbar fur Drucker Epson/Nec/Itoh/Okidata u. andere m. 32K Buffer	DM	348,—
Buffer Interface wie vor aber mit 64K Buffer	**DM	595,—
Anschluß Kabel f. Buffer Interface Karte	DM	39,—
232C Serielle Interface Karte	*DM	189,—
Super Serielle Interface Karte, Fullduplex	**DM	188,—
Sprach (Speech) Karte f. Sprachwiedergabe	*DM	58,—
6522 Parallell Interface Karte	*DM	149,—
Clock Karte (Datum/Uhrzeit) Ein/Ausgabe	*DM	129,—
Eprom Writer Karte (2716 + 2764)	*DM	129,—
IEEE-488 Interface Karte	*DM	398,—
Logo-Karte mit Diskette und Handbuch	DM	498,—
Musik Karte m. Diskette und Handbuch	*DM	119,—
PAL Color Interface Karte (UHF + Video)	*DM	169,—
RGB Interface Karte (f. Apple II + II+)	*DM	189,—
Wild Karte (kopiert über RAM-Bereich)	*DM	119,—
128K RAM Erweiterungs Karte m. Patchsoftware	**DM	398,—
256K RAM Erweiterungs Karte m. Patchsoftware	**DM	498,—
6809 Prozessor Exell-9 Interfacekarte	+ DM	398,—
IC-Tester Interface Karte (RAMS/TTL 54/74)	DM	448,—
Hauptplatine 48K, o. Firmware Eproms, 8 Slots	DM	388,—
Hauptplatine 64K, wie vor	DM	398,—

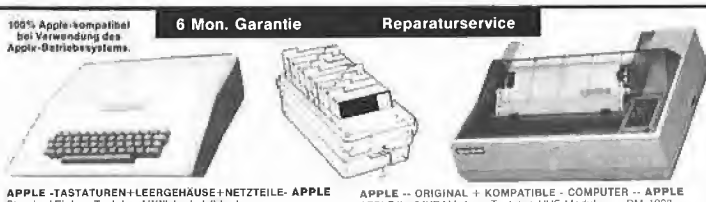
APPLE -- LEERPLATINEN -- APPLE

Leerplatinen der obigen Interface Karten sind alle in vergoldet. m. Bestuck-Aufdruck + Best.-Plan lieferbar

Leerplatinen mit der Kennzeichnung	**DM	39,—
Leerplatinen mit der Kennzeichnung	*DM	24,50
Leerplatinen mit der Kennzeichnung	+ DM	33,—
Experimentier Platine, für Apple Slot EX300	DM	29,90
Leerplatine Motherboard, 48K, mit Best.-Aufdruck	DM	66,—
Leerplatine Motherboard 64K, mit 6502 + Z80	DM	99,—

100% Apple-kompatibel bei Verwendung des Apple-Betriebssystems.

6 Mon. Garantie **Reparaturservice**



APPLE -TASTATUREN+LEERGEHÄUSE+NETZTEILE- APPLE

Standard Einbau-Tastatur, ASCII, freibel. 9 Tasten, Doppelbelg. aller Tasten, Groß+Kleinschr. Nr. N26	DM	139,—
Tastatur wie vor, jedoch mit 15er Block Nr.N67	DM	178,—
Tastatur Nr. N67, mit sep. Gehause, kpl. m. Kabel	DM	249,—
Seperate Tastatur IBM-Look, anschlußfertig mit Kabel, Dopp. belegt Tasten, frei prog. Funkt. Tasten	DM	298,—
Seperate Tastatur, Eprom progr.-bar, Cursorblock Tastenfeld mit 24 Funktionen, Deutsch o. ASCII	DM	398,—
Leergehause Standard, passend f. Tastatur Nr. N26	DM	98,—
dto. wie vor, jedoch passend fur 15er Tast. N67	DM	119,—
Leergehause wie Mewa 9000, fur Einbau von zwei Slimline-Laufwerken + Tastaturanschluß, Plastic	DM	159,—
Leergehause wie vor, jedoch in Metall-Ausfuhrung	DM	198,—

Schaltnetzteile fur APPLE u. kompatible Rechner

+5V/5 A -5V/0,5A +12V/2,5A -12V/0,5a	N74	DM	119,80
+5V/7 5A -5V/0,5A +12V/2A -12V/0,5A	N75	DM	149,50

Apple IIe*- Compatible mit 80-Zeichen-Karte!

Komplett mit Gehäuse und IIe Tastatur, getrennte Cursor-Steuerung, mit Apple-Tasten (gleichlaufend - offen!

128 K.. DM 898,–

Exakt wie Originalvon auf Knopf Rechner ohne Firmware

Zusatzkarten für Apple IIe		
80-Zeichen-Karte	DM	88,—
80-Zeichen-Karte + 64-K-RAM Preise z. Anlage		
Motherboard IIe, bestückt + gepruft	DM	598,—

APPLE -- ORIGINAL + KOMPATIBLE - COMPUTER -- APPLE

APPLE IIe, 64KRAM, Asci-Tastat. + UHF-Modul	DM	1998,—
APPLE IIe, Einstiegspaket = Computer 64KRAM, + Philips-Monitor 18 Mhz grun + Siemens F122 - Laufwerk 143K m. Controller + Lerndiskette	DM	2998,—
APPLE II C, 128 KRAM, ASCII Tastatur	DM	2795,—
APPLE II C, wie vor, jedoch deutsche Tastatur	DM	2898,—
MACINTOSH System komplett m. Mouse+Tastatur + Macwrite/Macpaint, Systemguide-Diskette	DM	5995,—
MEWA-48K, Apple kompat. Netzt. 5Amp, Tastatur mit Dopp-Belegung + 10 frei Prog. Tasten, mit UHF-Mod., ohne Firmware Eproms.	DM	698,—
MEWA-64K wie vor jedoch mit 15er Block	DM	728,—
MEWA 9000er Serie, mit seper. Tastat. DIN o. ASCII m. dopp. belegt. Funkt-Tasten, Cursorbl. + 15er Block Netzt. 5 Amp., Gehause fur 2x Slimline-Laufwerke UHF-Modulator, o. Firmw.-Proms (12KROM), 48K RAM	DM	998,—
MEWA 9000-64-C, wie vor jedoch 64K RAM	DM	1098,—
MEWA 9000-64-C Einstiegspaket, wie vor, jedoch mit 1 Disk Drive eingeb. + Disk-Contr. + Monitor, grun	DM	1898,—
IBM-Look Gehause fur MEWA 9000 Serie Aufpreis	DM	39,—

MATRIX- UND TYPENRAD - DRUCKER

CP-80 Matrix-Drucker, Epson kompat., vollgrafik	DM	739,—
GP-80 X mit angeb. Interface fur VC 64 f. Sem. B.	DM	899,—
SPEEDY 100-80, Epson kompat., 100 Z/s, v.-grafik	DM	799,—
GEMINI 10X, Neu mit Textpp., 100 Z/s, 9x9 Mat.	DM	998,—
GEMINI 15X, wie 10X, jedoch bis 375 mm Papierbr.	DM	1198,—
ITOH 8510 B, die Zuverlässigkeit in Person	DM	1475,—
JUKI 6100, Typenraddr., 22 Z/s, 14-Typen/sec.	DM	1495,—

Computer-Artikel Nachnahmeversand unfrei, Zwischenverkauf vorbehalten. Angebote freibleibend unter Anerkennung unserer Lieferbedingungen. Technische Änderungen vorbehalten. *Apple ist eingetragenes Warenzeichen der Fa. Apple-Computer Inc., Kalifornien. Ware mit Rückgaberecht, besonders gekennzeichnet, muß frei zurückgeschickt werden. *IBM* ist eingetragenes Warenzeichen der Firma IBM GmbH Ffm.

COMPUTER CENTER CONEX
5650 SOLINGEN 11 Postfach 11 02 06
Telefon (02 12) 7 54 49

+

ERICH-WILLI MEYER
6343 FROHNHAUSEN Postfach [illegible] 11
Telefon (0 27 71) 3 50 71

IBM * KOMPATIBEL ***** IBM *** KOMPATIBEL ***

Alle Teile unterliegen einer sorgfältigen Endkontrolle und wir übernehmen daher volle Garantie für die Funktionsfähigkeit, sowohl für die gelieferten Leerplatinen, als auch für die fertig bestückten Boards und Interface Karten, die fast ohne Ausnahme mit gesockelten IC's geliefert werden.

IBM -- Kompatible Interface Karten und Mainboards -- IBM

Disk Controller Karte fur 2 Disk-Drives	*KL-2020	DM	129,—
Color Video Board,	*KL-2050	DM ab	198,—
Monochrome Video Boards	*KL-2060	DM ab	270,—
RS 232 Drucker interface Karte	KL-2074	DM	148,—
Centronics Parallell Interface Karte	KL-2072	DM	95,—
Multifunktion Karte, RAM-Bereich, RS232, Parallell, Clock, mit O K bestuckt	*KL-2040	DM	298,—
dto. wie vor, jedoch mit 128K bestuckt	*KL-2041	DM	348,—
dto. wie vor, jedoch mit 256K bestuckt	*KL-2042	DM	398,—
512K RAM Karte, mit O K bestuckt	KL-2095	DM	248,—
dto. wie vor, mit 128K bestuckt	KL-2096	DM	298,—
Multi I/O Interface Board, RS232, Disk-Controller, Centronics, Clock, Joystick und Lightpen-Port	*KL-2071	DM	498,—
Eprom Writer Karte (bis 128K benutzbar)			598,—
Hardisk/Winchester Host-Adapter Karte	*KL-2022	DM ab	598,—
Mainboard XT Version, 8 Slots, 0 bestückt			
mit 1 Boot-Eprom, 6 Eprom-Platze frei	*KL-1004	DM	449,—
dto. wie vor, jedoch mit 128K bestuckt	*KL-1005	DM	498,—
dto. wie vor, jedoch mit 256K bestuckt	*KL-1006	DM	549,—

Bei den mit * gekennzeichneten Artikeln gehört zum Lieferumfang ein Manual, und z. T. ein Schaltbild.

Preislisten + Prospekte kostenlos gegen frankierten Rückumschlag

DISKETTEN-BOXEN

Profi Disk Box m. Schloß Klars.-Deckel 100 Disks	DM	39,—
Disk Box mit Klars.-Deckel, ca. 70 Disketten 5"	DM	26,—
Hartbox für 10 Disketten, mit Klappdeckel	DM	6,90

IBM -- Kompatible Leerplatinen + Boards -- IBM

Disk Controller f. 2 Drives	DM	39,—
Color Grafik Video Board	DM	68,—
RS232 Interface Karte	DM	39,—
Centronics Parallell Interface Karte	DM	39,—
Multifunktion Interface Board	DM	78,—
Eprom Writer Interface Karte	DM	78,—
Multi I/O Board	DM	72,—
Mainboard XT Version, 8 Slots	DM	98,—
RAM Card fur Speicher-Erweiterung	DM	68,—

Alle Leerplatinen werden mit Bestuckungsplan geliefert. Schaltbilder, soweit verfugbar, Lieferung gegen Aufpreis.

IBM -- Gehäuse/Netzteile/Tastaturen/Diskdrives -- IBM

Rechner Leergehause f. Einbau v. 2 Drives	DM	148,—
Profi-Tastatur, m. Kabel im Gehause, ASCII	DM	398,—
dto. wie vor, jedoch deutsche DIN Belegung	DM	418,—
Standard Tastatur, IBM-Look, ASCII	DM	235,—
Netzteil mit Ventilator 150 Watt	DM	298,—
dto. wie vor jedoch 135 Watt	DM	235,—
Diskettten Laufwerk 2x40 Track DS/DD 48	DM	333,—
Harddisk 10MByte kpl. m. Controller	DM	2096,—

Apple und IBM kompatible Computer

16K, Z80, Diskcontroler je	75,–
80 Zeichenkarte mit Softswitch 2 Zeichensätze	149,–
Motherboard 48K ohne Firmware	419,–
Erphi-controler mit Autopatch	300,–
Siemenslaufwerk F 122	515,–
TEAC FD-55B 2 × 40 Track	448,–
TEAC FD-55F 2 × 80 Track	475,–
FD4 Spezialcontroler für Laufwerke mit bis zu 2 × 80 Track	130,–
Drucker Star SG 10	**940,–**
Monochrome Monitore	ab 375,–
Farbmonitore	ab 998,–
Tastaturen für IBM und Apple	ab 330,–

Versand nur per Nachnahme oder Vorkasse.
Weiteres Zubehör für Apple und IBM gegen frankierten Rückumschlag.

Preissenkung:
128K Karte (Saturn kompatibel) . **375,–**
Preissenkung 4164-200 ns
Mindestabnahme 20 Stck. 3,90

Zusatzkarten und Motherboard ausnahmslos deutsche Fertigung mit ausgesuchten Bauteilen.

Ulf Mohwinkel Electronic
Berliner Straße 73 Pf: 250 166
5090 Leverkusen Fettehenne
Telefon 02 14 / 9 37 81

ABACOMP

Unsere Apple - Hits

Preise nur solange Vorrat reicht

Computer 48 KB o. Firmware	750,– DM
Monitor 9", grüne Röhre	200,– DM
Disk-Controler für Apple	75,– DM
16 K-Karte	90,– DM
Z-80-Karte	80,– DM
80 Zeichen-Karte	140,– DM
Original IBS-Drucker-Karte komplett m. Anschlußkabel (Druckertype angeben)	248,– DM
Disk-Laufwerk Slim-Line	348,– DM
Joy-Stick od. **Paddles** (2er-Set) je	28,– DM
Drucker Centronics GLP 4	465,– DM
Drucker SP 80 (80 Zchn./sec.)	580,– DM
Drucker Comdata M 100	690,– DM
Drucker Riteman	660,– DM
Drucker Panasonic KX-P 1091	880,– DM
Drucker Riteman F +	960,– DM
Drucker Epson LX-80	880,– DM
Drucker Epson FX-85	1480,– DM
Typenraddrucker ab	725,– DM
10 Disketten **Qualimex Durolife**	34,– DM

Bestellungen bitte nur schriftlich an:
Abacomp GmbH, Kransberger Weg 24, 6 Frankfurt 50
Mindestbestellwert: 50,– DM
Tel. Auskunft: Mo-Sa 8-9.30 Uhr unter (069) 70 03 08
Ladenöffnung: Mo-Fr 10-12 und 14-18 Uhr in Ffm. 90, Ginnheimer Landstraße 1

HOBBY ELEKTRONIK '85
7. 11. – 10. 11.
Halle 14, Stand 1437

Unser November-Angebot:

Profimax IIe, APPLE IIe kompatibel, IBM-Look	**nur 1398.–**
Profimax III, APPLE II kompatibel, IBM-Look	**nur 1248.–**
Chinon-Laufwerk 051A	**nur 398.–**
Video-Digitalisierer incl. Software	**nur 298.–**
STAR sg-10, original	**nur 998.–**
Centronics-Interface, grafikfähig	**nur 118.–**

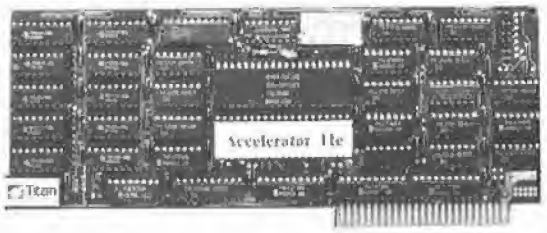
D.O.S Computersysteme
der Partner für Schule, Hochschule, Forschung, Fertigung

Am Kühnbach 42, 7170 Schwäb. Hall 11
Tel. 07 91/5 17 36

```
8599: C8          249           INY
859A: C6 Ø8       25Ø           DEC  XCOUNT     ;Spaltenzähler
859C: DØ EB       251           BNE  SZEICH3    ;Zeile fertig?
859E: E6 1C       252           INC  YWERT
85AØ: C6 Ø9       253           DEC  YCOUNT     ;Zeilenzähler
85A2: DØ C9       254           BNE  SZEICH1    ;Sprite fertig?
85A4: 6Ø          255           RTS
                  256 *-----------------------------
                  257 *
85A5: A9 EA       258 UEBERZ    LDA  #$EA       ;$EA = NOP
85A7: 8D 9Ø 85    259           STA  PATCH      ;ORA (SCRLO),Y
85AA: 8D 91 85    26Ø           STA  PATCH+1    ;löschen
85AD: 2Ø 24 85    261           JSR  SZEICH
85BØ: A9 11       262           LDA  #$11       ;ORA (),Y OPCODE
85B2: 8D 9Ø 85    263           STA  PATCH
85B5: A9 Ø6       264           LDA  #SCRLO
85B7: 8D 91 85    265           STA  PATCH+1
85BA: 6Ø          266           RTS
                  267 ******************************
                  268 *                            *
                  269 * Löschen der POS-Tabelle    *
                  27Ø *                            *
                  271 ******************************
                  272 *
85BB: A9 8Ø       273 POSINIT   LDA  #>PTAB
85BD: 85 E1       274           STA  BUFFHI
85BF: A9 ØØ       275           LDA  #$ØØ
85C1: 85 EØ       276           STA  BUFFLO
85C3: AØ ØØ       277           LDY  #<PTAB
85C5: A9 ØØ       278           LDA  #$ØØ
85C7: AA          279           TAX
85C8: 8A          28Ø POSL1     TXA
85C9: 91 EØ       281           STA  (BUFFLO),Y
85CB: C8          282           INY
85CC: DØ Ø2       283           BNE  POSL2
85CE: E6 E1       284           INC  BUFFHI
85DØ: CØ DØ       285 POSL2     CPY  #<POSEND
85D2: DØ F4       286           BNE  POSL1
85D4: A5 E1       287           LDA  BUFFHI
85D6: C9 82       288           CMP  #>POSEND
85D8: DØ EE       289           BNE  POSL1
85DA: 6Ø          29Ø           RTS
                  291 *
                  292 ********************************
                  293 *        HGR-ADR-Tabelle       *
                  294 *  YWERT -> Bildschirmadresse  *
                  295 ********************************
                  296 *
85DB: ØØ ØØ ØØ    297 GADRLO    HEX  ØØØØØØØØØØØØØØØØ
85DE: ØØ ØØ ØØ ØØ ØØ
85E3: 8Ø 8Ø 8Ø    298           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
85E6: 8Ø 8Ø 8Ø 8Ø 8Ø
85EB: ØØ ØØ ØØ    299           HEX  ØØØØØØØØØØØØØØØØ
85EE: ØØ ØØ ØØ ØØ ØØ
85F3: 8Ø 8Ø 8Ø  3ØØ           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
85F6: 8Ø 8Ø 8Ø 8Ø 8Ø
85FB: ØØ ØØ ØØ  3Ø1           HEX  ØØØØØØØØØØØØØØØØ
85FE: ØØ ØØ ØØ ØØ ØØ
86Ø3: 8Ø 8Ø 8Ø  3Ø2           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
86Ø6: 8Ø 8Ø 8Ø 8Ø 8Ø
86ØB: ØØ ØØ ØØ  3Ø3           HEX  ØØØØØØØØØØØØØØØØ
86ØE: ØØ ØØ ØØ ØØ ØØ
8613: 8Ø 8Ø 8Ø  3Ø4           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
8616: 8Ø 8Ø 8Ø 8Ø 8Ø
861B: 28 28 28  3Ø5           HEX  2828282828282828
861E: 28 28 28 28 28
8623: A8 A8 A8  3Ø6           HEX  A8A8A8A8A8A8A8A8
8626: A8 A8 A8 A8 A8
862B: 28 28 28  3Ø7           HEX  2828282828282828
862E: 28 28 28 28 28
8633: A8 A8 A8  3Ø8           HEX  A8A8A8A8A8A8A8A8
8636: A8 A8 A8 A8 A8
863B: 28 28 28  3Ø9           HEX  2828282828282828
863E: 28 28 28 28 28
8643: A8 A8 A8  31Ø           HEX  A8A8A8A8A8A8A8A8
8646: A8 A8 A8 A8 A8
864B: 28 28 28  311           HEX  2828282828282828
864E: 28 28 28 28 28
8653: A8 A8 A8  312           HEX  A8A8A8A8A8A8A8A8
8656: A8 A8 A8 A8 A8
865B: 5Ø 5Ø 5Ø  313           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
865E: 5Ø 5Ø 5Ø 5Ø 5Ø
8663: DØ DØ DØ  314           HEX  DØDØDØDØDØDØDØDØ
8666: DØ DØ DØ DØ DØ
866B: 5Ø 5Ø 5Ø  315           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
866E: 5Ø 5Ø 5Ø 5Ø 5Ø
8673: DØ DØ DØ  316           HEX  DØDØDØDØDØDØDØDØ
8676: DØ DØ DØ DØ DØ
867B: 5Ø 5Ø 5Ø  317           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
867E: 5Ø 5Ø 5Ø 5Ø 5Ø
8683: DØ DØ DØ  318           HEX  DØDØDØDØDØDØDØDØ
8686: DØ DØ DØ DØ DØ
```

```
868B: 5Ø 5Ø 5Ø  319           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
868E: 5Ø 5Ø 5Ø 5Ø 5Ø
8693: DØ DØ DØ  32Ø           HEX  DØDØDØDØDØDØDØDØ
8696: DØ DØ DØ DØ DØ
                321 *
                322 *-----------------------------
869B: ØØ Ø4 Ø8  323 GADRHI    HEX  ØØØ4Ø8ØC1Ø14181C
869E: ØC 1Ø 14 18 1C
86A3: ØØ Ø4 Ø8  324           HEX  ØØØ4Ø8ØC1Ø14181C
86A6: ØC 1Ø 14 18 1C
86AB: Ø1 Ø5 Ø9  325           HEX  Ø1Ø5Ø9ØD1115191D
86AE: ØD 11 15 19 1D
86B3: Ø1 Ø5 Ø9  326           HEX  Ø1Ø5Ø9ØD1115191D
86B6: ØD 11 15 19 1D
86BB: Ø2 Ø6 ØA  327           HEX  Ø2Ø6ØAØE12161A1E
86BE: ØE 12 16 1A 1E
86C3: Ø2 Ø6 ØA  328           HEX  Ø2Ø6ØAØE12161A1E
86C6: ØE 12 16 1A 1E
86CB: Ø3 Ø7 ØB  329           HEX  Ø3Ø7ØBØF13171B1F
86CE: ØF 13 17 1B 1F
86D3: Ø3 Ø7 ØB  33Ø           HEX  Ø3Ø7ØBØF13171B1F
86D6: ØF 13 17 1B 1F
86DB: ØØ Ø4 Ø8  331           HEX  ØØØ4Ø8ØC1Ø14181C
86DE: ØC 1Ø 14 18 1C
86E3: ØØ Ø4 Ø8  332           HEX  ØØØ4Ø8ØC1Ø14181C
86E6: ØC 1Ø 14 18 1C
86EB: Ø1 Ø5 Ø9  333           HEX  Ø1Ø5Ø9ØD1115191D
86EE: ØD 11 15 19 1D
86F3: Ø1 Ø5 Ø9  334           HEX  Ø1Ø5Ø9ØD1115191D
86F6: ØD 11 15 19 1D
86FB: Ø2 Ø6 ØA  335           HEX  Ø2Ø6ØAØE12161A1E
86FE: ØE 12 16 1A 1E
87Ø3: Ø2 Ø6 ØA  336           HEX  Ø2Ø6ØAØE12161A1E
87Ø6: ØE 12 16 1A 1E
87ØB: Ø3 Ø7 ØB  337           HEX  Ø3Ø7ØBØF13171B1F
87ØE: ØF 13 17 1B 1F
8713: Ø3 Ø7 ØB  338           HEX  Ø3Ø7ØBØF13171B1F
8716: ØF 13 17 1B 1F
871B: ØØ Ø4 Ø8  339           HEX  ØØØ4Ø8ØC1Ø14181C
871E: ØC 1Ø 14 18 1C
8723: ØØ Ø4 Ø8  34Ø           HEX  ØØØ4Ø8ØC1Ø14181C
8726: ØC 1Ø 14 18 1C
872B: Ø1 Ø5 Ø9  341           HEX  Ø1Ø5Ø9ØD1115191D
872E: ØD 11 15 19 1D
8733: Ø1 Ø5 Ø9  342           HEX  Ø1Ø5Ø9ØD1115191D
8736: ØD 11 15 19 1D
873B: Ø2 Ø6 ØA  343           HEX  Ø2Ø6ØAØE12161A1E
873E: ØE 12 16 1A 1E
8743: Ø2 Ø6 ØA  344           HEX  Ø2Ø6ØAØE12161A1E
8746: ØE 12 16 1A 1E
874B: Ø3 Ø7 ØB  345           HEX  Ø3Ø7ØBØF13171B1F
874E: ØF 13 17 1B 1F
8753: Ø3 Ø7 ØB  346           HEX  Ø3Ø7ØBØF13171B1F
8756: ØF 13 17 1B 1F
                347 *
                348 ********************************
                349 *  Byte- und Bit-Umrechnugs-   *
                35Ø *         tabelle              *
                351 *  XWERT -> XBYTE / XBIT       *
                352 ********************************
                353 *
                354 *
875B: ØØ Ø4 Ø8  355 BITAB     HEX  ØØØ4Ø8ØC02Ø6ØA
875E: ØC Ø2 Ø6 ØA
8762: ØØ Ø4 Ø8  356           HEX  ØØØ4Ø8ØC02Ø6ØA
8765: ØC Ø2 Ø6 ØA
8769: ØØ Ø4 Ø8  357           HEX  ØØØ4Ø8ØC02Ø6ØA
876C: ØC Ø2 Ø6 ØA
877Ø: ØØ Ø4 Ø8  358           HEX  ØØØ4Ø8ØC02Ø6ØA
8773: ØC Ø2 Ø6 ØA
8777: ØØ Ø4 Ø8  359           HEX  ØØØ4Ø8ØC02Ø6ØA
877A: ØC Ø2 Ø6 ØA
877E: ØØ Ø4 Ø8  36Ø           HEX  ØØØ4Ø8ØC02Ø6ØA
8781: ØC Ø2 Ø6 ØA
8785: ØØ Ø4 Ø8  361           HEX  ØØØ4Ø8ØC02Ø6ØA
8788: ØC Ø2 Ø6 ØA
878C: ØØ Ø4 Ø8  362           HEX  ØØØ4Ø8ØC02Ø6ØA
878F: ØC Ø2 Ø6 ØA
8793: ØØ Ø4 Ø8  363           HEX  ØØØ4Ø8ØC02Ø6ØA
8796: ØC Ø2 Ø6 ØA
879A: ØØ Ø4 Ø8  364           HEX  ØØØ4Ø8ØC02Ø6ØA
879D: ØC Ø2 Ø6 ØA
87A1: ØØ Ø4 Ø8  365           HEX  ØØØ4Ø8ØC02Ø6ØA
87A4: ØC Ø2 Ø6 ØA
87A8: ØØ Ø4 Ø8  366           HEX  ØØØ4Ø8ØC02Ø6ØA
87AB: ØC Ø2 Ø6 ØA
87AF: ØØ Ø4 Ø8  367           HEX  ØØØ4Ø8ØC02Ø6ØA
87B2: ØC Ø2 Ø6 ØA
87B6: ØØ Ø4 Ø8  368           HEX  ØØØ4Ø8ØC02Ø6ØA
87B9: ØC Ø2 Ø6 ØA
87BD: ØØ Ø4 Ø8  369           HEX  ØØØ4Ø8ØC02Ø6ØA
87CØ: ØC Ø2 Ø6 ØA
```

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



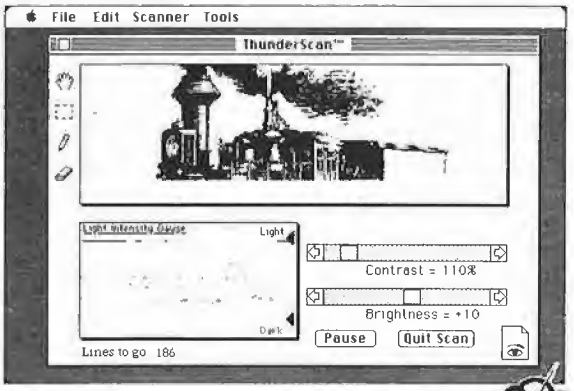
Jetzt laufen VisiCalc®, Apple Writer, PASCAL, BASIC, Datenbanken usw. endlich ohne langen Zeitverlust. Stecken Sie einfach die ACCELERATOR IIe Karte in irgendeinen Slot und beobachten Sie, wir Ihr Apple loslegt!
ACCELERATOR IIe besitzt seinen eigenen schnellen 6502 Prozessor und 80 K-Byte Hochgeschwindigkeitsspeicher,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

pandasoft Dr.-Ing. Eden

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

ThunderScan.™

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



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

ThunderScan

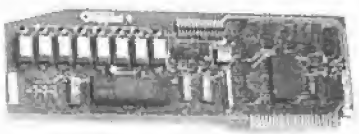
pandasoft Dr.-Ing. Eden

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

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

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

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



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

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

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

pandasoft Dr.-Ing. Eden

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

Sie haben einen Apple...

wir haben die Software...

und die Hardware...

wir haben die Bücher...

und die Zeitschriften...

Fordern Sie unseren Gratiskatalog an!

ALLES FÜR DEN APPLE II+, IIe, IIc UND MACINTOSH

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

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

```
87C4: ØØ Ø4 Ø8  37Ø          HEX  ØØØ4Ø8ØC02Ø6ØA
87C7: ØC Ø2 Ø6 ØA
87CB: ØØ Ø4 Ø8  371          HEX  ØØØ4Ø8ØC02Ø6ØA
87CE: ØC Ø2 Ø6 ØA
87D2: ØØ Ø4 Ø8  372          HEX  ØØØ4Ø8ØC02Ø6ØA
87D5: ØC Ø2 Ø6 ØA
87D9: ØØ Ø4 Ø8  373          HEX  ØØØ4Ø8ØC02Ø6ØA
87DC: ØC Ø2 Ø6 ØA
87EØ: ØØ Ø4 Ø8  374          HEX  ØØØ4Ø8ØC02Ø6ØA
87E3: ØC Ø2 Ø6 ØA
                375  *
                376  *-------------------------------
                377  *
87E7: ØØ ØØ ØØ  378  BYTAB    HEX  ØØØØØØØØ1Ø1Ø1
87EA: ØØ Ø1 Ø1 Ø1
87EE: Ø2 Ø2 Ø2  379          HEX  Ø2Ø2Ø2Ø2Ø3Ø3Ø3
87F1: Ø2 Ø3 Ø3 Ø3
87F5: Ø4 Ø4 Ø4  38Ø          HEX  Ø4Ø4Ø4Ø4Ø5Ø5Ø5
87F8: Ø4 Ø5 Ø5 Ø5
87FC: Ø6 Ø6 Ø6  381          HEX  Ø6Ø6Ø6Ø6Ø7Ø7Ø7
87FF: Ø6 Ø7 Ø7 Ø7
8803: Ø8 Ø8 Ø8  382          HEX  Ø8Ø8Ø8Ø8Ø9Ø9Ø9
8806: Ø8 Ø9 Ø9 Ø9
880A: ØA ØA ØA  383          HEX  ØAØAØAØAØBØBØB
880D: ØA ØB ØB ØB
8811: ØC ØC ØC  384          HEX  ØCØCØCØCØDØDØD
8814: ØC ØD ØD ØD
8818: ØE ØE ØE  385          HEX  ØEØEØEØEØFØFØF
881B: ØE ØF ØF ØF
881F: 1Ø 1Ø 1Ø  386          HEX  1Ø1Ø1Ø1Ø111111
8822: 1Ø 11 11 11
8826: 12 12 12  387          HEX  12121212131313
8829: 12 13 13 13
882D: 14 14 14  388          HEX  14141414151515
8830: 14 15 15 15
8834: 16 16 16  389          HEX  16161616171717
8837: 16 17 17 17
883B: 18 18 18  39Ø          HEX  18181818191919
883E: 18 19 19 19
8842: 1A 1A 1A  391          HEX  1A1A1A1A1B1B1B
8845: 1A 1B 1B 1B
8849: 1C 1C 1C  392          HEX  1C1C1C1C1D1D1D
884C: 1C 1D 1D 1D
8850: 1E 1E 1E  393          HEX  1E1E1E1E1F1F1F
8853: 1E 1F 1F 1F
8857: 2Ø 2Ø 2Ø  394          HEX  2Ø2Ø2Ø2Ø212121
885A: 2Ø 21 21 21
885E: 22 22 22  395          HEX  22222222232323
8861: 22 23 23 23
8865: 24 24 24  396          HEX  24242424252525
8868: 24 25 25 25
886C: 26 26 26  397          HEX  26262626272727
886F: 26 27 27 27
8873: 28 28 28  398          HEX  28282828292929
8876: 28 29 29 29
                399  *
                4ØØ  *-------------------------------
                4Ø1  *
                4Ø2  * Der nachfolgende Teil wird
                4Ø3  * nur zur Label-Berechnung
                4Ø4  * gebraucht.
                4Ø5  * Daher sollte der erzeugte
                4Ø6  * Objektcode an dieser Stelle
                4Ø7  * abgespeichert werden.
                4Ø8  * Beim MERLIN erfogt dies durch:
                4Ø9  *
                41Ø           SAV  ASTA

Object saved as ASTA,A$84ØØ,L$Ø47A

                411  *
                412  *********************************
                413  *   Aufbau der Sprite-Tabelle   *
                414  *                               *
                415  * STAB DA $ØØØØ ; Sprite Ø      *
                416  *                 nicht erlaubt *
                417  *      DA SPRITE1               *
                418  *      DA SPRITE2               *
                419  *       ...                     *
                42Ø  *                               *
                421  * SPRITE1:                      *
                422  *    DA  Sprite für Bit 1       *
                423  *    DA  Sprite für Bit 2       *
                424  *      ...                      *
                425  *    DA  Sprite für Bit 7       *
                426  * SBIT1 DFB Länge,Breite        *
                427  *       DFB Bytes               *
                428  *                               *
                429  *   Codierung eines Sprites     *
                43Ø  *                               *
                431  *   ..**.. = %ØØ11ØØ = $ØB      *
                432  *   .*..*. = %Ø1ØØ1Ø = $12      *
                433  *   ****** = %111111 = $3F      *
                434  *                               *
                435  *********************************
                436  STAB     =    $83ØØ
                437  * Ab STAB sollten nur die
                438  * Anfangsadressen der Sprites
                439  * abgespeichert werden.
                44Ø  * Die einzelnen Sprites selbst
                441  * können an anderer Stelle
                442  * im Rechner gespeichert werden.
                443  *
                444  *********************************
                445  * Aufbau der Positionstabelle   *
                446  *                               *
                447  *  TD - Zeichnen                *
                448  *  TZ - Zwischenspeichern       *
                449  *  TL - Löschen                 *
                45Ø  *                               *
                451  *  SPR    - Nr. des Sprites     *
                452  *  XWR    - X-Koordinate        *
                453  *  XBY    - X-Byte              *
                454  *  XBI    - X-Bit               *
                455  *  YWR    - Y-Byte              *
                456  *  YMA    - YMAX beim Zeichnen  *
                457  *  DRW    - ZEICHENMODUS        *
                458  *  MSK    - Farbmaske Bit 8     *
                459  *                               *
                46Ø  *********************************
                461  *
                462  PTAB     =    $8ØØØ
                463           ORG  PTAB
                464  SPRTD    DS   ANZAHL
                465  XWRTD    DS   ANZAHL
                466  XBYTD    DS   ANZAHL
                467  XBITD    DS   ANZAHL
                468  YWRTD    DS   ANZAHL
                469  YMATD    DS   ANZAHL
                47Ø  DRWTD    DS   ANZAHL
                471  MSKTD    DS   ANZAHL
                472  *-------------------------------
                473  SPRTZ    DS   ANZAHL
                474  XWRTZ    DS   ANZAHL
                475  XBYTZ    DS   ANZAHL
                476  XBITZ    DS   ANZAHL
                477  YWRTZ    DS   ANZAHL
                478  YMATZ    DS   ANZAHL
                479  DRWTZ    DS   ANZAHL
                48Ø  MSKTZ    DS   ANZAHL
                481  *-------------------------------
                482  SPRTL    DS   ANZAHL
                483  XWRTL    DS   ANZAHL
                484  XBYTL    DS   ANZAHL
                485  XBITL    DS   ANZAHL
                486  YWRTL    DS   ANZAHL
                487  YMATL    DS   ANZAHL
                488  DRWTL    DS   ANZAHL
                489  MSKTL    DS   ANZAHL
82DØ: ØØ        49Ø  POSEND   HEX  ØØ

1867 Bytes
```

SPRITE.EDIT

```
1ØØØ REM Sprite-Editor
1Ø1Ø REM für ASTA
1Ø2Ø REM von Bernd Kl[illegible]nn
1Ø3Ø REM
1Ø4Ø REM
1Ø5Ø REM SF% - Ø => Unverändert
1Ø6Ø REM        1 => Sprite leer
1Ø7Ø REM        2 => Sprite neu
1Ø8Ø REM
1Ø9Ø PRINT CHR$ (21): HIMEM: 16383
11ØØ I$ = "INS":T$ = ":":R$ = ";":AD$ = " .DA ":B$ = " BYT
     ": REM LISA
111Ø I$ = "*-- SPRITEDEF --":T$ = "":R$ = "*":AD$ = " DA
     ":B$ = " DFB ": REM MERLIN
112Ø DEF FN A(Ø) = START + X + Y * 6Ø
113Ø DEF FN F(Ø) = START + 52 + Y * 6Ø
114Ø DEF FN K(K) = K * 4 + 11
115Ø ROT= Ø: SCALE= 1
116Ø GOTO 367Ø
117Ø REM -= Sprites zeigen =-
118Ø W = PEEK ( FN A(Ø)) * 3:F = PEEK ( FN F(Ø))
119Ø HCOLOR= W + F * 4
12ØØ HPLOT 188 + X,11 + Y: HPLOT 235 + X,11 + Y
121Ø HCOLOR= W + (1 - F) * 4
122Ø HPLOT 188 + X,6Ø + Y: HPLOT 235 + X,6Ø + Y
```

```
123Ø HCOLOR= W: DRAW 4 AT FN K(X), FN K(Y)
124Ø RETURN
125Ø REM -= Hintergrund =-
126Ø HGR2
127Ø HCOLOR= 1
128Ø FOR S = Ø TO 42
129Ø I = S * 4 + 11
13ØØ HPLOT I,11 TO I,179
131Ø HPLOT 11,I TO 179,I
132Ø NEXT
133Ø HCOLOR= 3
134Ø RETURN
135Ø REM -= Speicher löschen =-
136Ø REM 6Ø * 4Ø Felder
137Ø REM XØ-X49 - Punkte
138Ø REM X5Ø    - Startpunkt
139Ø REM X51    - Endpunkt
14ØØ REM X52    - Farbbit
141Ø CALL LOESCHEN: RETURN
142Ø REM -= Shapes =-
143Ø POKE 233,SHAPE / 256: POKE 232,SHAPE - PEEK (233) *
     256
144Ø PRINT
145Ø PRINT CHR$ (4);"BLOAD SPRITE.EDIT.ASS,A";ROUTINE
146Ø RETURN
147Ø REM -= Cursor =-
148Ø XDRAW 3 AT FN K(X), FN K(Y):N = 2Ø
149Ø IF N THEN N = N - 1: IF PEEK (KBD) < 128 THEN 149Ø
15ØØ XDRAW 3 AT FN K(X), FN K(Y):N = 2Ø
151Ø IF PEEK (KBD) > 127 THEN GET A$:A = ASC (A$): RETURN
152Ø N = N - 1: IF N THEN 151Ø
153Ø GOTO 148Ø
154Ø REM -= EOR bei X,Y =-
155Ø POKE FN A(Ø),1 - PEEK ( FN A(Ø))
156Ø GOSUB 117Ø
157Ø RETURN
158Ø REM -= Verschieben 'M' =-
159Ø FOR X = Ø TO A: FOR Y = B TO 1 STEP - 1
16ØØ POKE FN A(Ø), PEEK ( FN A(Ø) - 6Ø): NEXT : POKE FN
     A(Ø),Ø: NEXT : RETURN
161Ø REM -= Verschieben 'I' =-
162Ø FOR X = Ø TO A: FOR Y = Ø TO B - 1
163Ø POKE FN A(Ø), PEEK ( FN A(Ø) + 6Ø): NEXT : POKE FN
     A(Ø),Ø: NEXT : RETURN
164Ø REM -= Verschieben 'K' =-
165Ø FOR Y = Ø TO B: FOR X = A TO 1 STEP - 1
166Ø POKE FN A(Ø), PEEK ( FN A(Ø) - 1): NEXT : POKE FN
     A(Ø),Ø: NEXT : RETURN
167Ø REM -= Verschieben 'J' =-
168Ø FOR Y = Ø TO B: FOR X = Ø TO A - 1
169Ø POKE FN A(Ø), PEEK ( FN A(Ø) + 1): NEXT : POKE FN
     A(Ø),Ø: NEXT : RETURN
17ØØ REM -= Zeile plotten =-
171Ø A = X: REM X speichern
172Ø CALL ROUTINE:W1 = PEEK (SPUNKT + 6Ø * Y):W2 = PEEK
     (EPUNKT + 6Ø * Y)
173Ø IF W1 = 99 THEN RETURN
174Ø FOR X = W1 TO W2 + 1
175Ø GOSUB 117Ø
176Ø NEXT
177Ø X = A: RETURN
178Ø REM -= Sprite komplett plotten =-
179Ø CALL ROUTINE
18ØØ FOR Y = Ø TO 41:W1 = PEEK (SPUNKT + 6Ø * Y):W2 = PEEK
     (EPUNKT + 6Ø * Y)
181Ø IF W1 = 99 THEN 183Ø
182Ø FOR X = W1 TO W2 + 1: GOSUB 117Ø: NEXT
183Ø IF PEEK ( FN F(Ø)) THEN HCOLOR= 3: DRAW 3 AT 5, FN
     K(Y)
184Ø NEXT : RETURN
185Ø REM -= Eingabe =-
186Ø GET E$
187Ø IF E$ = CHR$ (13) THEN PRINT : RETURN
188Ø IF E$ = CHR$ (8) THEN IF LEN (X$) > Ø THEN 192Ø
189Ø IF E$ < " " THEN 186Ø
19ØØ IF LEN (X$) > = MAX THEN PRINT CHR$ (7);: GOTO 186Ø
191Ø PRINT E$;:X$ = X$ + E$: GOTO 186Ø
192Ø IF LEN (X$) = 1 THEN X$ = "": GOTO 194Ø
193Ø X$ = LEFT$ (X$, LEN (X$) - 1)
194Ø PRINT E$;" ";E$;: GOTO 186Ø
195Ø REM -= Länge/ Breite =-
196Ø Y = 42: CALL ROUTINE
197Ø Y = Y - 1: IF Y > = Ø THEN IF PEEK (SPUNKT + 6Ø * Y) =
     99 THEN 197Ø
198Ø X = Ø: FOR B = Ø TO Y
199Ø IF PEEK (SPUNKT + 6Ø * B) < > 99 THEN W = PEEK (EPUNKT
     + 6Ø * B): IF X < W THEN X = W
2ØØØ NEXT : RETURN
2Ø1Ø REM -= BITSPRITE =-
2Ø2Ø PRINT L$; STR$ (B + 1);T$
2Ø3Ø PRINT B$; INT ((B + X) / 7 + 1);",";Y + 1
2Ø4Ø FOR W1 = Ø TO Y
2Ø5Ø PRINT B$;
2Ø6Ø I = - 1:A$ = "":BY = Ø
2Ø7Ø FOR W2 = Ø - B TO X
2Ø8Ø I = I + 1: IF I = 7 THEN I = Ø:BY = BY + PEEK (FARBBIT
     + 6Ø * W1) * 128: PRINT A$; STR$ (BY);:A$ = ",":BY = Ø
2Ø9Ø W = Ø: IF W2 > = Ø THEN W = PEEK (START + 6Ø * W1 +
     W2)
21ØØ BY = BY + 2 ↑ I * W
211Ø NEXT
212Ø BY = BY + PEEK (FARBBIT + 6Ø * W1) * 128: PRINT A$;
     STR$ (BY)
213Ø NEXT
214Ø PRINT R$: RETURN
215Ø REM -= Help =-
216Ø HOME
217Ø INVERSE : HTAB 1Ø: PRINT "EDITOR-KOMMANDOS:": NORMAL
218Ø PRINT : PRINT "Editor verlassen          -   <ESC>"
219Ø VTAB 5: HTAB 31: PRINT "I": VTAB 7: HTAB 31: PRINT "M"
22ØØ VTAB 6: PRINT "Zeichen-Cursor bewegen  -   J + K"
221Ø VTAB 9: PRINT "Punkte setzen/löschen   -  <CTRL> IJKM"
222Ø PRINT : PRINT "Sprite löschen          -     L"
223Ø PRINT : PRINT "Farbbit setzen/löschen  -     F"
224Ø PRINT : PRINT "Sprite verschieben      -   V + IJKM"
225Ø TEXT
226Ø VTAB 23: HTAB 7: INVERSE : PRINT "< BITTE TASTE
     DRÜCKEN >";: GET A$: PRINT : NORMAL
227Ø POKE - 16297,Ø: POKE - 163Ø4,Ø: POKE - 16299,Ø: RETURN
228Ø REM -= Initalisierung =-
229Ø ROUTINE = 25ØØØ:LOESCHEN = ROUTINE + 68:SHAPE =
     ROUTINE + 11Ø
23ØØ HOME :START = 26ØØØ:SPUNKT = 5Ø + START:EPUNKT = 51 +
     START:FARBBIT = 52 + START:KBD = - 16384
231Ø IF PEEK (768) < > 123 THEN POKE 769,Ø: GOSUB 142Ø:
     GOSUB 135Ø:SF% = 1: POKE 768,123: REM Speicher löschen
232Ø RETURN
233Ø REM -= Editor =-
234Ø IF SF% > Ø THEN GOSUB 125Ø
235Ø IF SF% = 2 THEN GOSUB 178Ø
236Ø POKE - 163Ø2,Ø: POKE - 16297,Ø: POKE - 163Ø4,Ø: POKE -
     16299,Ø
237Ø SF% = Ø:X = Ø:Y = Ø
238Ø GOSUB 147Ø: REM Cursor
239Ø IF A = ASC ("I") - 64 THEN GOSUB 154Ø:A$ = "I"
24ØØ IF A = ASC ("K") - 64 THEN GOSUB 154Ø:A$ = "K"
241Ø IF A = ASC ("J") - 64 THEN GOSUB 154Ø:A$ = "J"
242Ø IF A = ASC ("M") - 64 THEN GOSUB 154Ø:A$ = "M"
243Ø IF A$ = "I" THEN Y = Y - 1: IF Y < Ø THEN Y = 41
244Ø IF A$ = "M" THEN Y = Y + 1: IF Y > 41 THEN Y = Ø
245Ø IF A$ = "K" THEN X = X + 1: IF X > 41 THEN X = Ø
246Ø IF A$ = "J" THEN X = X - 1: IF X < Ø THEN X = 41
247Ø IF A$ = "F" THEN GOTO 253Ø
248Ø IF A$ = "L" THEN 257Ø
249Ø IF A$ = "V" THEN 265Ø
25ØØ IF A$ = CHR$ (27) THEN RETURN
251Ø IF A$ = "?" THEN GOSUB 215Ø: GOTO 238Ø
252Ø GOTO 238Ø
253Ø F = PEEK ( FN F(Ø)):F = 1 - F
254Ø HCOLOR= 3 * F: DRAW 3 AT 5, FN K(Y)
255Ø POKE FN F(Ø),F
256Ø GOSUB 17ØØ: GOTO 238Ø
257Ø REM Sprite löschen
258Ø HOME : POKE - 163ØØ,Ø: POKE - 163Ø3,Ø
259Ø VTAB 1Ø: PRINT "Sprite löschen (J/N)? ";: GET A$:
     PRINT A$
26ØØ IF A$ < > "J" THEN POKE - 163Ø4,Ø: POKE - 16299,Ø:
     GOTO 238Ø
261Ø A = X:B = Y
262Ø GOSUB 135Ø: GOSUB 125Ø
263Ø X = A:Y = B
264Ø GOTO 238Ø
265Ø REM Sprite verschieben
266Ø HOME : POKE - 163ØØ,Ø: POKE - 163Ø3,Ø: GOSUB 195Ø:A =
     X + 1:B = Y + 1
267Ø VTAB 1Ø: PRINT "Verschieben - welche Richtung?"
268Ø VTAB 12: HTAB 15: PRINT "I": HTAB 13: PRINT "J   K   -
     Richtung": HTAB 15: PRINT "M"
269Ø PRINT : PRINT "andere Eingabe -> zurück": VTAB 1Ø:
     HTAB 33: PRINT ;: GET A$: PRINT A$
27ØØ IF A$ = "M" THEN GOSUB 158Ø: GOTO 275Ø
271Ø IF A$ = "I" THEN GOSUB 161Ø: GOTO 275Ø
272Ø IF A$ = "K" THEN GOSUB 164Ø: GOTO 275Ø
273Ø IF A$ = "J" THEN GOSUB 167Ø: GOTO 275Ø
274Ø POKE - 163Ø4,Ø: POKE - 16299,Ø: GOTO 238Ø
275Ø POKE - 163Ø4,Ø: POKE - 16299,Ø
276Ø GOSUB 125Ø: GOSUB 178Ø
277Ø X = A:Y = B: GOTO 238Ø
278Ø REM -= BSPRITE-Berechnung =-
279Ø HOME : INVERSE : HTAB 9: PRINT "BSPRITE-BERECHNUNG":
     NORMAL
28ØØ PRINT : PRINT "1 -> Normal"
281Ø PRINT : PRINT "2 -> Normal (alle Farben)"
282Ø PRINT : PRINT "3 -> Einzel-Sprite"
```

```
2830 PRINT : PRINT "Nr.:";: GET A$:A = VAL (A$): HTAB 1:
     PRINT "    "
2840 IF A > 3 OR A < 1 THEN PRINT CHR$ (7): RETURN
2850 INVERSE : VTAB 2 * A + 1: PRINT STR$ (A): NORMAL
2860 POKE 34,11: HOME
2870 X$ = "T.":MAX = 33: PRINT "Filename: T.";: GOSUB
     1850:F$ = X$: VTAB 9: PRINT "File: ";F$
2880 IF F$ = "" THEN PRINT CHR$ (7): RETURN
2890 HOME : PRINT "Labelname (max 5 Zeichen): ";:MAX = 5:X$
     = "": GOSUB 1850
2900 IF X$ = "" THEN PRINT CHR$ (7): RETURN
2910 L$ = X$: VTAB 10: PRINT "Label: ";L$: HOME
2920 IF A = 3 THEN 3120
2930 GOSUB 1950
2940 IF PEEK (KBD) = 171 THEN PRINT CHR$ (7):X = X + 1
2950 VTAB 11: PRINT "Sprite: ";X;" * ";Y
2960 PRINT CHR$ (4);"MONCIO"
2970 HOME : PRINT CHR$ (4);"OPEN ";F$
2980 PRINT CHR$ (4);"WRITE ";F$
2990 PRINT I$
3000 PRINT L$;T$
3010 FOR B = 1 TO 7: PRINT AD$;L$; STR$ (B): NEXT
3020 IF A = 1 THEN 3050
3030 PRINT L$;"S";T$
3040 FOR B = 2 TO 8: PRINT AD$;L$; STR$ (B): NEXT
3050 PRINT R$
3060 FOR B = 0 TO A + 5
3070 GOSUB 2010
3080 NEXT
3090 PRINT CHR$ (4);"CLOSE ";F$
3100 PRINT CHR$ (4);"NOMONCIO"
3110 RETURN
3120 REM Einzel-Sprite
3130 HOME
3140 PRINT "Begrenzung festlegen -"
3150 PRINT "Cursor bewegen mit I J K M"
3160 PRINT "Mit <ESC> beenden"
3170 PRINT : INVERSE : PRINT "<RETURN> DRÜCKEN": NORMAL
3180 GET A$: IF A$ < > CHR$ (13) THEN 3180
3190 PRINT : POKE - 16302,0: POKE - 16297,0: POKE -
     16304,0: POKE - 16299,0
3200 GOSUB 1950
3210 HCOLOR= 1: HPLOT 0, FN K(Y) + 2 TO 4, FN K(Y) + 2:
     HPLOT FN K(X) + 2,0 TO FN K(X) + 2,4
3220 GOSUB 1470
3230 HCOLOR= 0: HPLOT 0, FN K(Y) + 2 TO 4, FN K(Y) + 2:
     HPLOT FN K(X) + 2,0 TO FN K(X) + 2,4
3240 IF A$ = "I" THEN Y = Y - 1: IF Y < 0 THEN Y = 41
3250 IF A$ = "M" THEN Y = Y + 1: IF Y > 41 THEN Y = 0
3260 IF A$ = "K" THEN X = X + 1: IF X > 41 THEN X = 0
3270 IF A$ = "J" THEN X = X - 1: IF X < 0 THEN X = 41
3280 IF A$ < > CHR$ (27) THEN 3210
3290 TEXT : POKE 34,11: HOME : VTAB 11: PRINT "Sprite:
     ";X;" * ";Y: HOME
3300 INPUT "Ab welchem Bit (1-7) ?";A$
3310 B = VAL (A$): IF B < 1 OR B > 7 THEN PRINT CHR$ (7);:
     RETURN
3320 VTAB 11: HTAB 20: PRINT "Bit :";B: HOME
3330 PRINT CHR$ (4);"MONCIO"
3340 PRINT CHR$ (4);"OPEN ";F$
3350 PRINT CHR$ (4);"WRITE ";F$
3360 PRINT L$;T$
3370 FOR W = 0 TO 6: PRINT AD$;L$; STR$ (B): NEXT
3380 B = B - 1
3390 GOSUB 2010
3400 PRINT CHR$ (4);"CLOSE ";F$
3410 PRINT CHR$ (4);"NOMONCIO"
3420 RETURN
3430 REM -= Sprite laden/sichern =-
3440 HOME
3450 INVERSE : HTAB 4: PRINT "  SPRITE LADEN/SICHERN  ":
     NORMAL
3460 VTAB 3: PRINT "L)aden oder S)ichern ";
3470 INVERSE
3480 GET A$: IF A$ = "L" THEN HTAB 1: PRINT "L)ADEN": GOTO
     3510
3490 IF A$ < > "S" THEN PRINT CHR$ (7): RETURN
3500 HTAB 13: PRINT "S)ICHERN"
3510 NORMAL : POKE 34,5
3520 HOME : PRINT "? - CATALOG": PRINT
3530 MAX = 33:X$ = "SPR.": PRINT "Filename: SPR.";
3540 GOSUB 1850
3550 IF X$ = "?" OR X$ = "SPR.?" THEN PRINT CHR$
     (4);"CATALOG": GET X$: GOTO 3520
3560 IF A$ = "S" THEN 3610
3570 REM Laden
3580 CALL LOESCHEN:SF% = 1
3590 PRINT CHR$ (4),"BLOAD";X$;",A";START
3600 SF% = 2: RETURN
3610 REM Sichern
3620 CALL ROUTINE:Y = 42
3630 Y = Y - 1: IF Y > = 0 THEN IF PEEK (SPUNKT + 60 * Y) =
     99 THEN 3630
3640 Y = Y + 1: IF Y = 0 THEN RETURN
3650 PRINT CHR$ (4);"BSAVE";X$;",A";START;",L";Y * 60
3660 RETURN
3670 REM -= Hauptprogramm =-
3680 SF% = PEEK (769)
3690 ONERR GOTO 3830
3700 GOSUB 2280
3710 REM -= Menü =-
3720 TEXT : HOME
3730 HTAB 15: INVERSE : PRINT "SPRITE-EDITOR": NORMAL :
     VTAB 5
3740 PRINT "1 -> Sprite editieren (?-Help)": PRINT
3750 PRINT "2 -> BSPRITE berechnen": PRINT
3760 PRINT "3 -> Sprite laden/sichern": PRINT
3770 PRINT "4 -> Programmende": PRINT
3780 VTAB 15: PRINT "Nr.:";
3790 GET A$:A = VAL (A$): PRINT A$
3800 IF A = 4 THEN POKE 769,SF%: HOME : END
3810 ON A GOSUB 2330,2780,3430
3820 GOTO 3710
3830 REM -= Error =-
3840 POKE 769,SF%: TEXT
3850 PRINT PEEK (222); CHR$ (7);" ERROR IN "; PEEK (218) +
     PEEK (219) * 256
3860 END
```

SPRITE.EDIT.ASS

```
BSAVE SPRITE.EDIT.ASS, A$2000, L$00A6

2000:A9 90 85 19 A9 65 85 1A
2008:A9 2A 85 07 A0 32 A9 63
2010:91 19 A0 00 B1 19 D0 17
2018:C8 C0 32 D0 F7 18 A5 19
2020:69 3C 85 19 A5 1A 69 00
2028:85 1A C6 07 D0 DE 60 98
2030:AA A0 32 B1 19 C9 63 D0
2038:03 8A 91 19 8A C8 91 19
2040:A8 18 90 D4 A9 90 85 19
2048:A9 65 85 1A A9 2A 85 07
2050:A2 00 A0 00 8A 91 19 C8
2058:C0 60 D0 F9 18 A5 19 69
2060:3C 85 19 A5 1A 69 00 85
2068:1A C6 07 D0 E5 60 04 00
2070:0A 00 19 00 22 00 2A 00
2078:2D 2D 36 36 3F 3F 24 2C
2080:2D 36 3F 2C 05 43 00 0D
2088:0D 36 36 1F 1F 24 24 00
2090:89 36 05 43 1F 04 43 00
2098:11 2D 36 3F 2C 05 43 00
20A0:00 00 00 00 00 00
```

Kurzanleitung

Das Sprite in dem Programm ASTA.DEMO kann man folgendermaßen selbst erstellen:

1) SPRITE.EDIT eingeben und mit
   SAVE SPRITE.EDIT
   abspeichern.
2) SPRITE.EDIT.ASS abtippen und mit
   BSAVE SPRITE.EDIT.ASS, A$2000, L$00A6
   abspeichern.
3) Mit dem Sprite-Editor ein eigenes Sprite erstellen
   und mit der Option "Bitsprite berechnen" als normales
   Sprite unter dem File- und Labelnamen "SPRTE"
   abspeichern.
4) ASTA.DEMO als Assembler-Quelltext bis Zeile 93
   eingeben und danach mit dem R-Kommando (Merlin-
   Assembler) das eigene Sprite einladen.
   (Quelltext als T.ASTA.DEMO oder ASTA.DEMO.S speichern.)
5) Nach dem Assemblieren: Objektcode unter dem Namen
   ASTA.DEMO abspeichern.
6) Aufruf durch
   BLOAD ASTA
   BRUN ASTA.DEMO

ASTA.DEMO

```
                        1   *-------------------------------------*
                        2   *   Beispiel zur Anwendung von ASTA   *
                        3   *                                     *
                        4   * Abspeichern mit                     *
                        5   * BSAVE ASTA.DEMO, A$0800, L$00C8      *
                        6   * Aufrufen mit                        *
                        7   * BLOAD ASTA, dann BRUN ASTA.DEMO     *
                        8   * Abbrechen mit Reset                 *
                        9   *                                     *
                        10  * von H. Grumser                      *
                        11  *-------------------------------------*
                        12
                        13          ORG  $800
                        14
                        15  HPAG    EQU  $E6
                        16  HGR2    EQU  $F3D8
                        17  HGR     EQU  $F3E2
                        18  BKGND1  EQU  $F3F8
                        19
                        20  * ASTA-Routinen und -Tabellen
                        21
                        22  SPRTD   EQU  $8000
                        23  XWRTD   EQU  SPRTD+30
                        24  YWRTD   EQU  XWRTD+90
                        25  YMATD   EQU  YWRTD+30
                        26  STAB    EQU  $8300
                        27  POSINIT EQU  $85BB
                        28  ASTA    EQU  $8400
                        29
0800: 20 BB 85          30          JSR  POSINIT    ;Tab. löschen
0803: 20 29 08          31          JSR  INIT       ;Initialisierung
                        32
0806: AD 54 C0          33  LOOP    LDA  $C054      ;HGR2 zeichnen
0809: A9 40             34          LDA  #$40       ;HGR1 anzeigen
080B: 85 E6             35          STA  HPAG
080D: 20 59 08          36          JSR  PROGRAMM
0810: 20 00 84          37          JSR  ASTA       ;Sprites zeichnen
0813: 20 71 08          38          JSR  GRAFIK     ;Bild ändern
                        39
0816: AD 55 C0          40          LDA  $C055      ;HGR1 anzeigen
0819: A9 20             41          LDA  #$20       ;HGR2 zeichnen
081B: 85 E6             42          STA  HPAG
081D: 20 59 08          43          JSR  PROGRAMM
0820: 20 00 84          44          JSR  ASTA       ;Sprites zeichnen
0823: 20 71 08          45          JSR  GRAFIK     ;Bild ändern
                        46
0826: 18                47          CLC             ;Endlos-
0827: 90 DD             48          BCC  LOOP       ; schleife
                        49
                        50  * Unterprogramme
                        51
0829: 20 E2 F3          52  INIT    JSR  HGR        ;HGR1 löschen
082C: 20 D8 F3          53          JSR  HGR2       ;HGR2 löschen
082F: A9 60             54          LDA  #$60       ;Hintergrund
0831: 20 F8 F3          55          JSR  BKGND1     ; löschen
0834: A9 75             56          LDA  #<SPRTE    ;Sprite-Zeiger
0836: 8D 02 83          57          STA  STAB+2     ; in Sprite-
0839: A9 08             58          LDA  #>SPRTE    ; tabelle
083B: 8D 03 83          59          STA  STAB+3     ; eintragen
083E: A0 1D             60          LDY  #29
0840: A9 01             61  INTLOOP LDA  #1         ;Sprite-Nummer
0842: 99 00 80          62          STA  SPRTD,Y    ; eintragen
0845: A9 BF             63          LDA  #191       ;Unterkante
0847: 99 96 80          64          STA  YMATD,Y    ; eintragen
084A: 98                65          TYA
084B: 0A                66          ASL             ;Y-Koord. =
084C: 0A                67          ASL             ; 4 * Y-Reg.
084D: 99 78 80          68          STA  YWRTD,Y
0850: 29 1F             69          AND  #%00011111 ;X-Koord. =
0852: 99 1E 80          70          STA  XWRTD,Y    ; 4 * Y-Reg.
0855: 88                71          DEY             ; MOD 16
0856: 10 E8             72          BPL  INTLOOP
0858: 60                73          RTS
                        74
0859: A0 1D             75  PROGRAMM LDY  #29
085B: 38                76  PRGMLOOP SEC
085C: B9 35 D1          77          LDA  $D135,Y    ;"Zufallszahl"
085F: 29 03             78          AND  #%00000011 ; MOD 4
0861: 79 1E 80          79          ADC  XWRTD,Y    ; addieren
0864: C9 8C             80          CMP  #140       ;Rechter Rand?
0866: 90 02             81          BCC  PRGRM1
0868: A9 00             82          LDA  #$00       ;ja, von vorn
086A: 99 1E 80          83  PRGRM1  STA  XWRTD,Y    ;neue Koord.
086D: 88                84          DEY
086E: 10 EB             85          BPL  PRGMLOOP
0870: 60                86          RTS
                        87
0871: 2C 30 C0          88  GRAFIK  BIT  $C030      ;keine GRAFIK
0874: 60                89          RTS
                        90
                        91  * Folgende Tabelle wurde mit
                        92  * SPRITE.EDIT erstellt:
                        93  *
                        94  *-- SPRITEDEF --
0875: 83 08             95  SPRTE   DA   SPRTE1
0877: 8A 08             96          DA   SPRTE2
0879: 91 08             97          DA   SPRTE3
087B: 98 08             98          DA   SPRTE4
087D: A4 08             99          DA   SPRTE5
087F: B0 08            100          DA   SPRTE6
0881: BC 08            101          DA   SPRTE7
0883: 01 05            102  SPRTE1  DFB  1,5
0885: 0E               103          DFB  14
0886: 1F               104          DFB  31
0887: 1F               105          DFB  31
0888: 1F               106          DFB  31
0889: 0E               107          DFB  14
088A: 01 05            108  SPRTE2  DFB  1,5
088C: 1C               109          DFB  28
088D: 3E               110          DFB  62
088E: 3E               111          DFB  62
088F: 3E               112          DFB  62
0890: 1C               113          DFB  28
0891: 01 05            114  SPRTE3  DFB  1,5
0893: 38               115          DFB  56
0894: 7C               116          DFB  124
0895: 7C               117          DFB  124
0896: 7C               118          DFB  124
0897: 38               119          DFB  56
0898: 02 05            120  SPRTE4  DFB  2,5
089A: 70 00            121          DFB  112,0
089C: 78 01            122          DFB  120,1
089E: 78 01            123          DFB  120,1
08A0: 78 01            124          DFB  120,1
08A2: 70 00            125          DFB  112,0
08A4: 02 05            126  SPRTE5  DFB  2,5
08A6: 60 01            127          DFB  96,1
08A8: 70 03            128          DFB  112,3
08AA: 70 03            129          DFB  112,3
08AC: 70 03            130          DFB  112,3
08AE: 60 01            131          DFB  96,1
08B0: 02 05            132  SPRTE6  DFB  2,5
08B2: 40 03            133          DFB  64,3
08B4: 60 07            134          DFB  96,7
08B6: 60 07            135          DFB  96,7
08B8: 60 07            136          DFB  96,7
08BA: 40 03            137          DFB  64,3
08BC: 02 05            138  SPRTE7  DFB  2,5
08BE: 00 07            139          DFB  0,7
08C0: 40 0F            140          DFB  64,15
08C2: 40 0F            141          DFB  64,15
08C4: 40 0F            142          DFB  64,15
08C6: 00 07            143          DFB  0,7

200 Bytes
```

Hinweis: Die Peeker-Sammeldisk # 11 enthält zusätzlich ein komplettes Demo, das mit RUN ASTA.DEMO.1 gestartet werden kann und seinerseits die verschiedenen Binärfiles AD1 bis AD5 automatisch einlädt.

Großformatiger HGR-Ausdruck

von Mark Liebrand

Die hier vorgestellte Routine ermöglicht es, mit Hilfe eines Druckers Hardcopies vom HGR-Bildschirm in der Größe von Postern oder Plakaten anzufertigen. Im Extremfall können diese Ausdrucke eine Fläche von fast 30 qm einnehmen.

Eine „Posterroutine" ist sehr nützlich, wenn man Plakate erstellen will (z.B. von besonders guten Grafiken aus Spielen) oder wenn man mit einem Malprogramm ein Kunstwerk erstellt hat und sich dieses groß aufhängen möchte. Orientiert man sich am Imagewriter, so kann ein Ausdruck von 40facher Vergrößerung durchaus die Maße 7,12 m x 4,32 m annehmen. Es sind allerdings auch Zimmergrößen erreichbar, z.B. 1,43 m x 1,08 m bei Faktor 8 oder 0,71 m x 0,54 m bei Faktor 4. Wem dies immer noch zu groß ist, der sollte es einmal mit Faktor 2 versuchen, es wird ein „Pösterchen" von 35,6 cm x 27 cm ausgegeben.

Arbeitsweise des Programms

Dieses Programm läuft auf einem Apple II mit mindestens 48K, Diskettenlaufwerk und grafikfähigem Drucker. Man kann verschiedene Vergrößerungsfaktoren eingeben: 2, 4, 8, 20 und 40. Die Faktoren geben an, um das Wievielfache sich die Kantenlängen des Bildes steigern. Also wird ein Bild bei Faktor 4 viermal so breit und viermal so hoch.
Um dies zu erreichen, muß das zu vergrößernde Bild in die erste Grafikseite geladen werden. Darauf erfolgt ein Sprung in eine Maschinenroutine. Sie unterteilt das Originalbild in – den Faktoren entsprechend – kleine Teilbilder, die nacheinander vergrößert in die zweite Grafikseite kopiert werden. Ist ein Teilbild kopiert, erfolgt ein Rücksprung nach Applesoft. Nun kann eine Hardcopy erstellt werden. Durch den Rücksprung nach Applesoft erreicht man, daß diese Routine kompatibel zu allen Druckern ist, die überhaupt grafikfähig sind. Denn im Applesoft-Programm kann man die Codes, die man für die Grafikausgabe braucht, bequem einfügen. Diese Steuerzeichen müssen sich auf die zweite Grafikseite beziehen.
Ist die Hardcopy ausgegeben, erfolgt wieder ein Sprung in das Maschinenprogramm. Hier wird wieder ein Teilbild erstellt, dann erfolgt der Rücksprung usw., bis das ganze Originalbild abgetastet ist.
Wenn alles fertig ist, braucht man nur noch die gedruckten Bilder aneinanderzukleben. Um dabei nicht allzu lange puzzlen zu müssen, sollte man wissen, daß das Originalbild in Zeilen von links oben nach rechts unten abgetastet wird und genauso die Ausgabe erfolgt.
Wer eine 20- oder 40fache Vergrößerung macht, wird merken, daß das Bild etwas verzerrt wird. Das Verhältnis von Höhe und Breite ist nicht mehr dassebe wie beim Original. Dies ist dadurch zu erklären, daß die X- und Y-Achse im Grafikbildschirm nicht dieselben Teiler haben und so ein Kompromiß auf Kosten der Y-Achse eingegangen wurde, um den Programmieraufwand in Grenzen zu halten.

Eingeben des Programms

Beim Abtippen beginnt man zweckmäßigerweise mit dem Assemblerlisting **PLAKAT**. Der Objektcode sollte dann mit „BSAVE PLAKAT, A$6001, L$01D4" gespeichert werden. Als nächstes kommt das Applesoft-Programm **PLAKAT.INSTALL** an die Reihe. Es lädt den eben erzeugten Objectcode ein und fügt eine Tabelle hinzu, in der sortiert die Speicheradressen aller Zeilen der Grafikseiten stehen. Nach Anlage der Tabelle speichert das Programm den erweiterten Objektcode wieder. Die Ausführung des INSTALL-Programms dauert ungefähr eine Minute. (Auf der Peeker-Sammeldiskette ist das Programm PLAKAT bereits installiert, das INSTALL-Programm befindet sich nur der Vollständigkeit halber ebenfalls auf der Diskette.)
Nachdem nun das Gröbste geschafft ist, kann man das zweite Applesoft-Programm **PLAKAT.DEMO** eingeben. Dieses Programm soll lediglich zeigen, wie die Maschinenroutine zu bedienen ist. Es ist auch denkbar, es in ähnlicher Form in andere Programme einzubauen. Allerdings sollte man darauf achten, daß die Variablen des Programms nicht das Maschinenprogramm oder die Grafikseiten berühren, sonst können nämlich seltsame Dinge passieren. Gegebenenfalls ist HIMEM auf 8192 ($2000) zu setzen, um Konflikte zu vermeiden.

Aufruf des Programms PLAKAT

Beim Aufruf der Maschinenroutine kommt es darauf an, daß sich das zu vergrößernde Bild in der ersten Grafikseite befindet. Wenn das Bild von Diskette geladen werden soll, erreicht man dies auf jeden Fall mit „BLOAD Bildname, A$2000".
Außerdem muß der Vergrößerungsfaktor in Speicherstelle 255 gepokt werden. Dann sollte mit **CALL 24577** der erste Aufruf (Kaltstart) der Routine erfolgen. Er veranlaßt, daß die Zeiger und Variablen des Programms initialisiert werden. Direkt nach dem Kaltstart wird das erste Teilbild auf Grafikseite 2 ausgegeben oder es erfolgen zwei Piepstöne, die signalisieren, daß der Vergrößerungsfaktor falsch gewählt wurde. Damit das auch im Programm erkannt wird, muß nach diesem CALL-Befehl eine Anweisung folgen, die das „Statusflag" (Speicherstelle 254) auf den Wert Null hin überprüft. In diesem Fall sollte das Programm abgebrochen werden oder eine Fehlermeldung erfolgen.
Nachdem der Kaltstart erfolgreich durchgeführt wurde, sollten nun die Befehle er-

folgen, die den Drucker zur Ausgabe der zweiten Grafikseite veranlassen.
Danach kann nun durch wiederholten Aufruf des Warmstarts mittels **CALL 24663** ein Bild nach dem anderen ausgegeben werden, bis der Status den Wert Null einnimmt, d.h. alle Teilbilder wurden abgearbeitet.
Das Programm PLAKAT.DEMO erfüllt die oben beschriebenen Anforderungen und kann somit ohne Bedenken zum Testen benutzt werden.

Zum Schluß noch drei Hinweise:
– Wer über kein Grafik-Interface verfügt, kann die Teilbilder auch auf Diskette speichern (BSAVE Bildname, A$4000, L$2000) und den Ausdruck mit Hilfe des Programms SUPERDUMP aus Peeker, Heft 6/85 erledigen.
– Der Drucker bildet die Schwachstelle des ganze Systems. Erstens wird der Druckkopf bei Grafikausgabe ziemlich stark beansprucht: Deshalb sollte man bei einer größeren Zahl von Hardcopies die durchsichtige Kunststoffabdeckung, die Druckkopf und Papier schützt, wegen der stärkeren Wärmeentwicklung hochklappen.
Zweitens ist die Zeit zu beachten: Wenn der Drucker drei Minuten für eine Hardcopy braucht, dann benötigt er für eine 4fache Vergrößerung ungefähr 48 Minuten. Überträgt man die auf den Vergrößerungsfaktor 40, so werden 1280 Bilder gedruckt, was einer Druckzeit von ca. 2 1/2 Tagen nonstop entspricht. Allerdings macht man sicherlich nicht ständig solche Ausdrucke, so daß man sich darüber nicht den Kopf zerbrechen sollte.
– Wem das Format 7 m x 4 m noch nicht groß genug ist, der sollte bei 40facher Vergrößerung die Teilbilder nicht drucken, sondern auf ca. 45 Disketten (beidseitig) speichern lassen, um diese Bilder dann einzeln wieder um den Faktor 40 zu vergrößern. Das Ergebnis ist ein Superplakat mit den Maßen 284,8 m x 138,2 m, bestehend aus 1 638 400 Blättern und einer Fläche von beinahe 40 000 qm. Das wäre ein Fall fürs Guinness Buch der Rekorde.

Tabelle 1

Faktor	Maße (cm)	Blätter
2	35,6 x 27	4
4	71,2 x 54	16
8	142,2 x 108	64
20	356 x 216	320
40	712 x 432	1280

Auflistung der unterschiedlichen Druckformate.
Die Maßangaben beziehen sich auf die Größe einer normalen Hardcopy von 17,8 cm x 13,5 cm

Kurzanleitung

1. PLAKAT eingeben und mit "BSAVE PLAKAT, A$6001, L$01D4" abspeichern.
2. PLAKAT.INSTALL eingeben, Diskette mit PLAKAT einlegen und PLAKAT.INSTALL starten.
3. PLAKAT.DEMO eingeben und Zeile 70 durch eigene drucker-spezifische Hardcopy-Befehle ersetzen.

PLAKAT.DEMO

```
10 PRINT CHR$ (4)"BLOAD PLAKAT"
20 INPUT "Bildname ? ";FI$
30 PRINT CHR$ (4)"BLOAD"FI$",A$2000"
40 INPUT "Vergrößerung ?";V
50 HGR2 : POKE 255,V: CALL 24577
60 IF PEEK (254) = 0 THEN END : REM Faktor falsch
70 REM Hier die Hardcopy-Befehle des Druckers einfügen!
80 IF PEEK (254) = 0 THEN END
90 CALL 24663: GOTO 70
```

PLAKAT.INSTALL

```
100 PRINT CHR$ (4)"BLOAD PLAKAT"
110 T1 = 25088
120 T2 = T1 + 256
130 T3 = T2 + 256
140 T4 = T3 + 256
150 FOR A = 0 TO 2
160 FOR B = 0 TO 7
170 FOR C = 0 TO 7
180 D = C * 1024 + B * 128 + A * 40
190 H = INT ((D + 8192) / 256):L = (D + 8192) - H * 256
200 POKE T1,L: POKE T2,H
210 H = H + 32
220 POKE T3,L: POKE T4,H
230 T1 = T1 + 1:T2 = T2 + 1:T3 = T3 + 1:T4 = T4 + 1
240 NEXT C,B,A
250 PRINT
260 PRINT CHR$ (4)"BSAVE PLAKAT,A$6001,L$600"
```

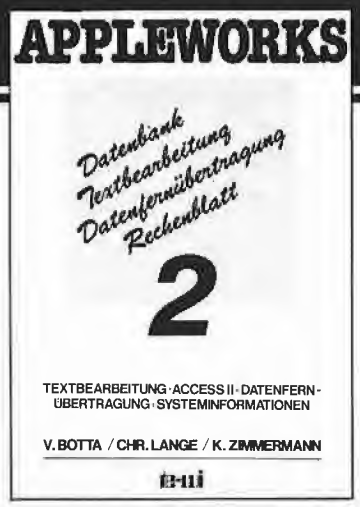
PLAKAT

```
                    1   ********************
                    2   *                  *
                    3   *     PLAKAT       *
                    4   *                  *
                    5   ********************
                    6   *
                    7   * Mark Liebrand, 1985
                    8   *
                    9   TAB      EQU  $F9
                    10  TAB1     EQU  $FB
                    11  MEM      EQU  $FD
                    12  STATUS   EQU  $FE
                    13  NUM      EQU  $FF
                    14  TAB1LZ   EQU  $50
                    15  TAB1HZ   EQU  $52
                    16  TAB2LZ   EQU  $54
                    17  TAB2HZ   EQU  $56
                    18  FAKZ     EQU  $58
                    19  Z1       EQU  $EB
                    20  Z3       EQU  $EC
                    21  Z4       EQU  $ED
                    22  ZW       EQU  $EE
                    23  ZW1      EQU  $EF
                    24  ZW2      EQU  $CE
                    25  BELL     EQU  $FF3A
                    26  *
                    27           ORG  $6001
                    28  *
6001: A2 05         29           LDX  #5          ;Prüfen
6003: BD 18 60      30  SCHLEI1  LDA  ZIFF,X      ;von
6006: C5 FF         31           CMP  NUM         ;NUM
6008: F0 26         32           BEQ  OK
600A: CA            33           DEX
600B: D0 F6         34           BNE  SCHLEI1
600D: 20 3A FF      35           JSR  BELL        ;ERROR,
6010: 20 3A FF      36           JSR  BELL        ;Faktor
6013: A9 00         37           LDA  #0          ;falsch
6015: 85 FE         38           STA  STATUS
6017: 60            39           RTS              ;Konstanten
6018: 00 02 04      40  ZIFF     HEX  000204081428.
601B: 08 14 28
601E: 00 02 04      41  ZIFF1    HEX  00020408102040
6021: 08 10 20
6024: 00 14 0A      42  ZIFF2    HEX  00140A05020 1
6027: 05 02 01
602A: 00 60 30      43  ZIFF3    HEX  006030180C06
602D: 18 0C 06
6030: A9 80         44  OK       LDA  #$80
6032: 85 FE         45           STA  STATUS
6034: BD 18 60      46           LDA  ZIFF,X      ;Konst.
6037: 8D C2 61      47           STA  FAK1        ;einlesen
603A: BD 1E 60      48           LDA  ZIFF1,X
603D: 8D C3 61      49           STA  FAK2
6040: BD 24 60      50           LDA  ZIFF2,X
6043: 8D C4 61      51           STA  FAK3
6046: 8D C5 61      52           STA  FAK4
6049: BD 2A 60      53           LDA  ZIFF3,X
604C: 8D C7 61      54           STA  FAK6
                    55  * Zeiger im Original setzen
```

```
6Ø4F: A9 ØØ     56            LDA  #Ø
6Ø51: 8D C8 61  57            STA  ZEIG1      ;X-Koord.
6Ø54: 8D C9 61  58            STA  ZEIG2      ;Y-Koord.
                59   * Zeiger im Teilbild setzen
6Ø57: A9 ØØ     6Ø   WSTART   LDA  #Ø         ;Warmstart
6Ø59: 8D CA 61  61            STA  ZEIG3      ;X-Koord.
6Ø5C: 8D CB 61  62            STA  ZEIG4      ;Y-Koord.
6Ø5F: AC C9 61  63   WEITER1  LDY  ZEIG2
6Ø62: AD DØ 61  64            LDA  TAB2L
6Ø65: 85 54     65            STA  TAB2LZ     ;Pointer
6Ø67: AD D1 61  66            LDA  TAB2L+1    ;in die
6Ø6A: 85 55     67            STA  TAB2LZ+1   ;Zeropage
6Ø6C: AD D2 61  68            LDA  TAB2H      ;legen
6Ø6F: 85 56     69            STA  TAB2HZ
6Ø71: AD D3 61  7Ø            LDA  TAB2H+1
6Ø74: 85 57     71            STA  TAB2HZ+1
6Ø76: AD CC 61  72            LDA  TAB1L
6Ø79: 85 5Ø     73            STA  TAB1LZ
6Ø7B: AD CD 61  74            LDA  TAB1L+1
6Ø7E: 85 51     75            STA  TAB1LZ+1
6Ø8Ø: AD CE 61  76            LDA  TAB1H
6Ø83: 85 52     77            STA  TAB1HZ
6Ø85: AD CF 61  78            LDA  TAB1H+1
6Ø88: 85 53     79            STA  TAB1HZ+1
6Ø8A: B1 5Ø     8Ø            LDA  (TAB1LZ),Y
6Ø8C: 85 F9     81            STA  TAB        ;Zeilen-
6Ø8E: B1 52     82            LDA  (TAB1HZ),Y ;adresse
6Ø9Ø: 85 FA     83            STA  TAB+1      ;(Original)
6Ø92: AC C8 61  84            LDY  ZEIG1
6Ø95: B1 F9     85            LDA  (TAB),Y
6Ø97: 85 FD     86            STA  MEM
6Ø99: 2Ø 5B 61  87            JSR  BITS       ;Bit-
6Ø9C: AD C3 61  88            LDA  FAK2       ;zerlegung
6Ø9F: 8D C6 61  89            STA  FAK5
6ØA2: AC CB 61  9Ø   SCHLEI3  LDY  ZEIG4
6ØA5: B1 54     91            LDA  (TAB2LZ),Y
6ØA7: 85 FB     92            STA  TAB1
6ØA9: B1 56     93            LDA  (TAB2HZ),Y
6ØAB: 85 FC     94            STA  TAB1+1
6ØAD: AD C2 61  95            LDA  FAK1
6ØBØ: 85 58     96            STA  FAKZ
6ØB2: A2 ØØ     97            LDX  #Ø
6ØB4: AC CA 61  98            LDY  ZEIG3      ;Ausgabe
6ØB7: BD D4 61  99   SCHLEI2  LDA  ARRAY,X    ;des
6ØBA: 91 FB     1ØØ           STA  (TAB1),Y   ;Arrays
6ØBC: E8        1Ø1           INX             ;auf
6ØBD: C8        1Ø2           INY             ;Bild-
6ØBE: C6 58     1Ø3           DEC  FAKZ       ;schirm
6ØCØ: DØ F5     1Ø4           BNE  SCHLEI2
6ØC2: EE CB 61  1Ø5           INC  ZEIG4
6ØC5: CE C6 61  1Ø6           DEC  FAK5
6ØC8: DØ D8     1Ø7           BNE  SCHLEI3
6ØCA: AD CB 61  1Ø8           LDA  ZEIG4
6ØCD: C9 CØ     1Ø9           CMP  #192       ;auf
6ØCF: DØ Ø4     11Ø           BNE  WEITER     ;Ende
6ØD1: CØ 28     111           CPY  #$28       ;prüfen
6ØD3: FØ 4C     112           BEQ  SCHLUSS
6ØD5: 38        113  WEITER   SEC
6ØD6: ED C3 61  114           SBC  FAK2       ;Zeiger
6ØD9: 8D CB 61  115           STA  ZEIG4      ;neu
6ØDC: AD CA 61  116           LDA  ZEIG3      ;setzen
6ØDF: 18        117           CLC
6ØEØ: 6D C2 61  118           ADC  FAK1
6ØE3: C9 28     119           CMP  #$28
6ØE5: FØ Ø9     12Ø           BEQ  NULL
6ØE7: 8D CA 61  121  BACK     STA  ZEIG3
6ØEA: 2Ø FF 6Ø  122           JSR  ADD
6ØED: 4C 5F 6Ø  123           JMP  WEITER1
6ØFØ: AD CB 61  124  NULL     LDA  ZEIG4
6ØF3: 18        125           CLC
6ØF4: 6D C3 61  126           ADC  FAK2
6ØF7: 8D CB 61  127           STA  ZEIG4
6ØFA: A9 ØØ     128           LDA  #Ø
6ØFC: 4C E7 6Ø  129           JMP  BACK
6ØFF: AE C8 61  13Ø  ADD      LDX  ZEIG1
61Ø2: E8        131           INX
61Ø3: CE C5 61  132           DEC  FAK4
61Ø6: FØ ØA     133           BEQ  ZEILE
61Ø8: 8E C8 61  134  BACK1    STX  ZEIG1
61ØB: 6Ø        135           RTS
61ØC: EE C9 61  136  PRUEF    INC  ZEIG2
61ØF: 4C Ø8 61  137           JMP  BACK1
6112: 8A        138  ZEILE    TXA
6113: 38        139           SEC
6114: ED C4 61  14Ø           SBC  FAK3
6117: AA        141           TAX
6118: AD C4 61  142           LDA  FAK3
611B: 8D C5 61  143           STA  FAK4
611E: 4C ØC 61  144           JMP  PRUEF
6121: 2Ø FF 6Ø  145  SCHLUSS  JSR  ADD
6124: A9 28     146           LDA  #$28
6126: 38        147           SEC
```

```
6127: ED C4 61  148           SBC  FAK3
612A: CD C8 61  149           CMP  ZEIG1
612D: FØ 15     15Ø           BEQ  NZ
612F: AD C9 61  151           LDA  ZEIG2
6132: 38        152           SEC
6133: ED C7 61  153           SBC  FAK6
6136: 8D C9 61  154           STA  ZEIG2
6139: AD C8 61  155           LDA  ZEIG1
613C: 18        156           CLC
613D: 6D C4 61  157           ADC  FAK3       ;zurück
614Ø: 8D C8 61  158           STA  ZEIG1      ;nach
6143: 6Ø        159           RTS             ;BASIC
6144: A9 ØØ     16Ø  NZ       LDA  #Ø
6146: 8D C8 61  161           STA  ZEIG1
6149: AD C9 61  162           LDA  ZEIG2
614C: C9 CØ     163           CMP  #192
614E: DØ ØA     164           BNE  BACK2
615Ø: 2Ø 3A FF  165           JSR  BELL
6153: 2Ø 3A FF  166           JSR  BELL
6156: A9 ØØ     167           LDA  #Ø
6158: 85 FE     168           STA  STATUS
615A: 6Ø        169  BACK2    RTS             ;fertig
615B: A2 28     17Ø  BITS     LDX  #4Ø
615D: A9 ØØ     171           LDA  #Ø
615F: 9D D4 61  172  SCHLEI7  STA  ARRAY,X    ;Array
6162: CA        173           DEX             ;löschen
6163: DØ FA     174           BNE  SCHLEI7
6165: 9D D4 61  175           STA  ARRAY,X
6168: 86 ED     176           STX  Z4         ;Byte-
616A: A2 Ø7     177           LDX  #7         ;zähler
616C: 86 CE     178           STX  ZW2
616E: 86 EC     179           STX  Z3
617Ø: 86 EB     18Ø           STX  Z1         ;Bit vom
6172: AD C2 61  181  SCHLEI4  LDA  FAK1       ;Bild
6175: 85 EF     182           STA  ZW1
6177: A5 FD     183           LDA  MEM
6179: ØA        184           ASL             ;Bit 7 raus
617A: ØA        185  SCHLEI41 ASL
617B: CA        186           DEX
617C: DØ FC     187           BNE  SCHLEI41
617E: BØ Ø5     188           BCS  CARRY
618Ø: A9 ØØ     189           LDA  #Ø
6182: 4C 87 61  19Ø           JMP  WEITER2
6185: A9 Ø1     191  CARRY    LDA  #1
6187: 85 EE     192  WEITER2  STA  ZW
6189: A4 EC     193           LDY  Z3
618B: A5 EE     194  SCHLEI6  LDA  ZW
618D: CØ Ø7     195  SCHLEI5  CPY  #7
618F: FØ Ø5     196           BEQ  NOROL
6191: ØA        197           ASL             ;schieben
6192: C8        198           INY
6193: 4C 8D 61  199           JMP  SCHLEI5
6196: A6 ED     2ØØ  NOROL    LDX  Z4
6198: 18        2Ø1           CLC
6199: 7D D4 61  2Ø2           ADC  ARRAY,X    ;addieren
619C: 9D D4 61  2Ø3           STA  ARRAY,X    ;speichern
619F: C6 EC     2Ø4           DEC  Z3
61A1: FØ ØE     2Ø5           BEQ  NEXTX
61A3: A4 EC     2Ø6  ZURUECK  LDY  Z3
61A5: C6 EF     2Ø7           DEC  ZW1
61A7: DØ E2     2Ø8           BNE  SCHLEI6
61A9: A6 EB     2Ø9           LDX  Z1
61AB: CA        21Ø           DEX
61AC: 86 EB     211           STX  Z1
61AE: DØ C2     212           BNE  SCHLEI4
61BØ: 6Ø        213  ENDE3    RTS             ;fertig
61B1: E6 ED     214  NEXTX    INC  Z4
61B3: A9 Ø7     215           LDA  #7
61B5: 85 EC     216           STA  Z3
61B7: A4 ED     217           LDY  Z4
61B9: 88        218           DEY
61BA: CC C2 61  219           CPY  FAK1
61BD: FØ F1     22Ø           BEQ  ENDE3
61BF: 4C A3 61  221           JMP  ZURUECK
61C2: ØØ        222  FAK1     HEX  ØØ         ;Programm-
61C3: ØØ        223  FAK2     HEX  ØØ         ;Variablen
61C4: ØØ        224  FAK3     HEX  ØØ
61C5: ØØ        225  FAK4     HEX  ØØ
61C6: ØØ        226  FAK5     HEX  ØØ
61C7: ØØ        227  FAK6     HEX  ØØ
61C8: ØØ        228  ZEIG1    HEX  ØØ
61C9: ØØ        229  ZEIG2    HEX  ØØ
61CA: ØØ        23Ø  ZEIG3    HEX  ØØ
61CB: ØØ        231  ZEIG4    HEX  ØØ
61CC: ØØ 62     232  TAB1L    HEX  ØØ62       ;Tabellen-
61CE: ØØ 63     233  TAB1H    HEX  ØØ63       ;zeiger
61DØ: ØØ 64     234  TAB2L    HEX  ØØ64
61D2: ØØ 65     235  TAB2H    HEX  ØØ65
61D4: ØØ        236  ARRAY    HEX  ØØ
```

468 Bytes



Band 1:
1. Einleitung
2. Was Sie benötigen
3. Starten von APPLE WORKS
4. Der Schreibtischmanager
5. Datenbank
6. Rechenblatt

A1 Anschluß der Festplatte ProFile
A2 APPLE II Easy Pieces Referenz
A3 Druckeranpassungen
A4 DOS 3,3 Konvertierungen
A5 APPLE WORKS Disketten sichern/kopieren
A6 Hilfsfunktionen nach Programmteilen

Band 2:
1. Einleitung
2. Was Sie benötigen
3. Starten von APPLE WORKS
4. Der Schreibtischmanager
5. Textbearbeitung
6. Datenfernübertragung

A1 ... A6 wie Band 1, dazu
A7 Modemkabel für APPLE IIe, IIc
A8 SuperSerialCard: Einstellung
A9 ASCII-Textdateien aus anderen Dateiformaten für ACCESS II
A10 DATEX-P20 F Verzeichnis
A11 Deutsche/Englische Menübilder von ACCESS II

APPLE WORKS auf APPLE II, IIe, IIc:
verwandelt APPLE-II-Computer in einen Elektronischen Schreibtischmanager mit:
Texterstellung ... Edition, Briefarchiv, Ausdruck etc.
Datenarchivierung ... Kontoführung, Buchhaltung etc. Formblattkalkulation ... Bilanzen, VisiCalc-Dateien etc. Datenfernübertragung ... Mailbox, Rechnerkopplung etc.

- ist ein erfolgreicheres Integrationspaket als LOTUS auf IBM PC!
- ist auf 1 MByte Speichererweiterungen Ihres APPLE II vorbereitet!
- erschließt Ihnen die Zukunftstechnik MAILBOX!
- ist ebenso einfach zu bedienen wie APPLE WRITER:
  Kein Befehlsstudium ... Einfachste Menüführung ... Sofortige Anwendbarkeit

te-wi's APPLE WORKS SYSTEMBÜCHER 1+2 zeigen Ihnen:
- Sämtliche APPLE WORKS Funktionen an Beispielen aus der Wirtschaft
- Das Wechseln zwischen Text/Rechenblatt/Datenarchiv/Dfü
- Umfassende Systeminformationen zu Dateikonvertierung, Druckeranpassung etc.

Von Botta/Lange/Zimmermann je 264 Seiten, Softcover, je DM 49,–

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

Weitere te-wi-Bücher

NEU

Das APPLE II/II+/IIe/IIc-Handbuch
(L. Poole)
Erst mit Hilfe dieses Leitfadens werden Sie Ihren Apple II erfolgreich einsetzen, denn Text und Bildmaterial gehen weit über das hinaus, was herstellerseitig an Literatur angeboten wird.
Neu überarbeitet und jetzt um die spezifischen Eigenheiten der Modelle **II e** und **II c** erweitert. 472 Seiten, Softcover, DM 66,–

LOGO – Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den U SA. Für die Computer APPLE II, C-64, IBM PC, ATARI bis 520 ST., TI-99 und Schneider CPCs.
Hochwertiges Textbuch für Logo-Kurse zu Hause und im Lehrbereich.
384 Seiten, A4, DM 59,–

NEU

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

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

NEU

Reparaturanleitung Computer: Apple II, IIplus
Einzigartige Serviceunterlage für Reparaturen und Entwicklungsarbeiten am Apple II. Enthält Schaltpläne, Bauteile- und Vergleichstypenliste; Prüfpunkte mit Oszillogrammen der Signalformen, Logiktabellen, Spannungsangaben; schnelle Servicetests; Anleitung zur systematischen Fehlersuche.
In A4-Mappe, DM 29,80

Erstes deutsches Referenzwerk sämtlicher Befehle und Systemroutinen von Apple II, IIplus, IIe
APPLE II PASCAL NEU
Betriebssystem, 272 S., DM 49,–
Sprache, 216 S., DM 39,–
Pascal 1,2 Addendum, 112 S., DM 36,–

Grundlagenbuch, Bestseller APPLE II PASCAL.
Eine praktische Anleitung.
544 S., DM 59,–

tm 4422

Noch im Programm:
Computer für Kinder, APPLE II, DM 29,80
6502 Programmieren in Assembler, DM 59,–
Umweltdynamik (Prospekt anfordern), DM 66,– (NEU)

Macintosh Programmierhandbuch mit MSBASIC 2.0 (Ende '85), DM 59,–
Einführung in die Mikrocomputer-Technik, DM 66,–
M68000-Familie, 2 Bände, DM 79,– und DM 69,–

Disk-Chirurg

Ein Programm zur Überprüfung schadhafter Diskettensektoren

von Wolf-J. Faust

Sind Sie sicher, daß Ihre Daten noch lange auf der Diskette erhalten bleiben? **DISK .CHIRURG** überprüft für Sie die Diskette auf dem Apple II. Sind die Daten nicht mehr sicher auf der Diskette gespeichert, können sie repariert werden.

Die Funktionsweise

Die Funktionsweise des Programms ist relativ einfach. Eine Diskette besteht aus 35 ($00-$22) Spuren mit jeweils 16 ($00-$0F) Sektoren. Soll ein Sektor auf einer Spur eingelesen werden, so wird der Schreib/Lesekopf zu der gewünschten Spur gefahren. Dann liest das DOS nacheinander alle Sektoren, bis der gewünschte Sektor gefunden ist. Die Anzahl der gelesenen Sektoren wird von dem Programm oben am Bildschirm mit der Meldung „GELESENE SEKTOREN" angezeigt (hexadezimal). Werden mehr als $0F Sektoren benötigt, so bedeutet dies, daß der Sektor beim ersten Umlauf der Spur nicht gelesen werden konnte. Ist der Sektor nach dem dritten Versuch immer noch nicht eingelesen (GELESENE SEKTOREN = $30), wird der Schreib/Lesekopf des Diskettenlaufwerks hörbar zur Spur Null gefahren, um wiederum dreimal zu versuchen, den Sektor einzulesen. Konnte der Sektor nach diesem sechsten Mal immer noch nicht richtig eingelesen werden, so wird ein „I/O ERROR" gemeldet.
Wer mehr zu diesem Thema wissen möchte, kann in dem Buch „All about DOS" von A.P.P.L.E. den Artikel „Disk Error Checking" von A. Rose nachlesen. Dort steht auch das kleine von mir verwendete Programm zum Umschreiben der DOS-Adressen und Anzeigen der Spur und der gelesenen Sektoren.
Mein Programm verwendet darüber hinaus die RWTS-Routine des DOS, um die Sektoren nacheinander einzulesen. Da die RWTS-Routine meist bei den verschiedenen DOS-Versionen gleich ist, läuft das Programm im allgemeinen auch mit gepatchtem DOS.
Sicherheitshalber sollte es jedoch zunächst an einer unwichtigen Diskette getestet werden, indem das Laufwerk während des Lesens kurz geöffnet wird, um einen Lesefehler vorzutäuschen.

Die Bedienung

Das Programm ist vollkommen menügesteuert:

D – Schaltet das Laufwerk um (Drive 1 nach Drive 2 oder umgekehrt).
R – Mit Hilfe der Taste „R" kann „REPARATUR" ein- oder ausgeschaltet werden. Bei eingeschalteter „REPARATUR" werden nicht mehr sehr gut erhaltene Daten repariert, d.h. sie werden erneut geschrieben. Ist „REPARATUR" ausgeschaltet, so wird nur die Diskette überprüft.
M – Diese Taste ist nur von Bedeutung, wenn die Option „REPARATUR" eingeschaltet ist. Ist die „MELDUNG" gewählt (AN), so wird sicherheitshalber gefragt, ob die Diskette tatsächlich repariert werden soll. Im anderen Fall wird der Sektor automatisch neu geschrieben.
S – Wird die Taste „S" gedrückt, muß zunächst die erste zu prüfende Spur eingegeben werden. Dies geschieht mit Hilfe der Cursor-Tasten. Der Links-Pfeil verkleinert und der Rechts-Pfeil vergrößert die Spur. Die Eingabe wird mit Return beendet. Danach wird nach der letzten Spur gefragt. Auch hier wird wieder die Eingabe mit den Cursor-Tasten und Return gesteuert. Beim Start des Programms werden die Werte zur Prüfung der ganzen Diskette eingesetzt (Spur $00-$22).
Leertaste – Mit der Leertaste starten Sie die Untersuchung der Diskette.
ESC – Die Escape-Taste beendet jede Eingabe oder die Ausführung des Programms und Sie kehren zu dem Menü zurück. Beim Drücken der Escape-Taste in dem Menü wird neu gebootet (da das DOS gepatcht wurde).

Meldungen

„GELESENE SEKTOREN" – Anzahl der gelesenen Sektoren (hexadezimal), bis der gewünschte Sektor richtig eingelesen wurde.
„LESE SPUR" – Anzeige der zuletzt gelesenen Spur (hexadezimal).
„;" – Der Sektor ist noch gut erhalten.
„1" – Der Sektor konnte bei dem ersten Versuch nicht eingelesen werden. Sie sollten die Diskette reparieren lassen.
„2" – Der Sektor konnte erst nach dem dritten Versuch eingelesen werden. Die Diskette ist fast zerstört. Lassen Sie die Diskette unbedingt reparieren!
„S" – Fehler bei der Diskettenstation: Irgend etwas Ungewöhnliches ist geschehen, oder der Sektor konnte nach dem sechsten Versuch nicht gelesen werden.
„W" – Die Diskette kann nicht repariert werden, da sie schreibgeschützt ist. Bitte entfernen Sie den Schreibschutz.
„U" – Lesefehler: Nach sechs Versuchen konnte der Sektor nicht eingelesen werden.

Anm. d. Red.: Das nochmalige Überschreiben eines Sektors schafft nicht in jedem Fall Abhilfe gegen Datenverlust (der häufigste Grund von Diskettenausfällen liegt an der Abnutzung der Magnetschicht durch Abrieb). Wird der Zustand eines Sektors beim Überprüfen der Diskette mit „1" oder „2" quittiert, sollte daher eine Kopie erstellt werden (z.B. mit COPYA oder QUICK-COPY aus Peeker, Heft 1/2-85).

Kurzhinweise

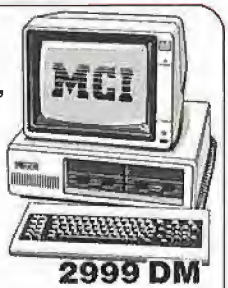
```
1. Zweck:
Überprüfung aller Sektoren einer Diskette
2. Konfiguration
II+, IIe oder IIc;
DOS 3.3
3. Test:
RUN DISK.CHIRURG
4. Sammeldisk:
DISK.CHIRURG
(Applesoft-Programm)
```

DISK.CHIRURG

```
100 PRINT CHR$ (21):A = 1:B$ = "AN ":C$ = B$:B = 0:C = 34
110 DATA 169,32,141,201,189,169,82,141,202,189,169,3,141,203,
    189,169,122,141,110,190,169,3,141,111,190,169,1,133,34,
    162,0,189,43,3,157,0,4,232,224,38,208,245,96,199,197,204,
    197,211,197
120 DATA 206,197,160,211,197,203,212,207,210,197,206,189,160,
    160,160,160,160,160,160,204,197,211,197,160,211,208,213,
    210,189,160,160,160,160,169,48,56,237,120,5,72,74,74,74,
    74,9,176,201,186,144
130 DATA 2,105,6,141,19,4,104,41,15,9,176,201,186,144,2,105,
    6,141,20,4,206,120,5,96,72,74,72,74,74,74,74,9,176,201,
    186,144,2,105,6,141,36,4,104,41,15,9,176,201,186
140 DATA 144,2,105,6,141,37,4,104,76,142,190,1,96,1,0,18,6,
    175,3,0,80,0,0,1,0,0,96,2,0,1,239,216,169,3,160,158,32,
    217,3,144,1,96,169,0,141,171,3,96
150 RESTORE : FOR I = 768 TO 962: READ I1: POKE I,I1: NEXT
160 HOME : VTAB 3: INVERSE : PRINT
    " 0000000000000000011111111111111111222 ": PRINT
    " 0123456789ABCDEF0123456789ABCDEF012 ":A$ =
    "0123456789ABCDEF"
170 FOR D = 5 TO 20: VTAB D:E = ASC ( MID$ (A$,D - 4,1)):
    PRINT CHR$ (E);: HTAB 37: PRINT CHR$ (E): NEXT
180 VTAB 9: HTAB 12: PRINT "DISK-CHIRURG";: NORMAL : PRINT
    : HTAB 12: PRINT "WOLF-J.FAUST": HTAB 12: PRINT
    "ODENWALDSCHULE": HTAB 12: PRINT "6148 HEPPENHEIM 4":
    CALL 768
190 VTAB 22: PRINT "(";: INVERSE : PRINT "D";: NORMAL :
    PRINT ")ISK= "; PEEK (928);" (";: INVERSE : PRINT "R";:
    NORMAL : PRINT ")EPARATUR=";B$;"  (";: INVERSE : PRINT
    "M";: NORMAL
200 PRINT ")ELDUNG=";C$;"(";: INVERSE : PRINT " ";: NORMAL
    : PRINT ")=START (";: INVERSE : PRINT "S";: NORMAL :
    PRINT ")PUR           (";: INVERSE : PRINT "ESC";:
    NORMAL : PRINT ")= STOP": POKE - 16368,0
210 D = PEEK ( - 16384): IF D < 127 THEN 210
220 POKE - 16368,0: IF D = 155 THEN POKE 34,0: HOME : CALL
    50688: END
230 IF D = 196 THEN D = PEEK (928): POKE 928,1: IF D = 1
    THEN POKE 928,2
240 IF D = 210 THEN A$ = B$:B$ = "AN ": IF A$ = "AN " THEN
    B$ = "AUS"
250 IF D = 205 THEN A$ = C$:C$ = "AN ": IF A$ = "AN " THEN
    C$ = "AUS"
260 IF D = 160 THEN 290
270 IF D = 211 THEN 400
280 GOTO 190
290 VTAB 8: FOR D = 1 TO 7: HTAB 12: PRINT SPC( 20): PRINT
    : NEXT : GOSUB 590: PRINT "DISKETTE IN LAUFWERK "; PEEK
    (928);" EINLEGEN": POKE - 16368,0
300 IF PEEK ( - 16384) < 127 THEN 300
310 IF PEEK ( - 16384) = 155 THEN 540
320 GOSUB 570: POKE - 16368,0: FOR F = B TO C: FOR G = 0 TO
    15: IF PEEK ( - 16384) = 155 THEN 540
330 POKE 930,F: POKE 931,G: POKE 938,1: POKE 939,0: CALL
    947: VTAB G + 5: HTAB F + 2: ON PEEK (939) / 16 + 1
    GOTO 340,500,510,510,510,510,510,510,520
340 D = PEEK (1043) - 176: IF D = 0 OR D = - 16 THEN PRINT
    ";": GOTO 530
350 INVERSE : PRINT D: NORMAL : IF B$ = "AUS" THEN 530
360 IF C$ = "AN " THEN GOSUB 590: PRINT "SOLL ICH
    REPARIEREN (J/N) ? N";: CALL 64528: POKE - 16368,0: GET
    A$: GOSUB 570: IF NOT (A$ = "J") THEN 530
370 POKE 938,2: CALL 947: POKE 938,1: VTAB G + 5: HTAB F +
    2: ON PEEK (939) / 16 + 1 GOTO
    380,500,510,510,510,510,510,510,520
380 IF PEEK (1043) - 176 < > D THEN 340
390 GOTO 530
400 GOSUB 590: PRINT "VON SPUR =>";:D = B:H = 0: GOSUB
    410:B = D: HTAB 1: PRINT "ZU ";:D = C:H = B: GOSUB
    410:C = D: GOTO 160
410 INVERSE :I = D: IF H > D THEN D = H
420 HTAB 12: PRINT INT (D / 16);:E = D - INT (D / 16) * 16
    + 176: IF E > 185 THEN E = E + 7
430 PRINT CHR$ (E);: POKE - 16368,0
440 E = PEEK ( - 16384): IF E < 127 THEN 440
450 IF E = 155 THEN D = E: NORMAL : GOTO 160
460 IF E = 149 THEN D = D + 1: IF D = 35 THEN D = 34
470 IF E = 136 THEN D = D - 1: IF D < H THEN D = H
480 IF E = 141 THEN NORMAL : RETURN
490 GOTO 420
500 INVERSE : PRINT "W": NORMAL : GOTO 530
510 INVERSE : PRINT "S": NORMAL : GOTO 530
520 INVERSE : PRINT "U": NORMAL
530 NEXT : NEXT
540 GOSUB 590: PRINT "TASTE DRUECKEN, UM WEITERZUMACHEN!":
    POKE - 16368,0
550 IF PEEK ( - 16384) < 127 THEN 550
560 GOTO 160
570 GOSUB 590: INVERSE : PRINT ";";: NORMAL : PRINT "=IN
    ORDNUNG  ";: INVERSE : PRINT "2,3";: NORMAL : PRINT
    "=FAST DEFEKT ";: INVERSE : PRINT "S";: NORMAL
580 PRINT "=FEHLER ";: INVERSE : PRINT "W";: NORMAL : PRINT
    " = SCHREIBGESCHUETZT     ";: INVERSE : PRINT "U";:
    NORMAL : PRINT " = UNLESBAR": RETURN
590 VTAB 21: HTAB 1: PRINT SPC( 120);: VTAB 22: RETURN
```


MCI XT16PC

XT16PC
16 Bit 8088 Rechner
vollständig Kompatibel,
ausgerüstet mit:
Colorcard,
256 RAM
Floppy Controller,
1 Drive, 360 K,
deutsche Tastatur,
130 W Netzteil



XT 16 PC 2999 DM
mit 6 MByte Winchester
XT 16 WLC 4499 DM

FLOPPY-DRIVES

FLÜSTER-FLOPPY
CHINON F-051 A
mit Kabel 379 DM

Billigversion!!!
CHINON A-II
mit Kabel und Gehause 345 DM

3,5" Pana. JU 363 479 DM
135 tpi 2 x 80 track 1 MByte
TEAC FD55B 379 DM
48 tpi 2 x 40 track 500 KByte
TEAC FD55F 449 DM
96 tpi 2 x 80 track 1 MByte
SANYO BFT 540 399 DM
48 tpi 2 x 40 track 500 KByte
SANYO BFT 580 449 DM
96 tpi 2 x 80 track 1 MByte
CHINON F-502 399 DM
48 tpi 2 x 40 track 500 KByte
REMEX RFD 480 329 DM
48 tpi 2 x 40 track 500 KByte
Apple AFD55A 399 DM
mit Kabel und Gehause

STATIONEN
2 x 8" SSDD KIT 1399 DM
2 x 8" DSDD KIT 1999 DM

DRUCKER

ESPRINT 80/ 801

- MX 80 + Graftrax Compatible
- 100Zeichen/9x11 Matrix,6Zeichensätze
- Parallel + RS232, Einzelblatt u. Traktor

für IBM angepaßt **749 DM** 699 DM

MCI GMBH
Postfach 20 04 01
J.-W.-Lindlar-Straße 8
5060 Bergisch Gladbach 2
Tel. (0 22 02) 3 10 07 - Tx. 8 873 518

Auf alle Geräte 6 Monate Garantie · Änderungen, die technischen Verbesserungen dienen, vorbehalten · Lieferbedingungen auf Anfrage · Lieferung solange Vorrat reicht ·

PEEKER Börse

Verkauf Software

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

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

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

Apple: AW-II-Manual (dtsch) + WP., VS = 25 DM, Privat-Datei (463 Sätze/Disk) VS = 80 DM, dBASE Privat-Datei VS = 80 DM, Inventur-Prog. für Steinmetze DM 250,– Manuale + WS + MM, EZ-Draw bei: 0 84 61 / 84 53 / 10 61

AP 22: Jetzt TURBO-PASCAL-Grafik. Include-File für Ihre Programme. Info frei, Prgm. auf Disk DM 70. Rüter, Rahdener Str. 65, 4955 Hille

MATHE-STAR mathematischer Editor für II/IIe: 4 freidefinierbare Zeichensätze; Blockkommandos; viele Spezialbefehle. Pr: 198,–, Manfred Albracht, Knospenweg 2, 5010 Bergheim, T. 0 22 71 / 9 45 04

Hochwertige Astrologie-Programme für Apple II/Plotter-Routinen für HP, EPSON HI-80, WATANABE, u.a. Alles weitere bei ROJASOFT, Postfach 44 61, CH-8022 Zürich oder unter T. 00 41-1-241 33 37

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

Für Ihren Apple II+ mit 80 Zeichen die Fakturierung! Programm 398,– Demodisk 38,– DM. Infos m. Rückumschlag. SSG Software Service Gruschke, Königstr. 53, 1000 Berlin 42

MEMDOS Kartenversion original neu, DM 398,–
Tel. 07 31 / 5 75 16

Apple Freunde, tolle Programme aus vielen Gebieten! Schach! Public Domain Software, jede Volume nur 22,– DM! Und Lehrprogramme! Gratisinfo: Fa. Waltraud Muhle, Waldwinkel 3, 2105 Seevetal 3

PIRATE DEFENCE 2.0 Kopierschutz. Die Antwort auf die neuen Nibble Kopierer wie: COPY II + 5.4, EDD, LOCKSMITH 5.0, ... Ab jetzt neben DOS 3.3 auch für ProDOS und DIVERSIDOS mit vielen Utilities. Info (50 Pf.) bei: Chr. Bregler, Tulpenstraße 2, 7519 Eppingen. Händleranfragen erwünscht.

fig-FORTH — DM 15,00
Assemblerlisting f. APPLE DM 15
FORTH auf Disk f. APPLE DM 45
Info kostenlos — C. Schmidt, Bungestr. 8, 3500 Kassel

********* **WARGAMES** *********

Die Besten von SSI für APPLE C 64 ATARI.
Info gegen 80 Pfg. RP

Computer-Service Th. Müller
Postfach 2526
7600 Offenburg

Apple II: Fahrschul-Lernprog. 32 Fragebögen, T. 0 71 51 / 6 15 84

Apple Works orig. Deutsch für IIc und IIe DM 480,–
T. 0 50 41 / 16 88

Verkauf Hardware

BASIS-108 (VERS.A4)-2 Siemens-F122 128KB-RAM-Apple + BASIS im ROM. Incl. Software 3000,– DM – 0 63 52 / 87 65.

TASTATUR DECODER für II+ AUTO REPEAT, groß/kleinsch. Umlaute, 8 Cursor- & 18 Funktionstasten. nur DM 99,–
T. 0 60 74 / 9 65 24

2 TEAC FD-55F Laufwerke + Ehring Controller (FDC4) unbenutzt 1250,– DM
T. 0 61 81 / 8 78 95

* **Grafik-Karte für Apple II+ komp** * Auflösung max. 512 × 512 Preis 450 DM inkl. Softw. u. Manual. 02 31 / 67 94 22

Disk-Laufwerk für Apple, 40 Tr., 100 % komp. – 250,– DM Apple-Schaltnetzteil 7,5 A – 150 DM.
T. 04 51 / 89 56 97 ab 16 Uhr.

Deutsche Sprachausgabe für Apple II Testbericht in Peeker 9/85 – M. Bönig, Wupperstr. 11, 4600 Dortmund 50, 02 31 / 71 87 21

Apple II + 64K, org., Disk, Contr., Z80, 80Z, Moni, Sep. Tast., Software, Lita, 2800 DM,
T. 040 / 279 04 00

Apple II (komp) 64 K, 2 Floppy, Z80, 80Z, Monitor, Joyst, Epson RX 80F/T Grafikinterface, div. Software, auch einzeln VB,
Tel. 0 26 78 / 10 01

Org. Applesoft-ROM-CARD DM 120,– Netzteil 5A – neu – DM 100,– H. Weise, Bergstr. 21, 7148 Remseck 3

Apple IIc, Monitor, Mouse IIe Joystick, Disk IIc, 6 Mon. alt VB 3700,– DM, T. 0 25 57 / 485 ab 18 h.

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

Apple II kompatibles 64 K System
– TEAC-Floppies: 160 K + 640 K
– Monitor (bernstein) mit 22 Mhz
– FX-80 F/T + Grappler Interface
– CP/M, 80 Z-Karte, Erphi-Cont.
Sehr günstiger Preis:
0 51 21 / 2 26 67

MACINTOSH UMRÜSTUNG von 128K auf 512 K DM 857,– + MWST FAST. Electronic GmbH, Kapuziner Str. 20, 8000 München 20, T. 089 / 77 72 52

Apple II komp. 64K CPM 80Z 2 Disk Operator Tast. Monitor Software 1 Jahr alt – 0 23 71 / 1 21 70

Kontakte

Suche Apple-Fan für Interessenaustausch im Raum Backnang – Chiffre P1006

Verschiedenes

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

APPLE REPARATUREN
SCHNELL + GUT, Innerh. 1 Wo. ist Ihr Gerät zurück 800/400 KB 31/2" MAC LW ab 880,– Gen. Musician Interface macht aus Apple II electron. Piano + Synth. Acryldiskbox f. 50 Dsks, Schl. 42,– 5.25 Diskreiniggset m. Fluid 26,–
PYRAMID Computer, Am Galgenberg 15, 7800 Freiburg,
T. 07 61 / 6 68 43 CPM IIc!

Hard- und Software für Apple-Computer gesucht. Wer schreibt gute Software? Auch Gebrauchtes angenehm. Zahle Höchstpreise! Suche noch Leute, die nebenbei verdienen wollen. Zuschriften Chiffre P1005

Umbau MAC von 128 K auf 512 K mit Qualitätsmaterial. RAMS (150µs) werden gesockelt – DM 848,–
näheres: T. 0 61 31 / 22 03 72

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

Wer hilft beim Anschluß einer Triumpf-Adler Typenrad-Schreibmaschine (SE 1010) an den Apple II? Heinrich J. Pohl, Weinstr. 54, 6749 Klingenmünster

Peeker und Peeker-Sammeldisketten alle Ausgaben zu verkaufen
Tel. 0 63 71 / 5 02 66 ab 18 Uhr

Anzeigenschluß für Ausgabe 12/85 ist am 25. 10. 85

Für Ihre Unterlagen

Abonnement bestellt

am: _____

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

peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

peeker

Abo-Karte

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

Ich wünsche jährliche Berechnung durch:

☐ Verlagsrechnung ☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank / PschA _____

Bankleitzahl _____ Kto.-Nr. _____

Datum _____ Unterschrift _____

peeker

Buch-Shop

Bitte senden Sie mir gegen Rechnung folgende Bücher:

Menge	Autor, Titel	à DM	gesamt DM

Datum _____ Unterschrift _____

peeker

Software-Karte

Bitte senden Sie mir
gegen Rechnung folgende Apple-Programme:

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

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

☐ CP/M ja ☐ CP/M nein
☐ Pascal ja ☐ Pascal nein

☐ Apple DOS 3.3, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 1, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 2, Begleitdiskette, DM 28,–
☐ Apple Assembler, Begleitdiskette, DM 28,–
☐ ProDOS-Editor 1.0, Programm, DM 98,–
☐ MMU 2.0, Programm, DM 98,–
☐ INPUT 2.0, Programm, DM 98,–
☐ Softbreaker 1.0, Programm, DM 48,–
☐ DB-Meister, Programm, DM 290,–
☐ Superplot, Programm, DM 48,–
☐ Superquick, Programm, DM 48,–

Datum _____ Unterschrift _____

Abo-Karte

Name

Firma

Abteilung

Straße

PLZ/Ort

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

Datum

Unterschrift

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

POSTKARTE

peeker
Leserservice

Postfach 10 28 69

6900 Heidelberg 1

Buch-Shop

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

peeker
Versandbuchhandlung

Postfach 10 28 69

6900 Heidelberg 1

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

peeker
Softwareabteilung

Postfach 10 28 69

6900 Heidelberg 1

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

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

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

MMU 2.0

Memory Managements Utilities

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

von U. Stiehl

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

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

Softbreaker 1.0

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

von U. Stiehl

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

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

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

ProDOS-Anpassung für Laufwerke mit 40 bis 160 Spuren

von Horst Hanke und Hans-Georg Hüneke

Das Disketten-Betriebssystem ProDOS ist ein leistungsstarkes DOS, das den Betrieb von Plattenspeichern bis 32 Megabyte zuläßt. Leider ist jedoch mit der Originalversion 1.0.1 nur der Betrieb von 35-Spur-Laufwerken möglich. Hieraus ergibt sich umgerechnet eine Diskettenkapazität von 280 Blocks (143K). Um nun den Besitzern von Laufwerken mit einer größeren Kapazität diesen Vorteil zugänglich zu machen, wird nachfolgend ein Patch-Programm beschrieben, das den Benutzer in die Lage versetzt, den vorhandenen File PRODOS für den Betrieb bis zu 1280 Blocks (= 160 Tracks) zu verändern. Außerdem wird auch der Filer dahingehend modifiziert, daß das Kopieren von 35-, 40- und 80-Track-Disketten sowie beliebiger Files möglich wird. Es können dann selbstverständlich auch Disketten der Formate 1 * 35 Tracks bis 2 * 80 Tracks formatieren werden.

Im folgenden werden nun Module vorgestellt, die in der Lage sind, das Original-ProDOS 1.0.1 sowie den FILER 1.0.1 zu modifizieren. Wegen der Länge kann hier nur der Hexdump wiedergegeben werden. Die Zeilennummern beziehen sich auf den Quellcode für den Apple-Assembler (unter ProDOS), der auf der Peeker-Sammeldiskette enthalten ist und nach ProDOS konvertiert werden kann.

1. Modifizieren des ProDOS 1.0.1 (640K-Diskdriver)

Um das ProDOS für größere Disketten-Laufwerke anzupassen, ist es notwendig, dem Diskdriver zu ermöglichen, mehr als 35 Spuren zu positionieren. Diese Modifikation läßt nun den Betrieb von max. 2 * 80 Spuren (= 160 Spuren = 640K) zu. Der erweiterte Driver (Datei PRODOS) wurde in einem Speicherbereich lauffähig gemacht, in dem normalerweise der RAM-Diskdriver installiert ist. Hieraus ergibt sich, daß der Betrieb größerer Disketten-Kapazitäten in diesem Fall auch einen Nachteil hat: Die RAM-Disk mußte geopfert werden. Diese Maßnahme hat jedoch den Vorteil, daß mit den meisten Programmen keine Kompatibilitätsprobleme auftreten. Es wäre nämlich durchaus möglich, diese Erweiterung in den noch freien Teil der Language-Card (Bank 2) zu legen. Spätestens wenn der unter ProDOS laufende Apple-Assembler gestartet wird, zeigt sich jedoch, daß auch dieser von dem Bereich der Bank 2 Gebrauch macht. Der 640K-Diskdriver besteht im einzelnen aus vier mehr oder weniger unabhängig voneinander laufenden Assembler-Routinen.

– Die Hauptroutine (Zeile 42-89)* hat nur die Aufgabe der Kopfumschaltung in Abhängigkeit von der momentan verwendeten Diskette. Dies bedeutet: Bei einer Konfiguration von 2 * 80 Spuren soll z.B. die Spur 136 positioniert werden. Da dieser Wert größer als 79 Tracks ist, muß notwendigerweise eine Kopfumschaltung erfolgen. (Dies bezieht sich nur auf den Ehring-Controller oder Kompatible. Beim Erphi-Controller wird z.B. von Spur zu Spur der Kopf umgeschaltet, so daß gerade Spurzahlen auf der einen Seite, ungerade Spurzahlen auf der anderen Seite gelesen werden). Aus dem **Bild 1** ist ersichtlich, daß sich die Spur 56 sowie die Spur 136 physikalisch an der gleichen Stelle befinden. Um nun auf Track 136 zu positionieren, muß ein Wert von 80 (Tracks) subtrahiert werden, was gleichzeitig mit einer Kopfumschaltung zu verbinden ist (Zeile 60-84).

– Dieses Ergebnis wird jetzt der „MYSEEK"-Routine (1,2) übergeben, die wiederum die Aufgabe hat, den Schreib/Lese-Kopf zu positionieren. Analog dazu verfährt man bei einer Konfiguration von 2 * 40 Tracks, wobei jedoch anstelle von 80 40 subtrahiert wird.

– Damit auch ein Mischbetrieb zwischen verschiedenen Disketten (1 * 35, 2 * 40, 1 * 80) möglich wird, war es erforderlich mit Hilfe der Routine „RECOGNIZE" (Zeile 215-252) eine Parametertabelle mit den momentan aktiven Laufwerken anzulegen. Diese Tabelle belegt einen Bereich von 7 Bytes, wobei jedes Byte einen Slot darstellt und jedes Halbbyte (Nibble) den Drive 1 oder 2 wiedergibt. Das **Bild 2** zeigt nun, wie in einem Nibble die Drive-Typkennung von einem 40- oder 80-Track-Laufwerk aussieht. Bei jeder Fehlpositionierung, bedingt durch einen Disketten-

* Zeilennummern beziehen sich auf Quelltext PRODOS,Ø (nur auf Sammeldiskette)

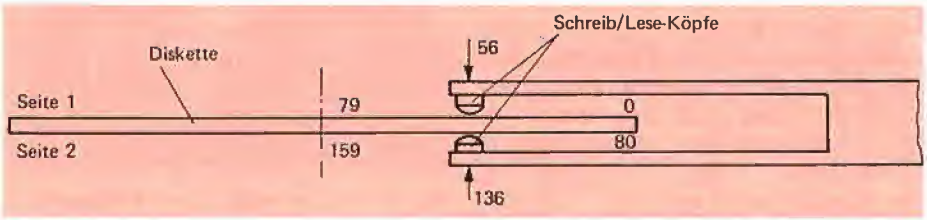


Bild 1. Zugriff auf eine doppelseitige Diskette (Ehring und Kompatible)

wechsel, wird diese Tabelle (Zeile 189-214) aktualisiert.
– Ausgehend von dem Inhalt dieser Tabelle kann nun die „ADJUST"-Routine (165-188), die vor jedem Seek-Befehl (Zeile 58) aufgerufen wird, die jeweils richtigen Parameter für das zur Zeit benutzte Laufwerk einstellen.
Als wichtigen Hinweis sollte man beachten, daß beim Einlegen einer 35-Track-initialisierten Diskette in ein 80-Track-Laufwerk eine (hardwaremäßige) Umschaltung auf 40-Spur-Betrieb erfolgen muß (48/96 TPI-Umschaltung).

Byte								
Nibble H				Nibble L				
Drive 2				Drive 1				
X	X	0	I	X	X	0	I	Drive 2=40 Drive 1=40
X	X	0	I	X	X	I	0	Drive 2=40 Drive 1=80
X	X	I	0	X	X	0	I	Drive 2=80 Drive 1=80
X	X	I	0	X	X	I	0	Drive 2=80 Drive 1=40

40-Track-Bit
80-Track-Bit

Bild 2. Nibble-Tabelle

2. Modifizieren des FILERSs 1.0.1

Eine Modifikation ist eigentlich nur aus dem Grunde notwendig, damit mehr als 35 Spuren formatiert werden können. Außerdem wird noch der Boot-Block verändert, um das Booten von nicht-apple-kompatiblen Disk-Controllern zu ermöglichen. Weiterhin wird beim Aufruf der beiden Menüpunkte „COPY A VOLUME" und „FORMAT A VOLUME" ein zusätzliches Menü eingebunden, das die verschiedenen Formatierungsformate zur Auswahl anbietet.
Die Änderung des Boot-Blocks 0 ist an zwei Stellen notwendig:
– Die Abfrage des Controller-Bytes $CnFF wird umgangen (Zeile 350-361)*.
– Sollte der Disk-Controller (3) (Ehring oder baugleiche) am DIL-Schalter 4 auf doppelseitigen Betrieb geschaltet sein, muß die Boot-Software (Block 0) in der Lage sein, den Schreib/Lese-Kopf auf die richtige Diskettenseite umzuschalten (Zeile 314-337).

Als nächstes muß der Formatter-Disk driver erweitert werden, damit das Formatieren von 1 * 35 bis 2 * 80 Tracks ermöglicht wird. Um diese Erweiterung im Filer zu integrieren, war es notwendig, den Kopier-Puffer um 1K (2 Blöcke) zu reduzieren (Zeile 398-408). Gleichzeitig muß dieser Bereich in der „Globel-Page" (4) vor Zugriffen durch das „MLI" geschützt werden (Zeile 388 - 397). Dies hat zur Folge, daß der Kopier-Puffer von 16K auf 15K gekürzt werden muß. Um dies zu verdeutlichen, sind in dem **Bild 3** der Original-Filer und der modifizierte Filer (NEW.-FILER) im Speicher dargestellt.
Die Filer-Änderungen werden erst beim Start durch eine Verschieberoutine installiert (Zeile 291-313 und 362-387), die von der Speicherstelle $2000 aufgerufen wird. Ansonsten läßt sich der Filer in gewohnter Weise bedienen, nur mit der zusätzlichen Möglichkeit, unter „VOLUME COMMANDS" verschiedene Formatierungsformate auswählen zu können (Zeile 48-109).

3. Beschreibung des Patch-Programms für ProDOS und FILER

Um die Patch-Files erzeugen zu können, muß eine Arbeitsdiskette erstellt werden, die die in **Tabelle 1** gelisteten Files enthalten muß.
Mit dem Aufruf „-SYSTEM.PATCH.1" wird das Patch-Programm gestartet. Hierauf meldet sich ein Menü mit den folgenden Möglichkeiten:

? – TUTOR
F – PATCH FILER
P – PATCH PRODOS
Q – QUIT

● Wählt man das „?", wird eine Kurzbeschreibung, ähnlich wie im Filer selbst, dargestellt. (Anm. d. Red.: Entgegen den normalen Peeker-Gepflogenheiten wird dieses Tutorial in Anlehnung an die FILER- und CONVERT-Hilfsmenüs in englischer Sprache ausgegeben.)
● Bei der Wahl von „F" wird der Filer-Patch-Vorgang folgendermaßen durchgeführt: Als erstes wird vom Programm der Original-FILER 1.0.1 geladen. Nachfolgend werden die modifizierenden Files FILER.1 sowie FILER.2 überlagernd dazugeladen. Somit ist diese Modifikation abgeschlossen und wird unter dem Namen **NEW.FILER** auf die Diskette zurück geschrieben.
● Durch die Wahl des Punktes „P", wird der PRODOS-Patch-Vorgang ausgeführt. Zuerst wird das Original-ProDOS 1.0.1, gefolgt von den modifizierenden Files PRODOS.1, PRODOS.2, PRODOS.3 und PRODOS.4, in den Speicher geladen. Damit ist auch dieser Patch abgeschlossen, und das neue ProDOS-System wird unter dem Namen **NEW.PRODOS** auf die Diskette zurückgeschrieben.
● Abschließend kann mit „Q" das Patch-Programm beendet werden.

4. Wie erstellt man eine Boot-Diskette

In diesem Abschnitt wird gezeigt, wie man z. B. eine 160-Track-formatierte Boot-Diskette erstellt.
Als erstes müssen die Files „NEW.PRODOS" und „NEW.FILER" mit den vorgenannten Punkten „F" und „P" des Patch-Programms erstellt werden (unter Angabe

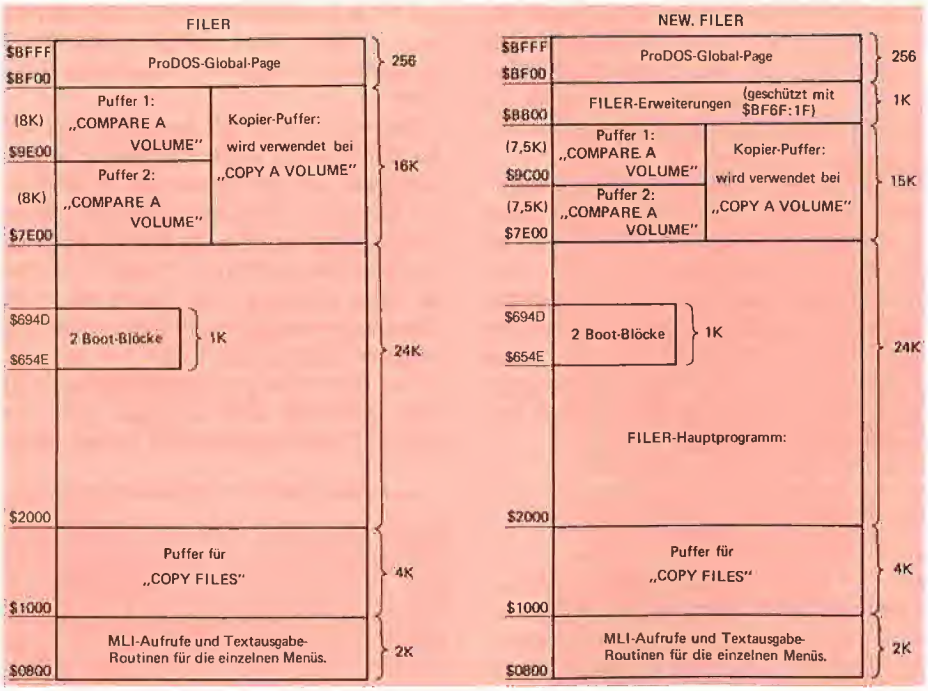


Bild 3. Speicheraufteilung des Original-FILER und des neuen NEW.FILER

* Zeilennummern beziehen sich auf Quelltext FILER.Ø (nur auf Sammeldiskette)

von „E“ für 160 Tracks). Durch Aufruf des neuen Filers mit dem Kommando „-NEW .FILER“ kann nun eine leere Diskette formatiert werden, damit diese einen modifizierten Boot-Block erhält. Als erstes wird dann mit Hilfe des Filer-Kommandos „FILE COMMAND“ und „COPY FILES“ der File „NEW.PRODOS“ auf die neu formatierte Diskette mit dem Namen „PRODOS“, kopiert. Abschließend muß noch von der ProDOS-Systemdiskette der File „BASIC.SYSTEM“ auf diese Diskette übertragen werden. Sollte diese Reihenfolge (PRODOS als erster Eintrag in dem Directory) nicht eingehalten werden, wird beim Booten sowie beim Aufruf durch das Strichkommando „-PRODOS“ die Fehlermeldung „RELOCATION/CONFIGURATION ERROR“ auf dem Bildschirm ausgegeben.

Literaturhinweis

(1) DOSSOURCE 3.3, LAZER MICRO SYSTEMS;
(2) Beneath Apple DOS von Don Worth und Peter Lechner;
(3) FDC4 Floppy-Disk-Controller von Ehring-Elektronik;
(4) Apple ProDOS für Aufsteiger von Ulrich Stiehl.

Kurzhinweise

1. Zweck:
Umstellung von ProDOS 1.0.1 auf Laufwerke mit höherer Kapazität.
2. Konfiguration:
Apple II+ oder IIe;
Ehring-Controller;
ProDOS 1.0.1.
3. Test:
-SYSTEM.PATCH.1
4. Sammeldisk:
PRODOS.0
FILER.0
SYSTEM.PATCH.0
(Quelltexte für Apple-Assembler unter ProDOS)
PRODOS.1
PRODOS.2
PRODOS.3
PRODOS.4
FILER.1
FILER.2
SYSTEM.PATCH.1
(Maschinenprogramme)

Tabelle 1

Liste aller benötigter Files zur Erstellung der gepatchten Files NEW.PRODOS und NEW.FILER.

NAME	TYPE	BLOCKS
*PRODOS	SYS	31
PRODOS.1	BIN	1
PRODOS.2	BIN	1
PRODOS.3	BIN	1
PRODOS.4	BIN	1
*FILER	SYS	51
FILER.1	BIN	3
FILER.2	BIN	1
SYSTEM.PATCH.1	BIN	6

Kurzanleitung

1. Auf eine normale ProDOS-1.0.1-Diskette die oben genannten Dateien aufnehmen.
   a) Mit Hilfe des KONVERT-Programms die Dateien von der Peeker-Sammeldisk (DOS 3.3) übertragen; oder
   b) die Dateien "PRODOS.0", "SYSTEM.PATCH.0" und "FILER.0" (als Quellcode auf Sammeldisk) mit dem Apple-Assembler assemblieren; oder
   c) die Hexdumps der Programme eingeben und wie angegeben abspeichern.
2. Gegebenfalls mit dem FILER die Dateien "PRODOS" und "FILER" (jeweils nur 1.0.1) ebenfalls auf Diskette kopieren.
3. Mit "-SYSTEM.PATCH.1" das Patch-Programm aufrufen und mit den Optionen "F" und "P" die Dateien "NEW.PRODOS" und "NEW.FILER" erzeugen.
4. Mit "-NEW.FILER" den gepatchten Filer aufrufen und eine neue Diskette unter Angabe der Laufwerkskapazität formatieren.
5. Die Datei "NEW.PRODOS" mit dem Namen "PRODOS" auf diese neu formatierte Diskette kopieren.
6. Gegebenenfalls auch die Dateien "BASIC.SYSTEM" und "NEW.FILER" übertragen.

BSAVE PRODOS.1, A$FEBE, S$0042

```
$FEBE: 48 20
$FEC0: 31 FF 68 C9 50 B0 0E 48
$FEC8: 8A 09 C0 8D D0 FE 8D 00
$FED0: C6 68 4C E0 FE A4 3E 99
$FED8: 88 C0 99 89 C0 ED C4 FE
$FEE0: 0A 8D 6F FB 60 20 98 FB
$FEE8: 08 AD 6F FB CD 56 FB F0
$FEF0: 03 28 38 60 28 60 AD 17
$FEF8: FF F0 14 20 50 FF A9 00
```

BSAVE PRODOS.2, A$FF00, L$0078

```
$FF00: 20 0C F9 AD 56 FB 20 0C
$FF08: F9 CE 17 FF 4C 9A F8 A9
$FF10: 27 CE 6A FB 4C B0 F8 00
$FF18: 00 A5 43 48 08 0A 4A 4A
$FF20: 4A 4A 4A AA BD 48 FF 28
$FF28: 30 04 6A 6A 6A 6A A8 68
$FF30: 60 20 19 FF 98 A0 02 4A
$FF38: B0 03 88 D0 FA B9 44 FF
$FF40: 8D C4 FE 60 A0 50 28 00
$FF48: 00 00 00 00 00 00 22 00
$FF50: 20 19 FF 48 98 6A B0 15
$FF58: 6A 0A 38 2A A8 68 30 0A
$FF60: 98 2A 2A 2A 2A 9D 48 FF
$FF68: D0 0D 98 D0 F8 6A 38 2A
$FF70: 0A A8 68 10 EB 30 F3 60
```

BSAVE PRODOS.3, A$F800, L$0010

```
$F800: A9 01 8D 17 FF A5 46 A6
$F808: 47 8E 56 FB E0 05 B0 26
```

BSAVE PRODOS.4, A$2900, L$0078

```
$2900: A0 78 B9 00 2C 99 00 FF
$2908: 88 C0 FF D0 F5 60
```

BSAVE FILER.1, A$BB00, L$0293

```
$BB00: AD B6 4E 48 A5 1A C9 03
$BB08: F0 02 68 60 68 C9 01 F0
$BB10: 05 C9 02 F0 01 60 20 58
$BB18: FC 20 96 BB AD 00 C0 10
$BB20: FB 2C 10 C0 C9 C1 F0 1C
$BB28: C9 C2 F0 20 C9 C3 F0 24
$BB30: C9 C4 F0 2C C9 C5 F0 33
$BB38: C9 9B F0 03 4C 16 BB 68
$BB40: 68 4C 92 BB A9 23 8D F4
$BB48: 79 4C 77 BB A9 28 8D F4
$BB50: 79 4C 77 BB A9 28 8D 76
$BB58: BD 0A 8D F4 79 4C 77 BB
$BB60: A9 50 8D 76 BD 8D F4 79
$BB68: 4C 77 BB A9 50 8D 76 BD
$BB70: 0A 8D F4 79 4C 77 BB 8D
$BB78: 44 42 A9 00 8D 46 42 A2
$BB80: 04 0E 46 42 CA F0 0B 0E
$BB88: 44 42 90 F5 2E 46 42 4C
$BB90: 84 BB AD B6 4E 60 A9 00
$BB98: 8D A3 BB A9 BB 8D A4 BB
$BBA0: A0 B5 B9 B5 BB F0 0D 09
$BBA8: 80 20 ED FD C8 D0 F3 EE
$BBB0: A4 BB D0 EE 60 AA AA AA
$BBB8: AA AA AA AA AA AA AA AA
$BBC0: AA AA AA AA AA AA AA AA
$BBC8: AA AA AA AA AA AA AA AA
$BBD0: AA AA AA AA AA AA AA AA
$BBD8: AA AA AA AA 0D AA A0 A0
$BBE0: A0 A0 A0 A0 A0 A0 A0 A0
$BBE8: A0 A0 A0 A0 A0 A0 A0 A0
$BBF0: A0 A0 A0 A0 A0 A0 A0 A0
$BBF8: A0 A0 A0 A0 A0 A0 A0 A0
$BC00: A0 A0 A0 AA 0D AA A0 A0
$BC08: A0 A0 C4 C9 D3 CB A0 C6
$BC10: CF D2 CD C1 D4 D4 C5 D2
$BC18: A0 C3 CF CE C6 C9 C7 D5
$BC20: D2 C1 D4 C9 CF CE A0 A0
$BC28: A0 A0 A0 AA 0D AA A0 A0
$BC30: A0 A0 A0 A0 A0 A0 A0 A0
$BC38: A0 A0 A0 A0 A0 A0 A0 A0
```

```
$BC40: A0 A0 A0 A0 A0 A0 A0 A0
$BC48: A0 A0 A0 A0 A0 A0 A0 A0
$BC50: A0 A0 A0 AA 0D AA AA AA
$BC58: AA AA AA AA AA AA AA AA
$BC60: AA AA AA AA AA AA AA AA
$BC68: AA AA AA AA AA AA AA AA
$BC70: AA AA AA AA AA AA AA AA
$BC78: AA AA AA AA 0D 0D AD AD
$BC80: C6 CF D2 CD C1 D4 D4 C5
$BC88: D2 A0 D0 C1 D2 C1 CD D3
$BC90: AD AD 0D 0D 0D A0 A0 A0
$BC98: A0 A0 A0 C1 A0 AD A0 A0
$BCA0: B3 B5 A0 D4 D2 C1 C3 CB
$BCA8: AC A0 D3 C9 CE C7 CC C5
$BCB0: A0 D3 C9 C4 C5 C4 0D 0D
$BCB8: A0 A0 A0 A0 A0 C2 A0
$BCC0: AD A0 A0 B4 B0 A0 D4 D2
$BCC8: C1 C3 CB AC A0 D3 C9 CE
$BCD0: C7 CC C5 A0 D3 C9 C4 C5
$BCD8: C4 0D 0D A0 A0 A0 A0 A0
$BCE0: A0 C3 A0 AD A0 A0 B4 B0
$BCE8: A0 D4 D2 C1 C3 CB AC A0
$BCF0: C4 CF D5 C2 CC C5 A0 D3
$BCF8: C9 C4 C5 C4 0D 0D A0 A0
$BD00: A0 A0 A0 A0 C4 A0 AD A0
$BD08: A0 B8 B0 A0 D4 D2 C1 C3
$BD10: CB AC A0 D3 C9 CE C7 CC
$BD18: C5 A0 D3 C9 C4 C5 C4 0D
$BD20: 0D A0 A0 A0 A0 A0 A0 C5
$BD28: A0 AD A0 A0 B8 B0 A0 D4
$BD30: D2 C1 C3 CB AC A0 C4 CF
$BD38: D5 C2 CC C5 A0 D3 C9 C4
$BD40: C5 C4 0D 0D 0D D3 C5 CC
$BD48: C5 C3 D4 A0 C1 CE A0 CF
$BD50: D0 D4 C9 CF CE A0 CF D2
$BD58: A0 BC C5 D3 C3 BE BA A0
$BD60: 0D 0D 0D A0 A0 A0 A0 A0
$BD68: A0 00 20 72 BD 0A 0E 24
$BD70: 7D 60 AC 37 7D C9 50 B0
$BD78: 10 48 98 4A 4A 4A 4A 09
$BD80: C0 8D 86 BD 8D 00 C6 68
$BD88: 60 99 88 C0 99 89 C0 ED
$BD90: 76 BD 60
```

BSAVE FILER.2, A$8693, L$00B0

```
$8693: A0 00 B9 00 84
$8698: 99 00 BB C8 D0 F7 B9 00
$86A0: 85 99 00 BC C8 D0 F7 B9
$86A8: 00 86 99 00 BD C0 93 F0
$86B0: 03 C8 D0 F3 A0 00 B9 27
$86B8: 87 99 9E 66 C0 1C F0 03
$86C0: C8 D0 F3 A9 14 8D 91 66
$86C8: A9 65 8D 9C 66 A9 EA 8D
$86D0: 06 67 8D 07 67 8D 08 67
$86D8: A9 20 8D B7 22 A9 00 8D
$86E0: B8 22 A9 BB 8D B9 22 A9
$86E8: A9 8D 6B 65 A9 00 8D 6C
$86F0: 65 AD 24 87 8D 00 20 AD
```

```
$86F8: 25 87 8D Ø1 2Ø AD 26 87
$87ØØ: 8D Ø2 2Ø A9 EA 8D 23 79
$87Ø8: A9 2Ø 8D 24 79 A9 6A 8D
$8718: 25 79 A9 BD 8D 26 79 A9
$8718: 1F 8D CA 2Ø A9 BB 8D E2
$8728: 2Ø 4C ØØ 2Ø A2 FF 9A D5
$8728: CE C1 C2 CC C5 AØ D4 CF
$8738: AØ CC CF C1 C4 AØ DØ D2
$8738: CF C4 CF D3 BD 88 CØ 4C
$8748: 68 Ø9 ØØ
```

BSAVE SYSTEM.PATCH.1, A$0803, L$099B

```
$Ø8Ø3: 2Ø 2F FB 2Ø 58
$Ø8Ø8: FC 2Ø 4D ØB 2Ø 2E Ø8 C9
$Ø81Ø: 3F FØ 28 C9 46 FØ ØE C9
$Ø818: 5Ø FØ ØD C9 51 FØ ØC 2Ø
$Ø82Ø: AE Ø8 DØ E8 6Ø 4C 4C Ø9
$Ø828: 4C D5 Ø8 4C A5 Ø8 AD ØØ
$Ø83Ø: CØ 1Ø FB 2C 1Ø CØ 29 7F
$Ø838: C9 1B 6Ø 2Ø 2F FB 2Ø 58
$Ø84Ø: FC A9 Ø8 2Ø 15 ØB 2Ø 93
$Ø848: Ø8 2Ø 86 Ø8 FØ Ø6 2Ø AE
$Ø85Ø: Ø8 4C 49 Ø8 2Ø 2F FB 2Ø
$Ø858: 58 FC A9 Ø9 2Ø 15 ØB 2Ø
$Ø86Ø: 93 Ø8 2Ø 86 Ø8 FØ Ø6 2Ø
$Ø868: AE Ø8 4C 62 Ø8 2Ø 2F FB
$Ø87Ø: 2Ø 58 FC A9 ØA 2Ø 15 ØB
$Ø878: 2Ø AØ Ø8 2Ø 2E Ø8 FØ 83
$Ø88Ø: 2Ø AE Ø8 4C 7B Ø8 2Ø 2E
$Ø888: Ø8 FØ Ø3 C9 ØD 6Ø 68 68
$Ø89Ø: 4C Ø3 Ø8 AØ Ø1 A9 17 85
$Ø898: 22 2Ø 5B FB 98 4C 15 ØB
$Ø8AØ: AØ Ø2 4C 95 Ø8 2Ø 2F FB
$Ø8A8: 2Ø 58 FC 4C DØ Ø3 Ø8 A9
$Ø8BØ: 2Ø 85 FB A9 Ø2 2Ø C9 Ø8
$Ø8B8: 8D 3Ø CØ A9 24 2Ø C9 Ø8
$Ø8CØ: 8D 3Ø CØ C6 FB DØ EC 28
$Ø8C8: 6Ø 38 48 E9 Ø1 DØ FC 68
$Ø8DØ: E9 Ø1 DØ F6 6Ø A9 Ø5 2Ø
$Ø8D8: 7B Ø9 A9 Ø6 2Ø 7B Ø9 A9
$Ø8EØ: Ø7 2Ø 7B Ø9 A9 Ø8 2Ø 7B
$Ø8E8: Ø9 A9 Ø9 2Ø 7B Ø9 2Ø FE
$Ø8FØ: Ø8 A9 ØA 2Ø 7B Ø9 A9 ØB
$Ø8F8: 2Ø 7B Ø9 4C Ø3 Ø8 A9 EA
$Ø9ØØ: 8D 5B 26 8D 5C 26 8D AB
$Ø9Ø8: 52 8D AC 52 8D ØC 53 A9
$Ø91Ø: 2Ø 8D A1 52 8D ØD 53 A9
$Ø918: 4C 8D AD 52 A9 E5 8D A2
$Ø92Ø: 52 A9 FE 8D A3 52 8D AF
$Ø928: 52 8D ØF 53 A9 F6 8D AE
$Ø93Ø: 52 A9 BE 8D ØE 53 A9 56
$Ø938: 8D C4 52 8D C9 52 A9 ØØ
$Ø94Ø: 8D DØ 2Ø 8D Ø8 25 A9 A9
$Ø948: 8D Ø7 25 6Ø A9 ØØ 2Ø 7B
$Ø95Ø: Ø9 A9 Ø1 2Ø 7B Ø9 A9 Ø2
$Ø958: 2Ø 7B Ø9 A9 Ø3 2Ø 7B Ø9
$Ø96Ø: 2Ø 6B Ø9 A9 Ø4 2Ø 7B Ø9
$Ø968: 4C Ø3 Ø8 A9 2Ø 8D ØØ 2Ø
$Ø97Ø: A9 93 8D Ø1 2Ø A9 86 8D
$Ø978: Ø2 2Ø 6Ø 18 ØA A8 B9 DD
$Ø98Ø: Ø9 85 FE B9 DE Ø9 85 FF
$Ø988: AØ ØØ B1 FE 99 ØØ Ø2 C8
$Ø99Ø: C9 ØD DØ F6 2Ø Ø3 BE BØ
$Ø998: Ø1 6Ø AD ØF BF C9 47 FØ
$Ø9A8: F8 AØ FF C8 B9 D8 Ø9 FØ
$Ø9A8: 22 CD ØF BF DØ F5 98 18
$Ø9BØ: 69 Ø3 A8 A9 15 2Ø 97 Ø8
$Ø9B8: 2Ø 42 FC 2Ø AØ Ø8 2Ø AE
$Ø9CØ: Ø8 2Ø 2E Ø8 DØ F8 68 68
$Ø9C8: 4C Ø3 Ø8 98 18 69 Ø3 2Ø
$Ø9DØ: 15 ØB 2Ø 8E FD 4C BB Ø9
$Ø9D8: 27 2B 46 48 ØØ F5 Ø9 ØD
$Ø9EØ: ØA 22 ØA 3Ø ØA 46 ØA 69
$Ø9E8: ØA 82 ØA 98 ØA AE ØA C4
$Ø9FØ: ØA DA ØA F1 ØA 42 4C 4F
$Ø9F8: 41 44 2Ø 46 49 4C 45 52
$ØAØØ: 2C 54 53 59 53 2C 41 24
$ØAØ8: 32 3Ø 3Ø 3Ø ØD 42 4C 4F
$ØA1Ø: 41 44 2Ø 46 49 4C 45 52
$ØA18: 2E 31 2C 41 24 38 34 3Ø
$ØA2Ø: 3Ø ØD 42 4C 4F 41 44 2Ø
$ØA28: 46 49 4C 45 52 2E 32 ØD
$ØA3Ø: 43 52 45 41 54 45 2Ø 4E
$ØA38: 45 57 2E 46 49 4C 45 52
$ØA4Ø: 2C 54 53 59 53 ØD 42 53
$ØA48: 41 56 45 2Ø 4E 45 57 2E
$ØA5Ø: 46 49 4C 45 52 2C 54 53
```

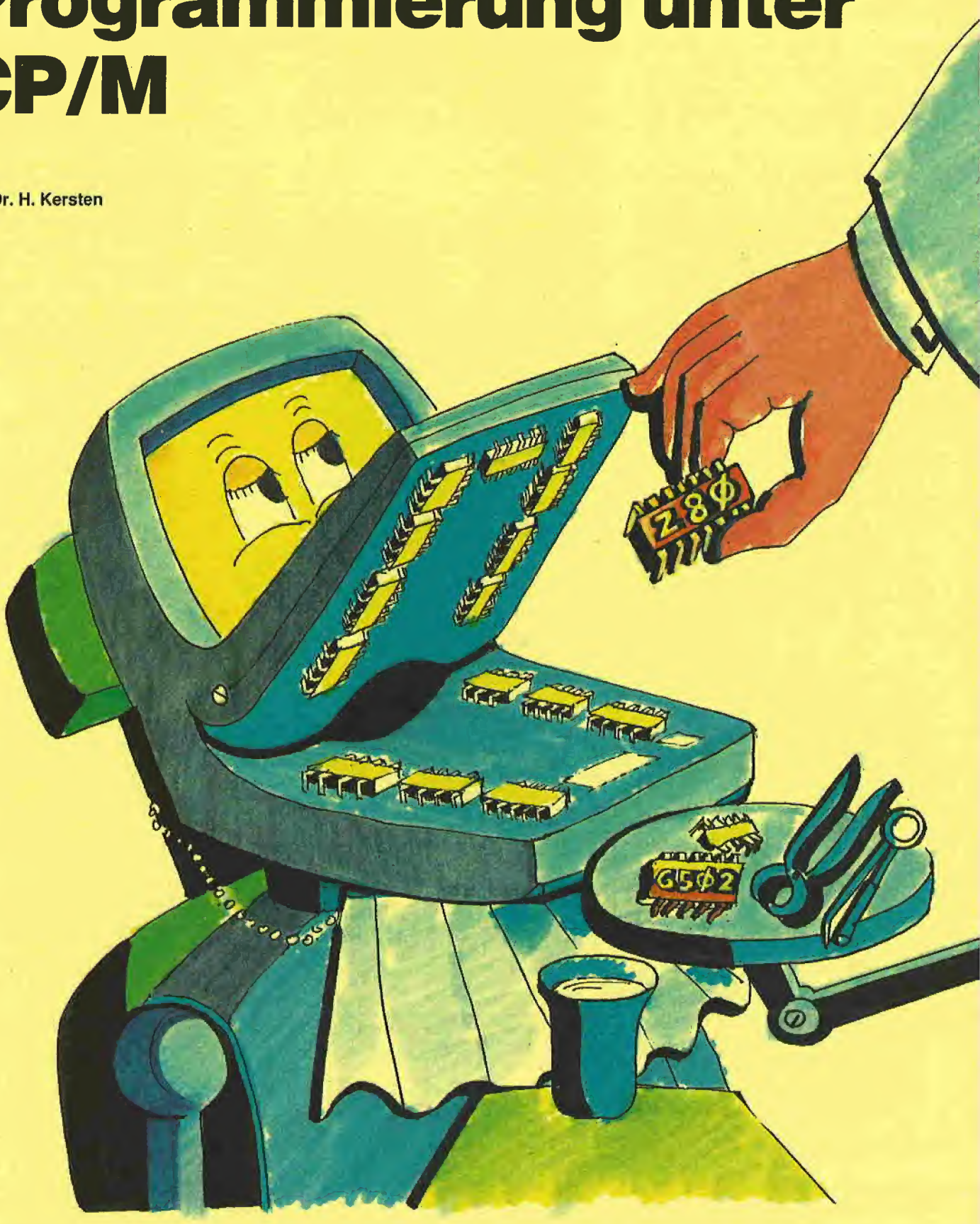
```
$ØA58: 59 53 2C 41 24 32 3Ø 3Ø
$ØA6Ø: 3Ø 2C 45 24 38 37 46 46
$ØA68: ØD 42 4C 4F 41 44 2Ø 5Ø
$ØA7Ø: 52 4F 44 4F 53 2C 54 53
$ØA78: 59 53 2C 41 24 32 3Ø 3Ø
$ØA8Ø: 3Ø ØD 42 4C 4F 41 44 2Ø
$ØA88: 5Ø 52 4F 44 4F 53 2E 31
$ØA9Ø: 2C 41 24 35 38 42 45 ØD
$ØA98: 42 4C 4F 41 44 2Ø 5Ø 52
$ØAAØ: 4F 44 4F 53 2E 32 2C 41
$ØAA8: 24 32 43 3Ø 3Ø ØD 42 4C
$ØABØ: 4F 41 44 2Ø 5Ø 52 4F 44
$ØAB8: 4F 53 2E 33 2C 41 24 35
$ØACØ: 32 3Ø 3Ø ØD 42 4C 4F 41
$ØAC8: 44 2Ø 5Ø 52 4F 44 4F 53
$ØADØ: 2E 34 2C 41 24 32 39 3Ø
$ØAD8: 3Ø ØD 43 52 45 41 54 45
$ØAE0: 2Ø 4E 45 57 2E 5Ø 52 4F
$ØAE8: 44 4F 53 2C 54 53 59 53
$ØAFØ: ØD 42 53 41 56 45 2Ø 4E
$ØAF8: 45 57 2E 5Ø 52 4F 44 4F
$ØBØØ: 53 2C 54 53 59 53 2C 41
$ØB8Ø: 24 32 3Ø 3Ø 3Ø 2C 4C 31
$ØB1Ø: 35 33 36 3Ø ØD 18 ØA A8
$ØB18: B9 37 ØB 85 FC B9 38 ØB
$ØB2Ø: 85 FD AØ ØØ B1 FC C9 ØØ
$ØB28: FØ ØC Ø9 8Ø 2Ø ED FD C8
$ØB3Ø: DØ F2 E6 FD DØ EE 6Ø 57
$ØB38: ØB 9C ØD 1F 11 42 11 4E
$ØB4Ø: 11 63 11 7A 11 8C 11 C4
$ØB48: ØD ØD ØF E2 ØF A9 ØØ 2Ø
$ØB5Ø: 15 ØB AØ 15 84 22 6Ø 2A
$ØB58: 2A 2A 2A 2A 2A 2A 2A 2A
$ØB6Ø: 2A 2A 2A 2A 2A 2A 2A 2A
$ØB68: 2A 2A 2A 2A 2A 2A 2A 2A
$ØB7Ø: 2A 2A 2A 2A 2A 2A 2A 2A
$ØB78: 2A 2A 2A 2A 2A 2A ØD 2A
$ØB8Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØB88: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØB9Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØB98: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBAØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØBA8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBBØ: 2Ø 2Ø 2Ø 2Ø 41 5Ø 5Ø 4C
$ØBB8: 45 2Ø 2D 2Ø 46 49 4C 45
$ØBCØ: 52 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBC8: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØBDØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBD8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBEØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBE8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØBFØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØBF8: 2Ø 2Ø 2Ø 2Ø 2Ø 31 32 38
$ØCØØ: 3Ø 2Ø 42 4C 4F 43 4B 53
$ØC8Ø: 2Ø 5Ø 41 54 43 48 2D 5Ø
$ØC1Ø: 52 4F 47 52 41 4D 2Ø 2Ø
$ØC18: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØC2Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC28: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC3Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC38: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC4Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØC48: 2Ø 2Ø 2Ø 2Ø 42 59 2Ø 48
$ØC5Ø: 2E 48 41 4E 4B 45 2Ø 2Ø
$ØC58: 26 2Ø 2Ø 48 2E 2D 2Ø 47
$ØC6Ø: 2E 48 55 45 4E 45 4B 45
$ØC68: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØC7Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC78: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC8Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC88: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØC9Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØC98: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 56
$ØCAØ: 45 52 53 49 4F 4E 2Ø 2Ø
$ØCA8: 31 2E 3Ø 2Ø 2Ø 2Ø 46 45
$ØCBØ: 42 2E 31 39 38 35 2Ø 2Ø
$ØCB8: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØCCØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØCC8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØCDØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØCD8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØCEØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØD 2A
$ØCE8: 2A 2A 2A 2A 2A 2A 2A 2A
$ØCFØ: 2A 2A 2A 2A 2A 2A 2A 2A
$ØCF8: 2A 2A 2A 2A 2A 2A 2A 2A
$ØDØØ: 2A 2A 2A 2A 2A 2A 2A 2A
$ØD8Ø: 2A 2A 2A 2A 2A 2A ØD ØD
$ØD1Ø: 2D 2D 5Ø 41 54 43 48 2D
$ØD18: 2D ØD 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØD2Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 3F 2Ø
$ØD28: 2D 2Ø 54 55 54 4F 52 ØD
```

```
$ØD3Ø: ØD 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØD38: 2Ø 2Ø 2Ø 2Ø 2Ø 46 2Ø 2D
$ØD4Ø: 2Ø 5Ø 41 54 43 48 2Ø 46
$ØD48: 49 4C 45 52 ØD ØD 2Ø 2Ø
$ØD5Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØD58: 2Ø 2Ø 5Ø 2Ø 2D 2Ø 5Ø 41
$ØD6Ø: 54 43 48 2Ø 5Ø 52 4F 44
$ØD68: 4F 53 ØD ØD 2Ø 2Ø 2Ø 2Ø
$ØD7Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØD78: 51 2Ø 2D 2Ø 51 55 49 54
$ØD8Ø: ØD ØD 5Ø 4C 45 41 53 45
$ØD88: 2Ø 53 45 4C 45 43 54 2Ø
$ØD9Ø: 41 4E 2Ø 4F 5Ø 54 49 4F
$ØD98: 4E 3A 2Ø ØØ 2D 5Ø 52 45
$ØDAØ: 53 53 2Ø 3C 52 45 54 3E
$ØDA8: 2Ø 54 4F 2Ø 43 4F 4E 54
$ØDBØ: 49 4E 55 45 2Ø 3C 45 53
$ØDB8: 43 3E 2Ø 54 4F 2Ø 45 58
$ØDCØ: 49 54 2D ØØ 2Ø 2Ø 2Ø 2Ø
$ØDC8: 2Ø 2Ø 2Ø 2Ø 54 55 54 4F
$ØDDØ: 52 3A 2Ø 2Ø 5Ø 41 54 43
$ØDD8: 48 2Ø 4D 45 4E 55 2Ø 4F
$ØDEØ: 5Ø 54 49 4F 4E 53 2E ØD
$ØDE8: ØD ØD 54 4F 2Ø 53 45 4C
$ØDFØ: 45 43 54 2Ø 41 4E 2Ø 4F
$ØDF8: 5Ø 54 49 4F 4E 2Ø 46 52
$ØE ØØ: 4F 4D 2Ø 54 48 45 2Ø 5Ø
$ØE Ø8: 41 54 43 48 2Ø 4D 45 4E
$ØE1Ø: 55 ØD 45 4E 54 45 52 2Ø
$ØE18: 54 48 45 2Ø 43 48 41 52
$ØE2Ø: 41 43 54 45 52 2Ø 5Ø 52
$ØE28: 45 43 45 44 49 4E 47 2Ø
$ØE3Ø: 59 4F 55 52 ØD 43 48 4F
$ØE38: 49 43 45 3A ØD ØD ØD ØD
$ØE4Ø: 3F 2Ø 2Ø 49 53 2Ø 46 4F
$ØE48: 52 2Ø 54 55 54 4F 52 2C
$ØE5Ø: 2Ø 42 55 54 2Ø 59 4F 55
$ØE58: 2Ø 4B 4E 45 57 2Ø 54 48
$ØE6Ø: 41 54 2Ø 4F 52 ØD 2Ø 2Ø
$ØE68: 2Ø 59 4F 55 2Ø 57 4F 55
$ØE7Ø: 4C 44 4E 27 54 2Ø 42 45
$ØE78: 2Ø 48 45 52 45 2E ØD ØD
$ØE8Ø: ØD 46 2Ø 2Ø 49 53 2Ø 46
$ØE88: 4F 52 2Ø 54 48 45 2Ø 46
$ØE9Ø: 49 4C 45 52 2Ø 5Ø 41 54
$ØE98: 43 48 2E ØD 2Ø 2Ø 2Ø 49
$ØEAØ: 54 2Ø 5Ø 41 54 43 48 45
$ØEA8: 53 2Ø 54 48 45 2Ø 4F 52
$ØEBØ: 49 47 49 4E 41 4C 2Ø 41
$ØEB8: 5Ø 5Ø 4C 45 2D 46 49 4C
$ØECØ: 45 52 2C ØD 2Ø 2Ø 2Ø 56
$ØEC8: 45 52 53 49 4F 4E 2Ø 31
$ØEDØ: 2E 3Ø 2E 31 2C 2Ø 2Ø 41
$ØED8: 4E 44 2Ø 43 52 45 41 54
$ØEEØ: 45 53 2Ø 41 2Ø 4E 45 57
$ØEE8: ØD 2Ø 2Ø 2Ø 4F 55 54 5Ø
$ØEFØ: 55 54 2D 46 49 4C 45 2C
$ØEF8: 2Ø 4E 41 4D 45 44 2Ø 27
$ØF ØØ: 4E 45 57 2E 46 49 4C 45
$ØF Ø8: 52 27 3A ØD ØØ ØD ØD ØD
$ØF1Ø: 54 48 45 2Ø 4E 45 57 2Ø
$ØF18: 46 49 4C 45 2Ø 43 4F 4D
$ØF2Ø: 4D 41 4E 44 2Ø 41 56 41
$ØF28: 49 4C 41 42 4C 45 2Ø 41
$ØF3Ø: 52 45 3A ØD ØD ØD ØD ØD
$ØF38: 2Ø 2Ø 2Ø 2Ø 41 2Ø 2D
$ØF4Ø: 2Ø 33 35 2Ø 54 52 41 43
$ØF48: 4B 53 2C 2Ø 53 49 4E 47
$ØF5Ø: 4C 45 2Ø 53 49 44 45 44
$ØF58: ØD ØD 2Ø 2Ø 2Ø 2Ø 2Ø 42
$ØF6Ø: 2Ø 2D 2Ø 34 3Ø 2Ø 54 52
$ØF68: 41 43 4B 53 2C 2Ø 53 49
$ØF7Ø: 4E 47 4C 45 2Ø 53 49 44
$ØF78: 45 44 ØD ØD 2Ø 2Ø 2Ø 2Ø
$ØF8Ø: 2Ø 43 2Ø 2D 2Ø 34 3Ø 2Ø
$ØF88: 54 52 41 43 4B 53 2C 2Ø
$ØF9Ø: 44 4F 55 42 4C 45 2Ø 53
$ØF98: 49 44 45 44 ØD ØD 2Ø 2Ø
$ØFAØ: 2Ø 2Ø 2Ø 44 2Ø 2D 2Ø 38
$ØFA8: 3Ø 2Ø 54 52 41 43 4B 53
$ØFBØ: 2C 2Ø 53 49 4E 47 4C 45
$ØFB8: 2Ø 53 49 44 45 44 ØD ØD
$ØFCØ: 2Ø 2Ø 2Ø 2Ø 2Ø 45 2Ø 2D
$ØFC8: 2Ø 38 3Ø 2Ø 54 52 41 43
$ØFDØ: 4B 53 2C 2Ø 44 4F 55 42
$ØFD8: 4C 45 2Ø 53 49 44 45 44
$ØFEØ: ØD ØØ 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
$ØFE8: 54 55 54 4F 52 3A 2Ø 2Ø
$ØFFØ: 5Ø 41 54 43 48 2Ø 4D 45
$ØFF8: 4E 55 2Ø 4F 5Ø 54 49 4F
$1ØØØ: 4E 53 2E ØD ØD ØD ØD 5Ø
```

```
$1ØØ8: 2Ø 2Ø 49 53 2Ø 46 4F 52
$1Ø1Ø: 2Ø 54 48 45 2Ø 5Ø 52 4F
$1Ø18: 44 4F 53 2Ø 5Ø 41 54 43
$1Ø2Ø: 48 2E ØD 2Ø 2Ø 2Ø 49 54
$1Ø28: 2Ø 5Ø 41 54 43 48 45 53
$1Ø3Ø: 2Ø 54 48 45 2Ø 4F 52 49
$1Ø38: 47 49 4E 41 4C 2Ø 41 5Ø
$1Ø4Ø: 5Ø 4C 45 2D 5Ø 52 4F 44
$1Ø48: 4F 53 2C ØD 2Ø 2Ø 2Ø 56
$1Ø5Ø: 45 52 53 49 4F 4E 2Ø 31
$1Ø58: 2E 3Ø 2E 31 2C 2Ø 2Ø 41
$1Ø6Ø: 4E 44 2Ø 43 52 45 41 54
$1Ø68: 45 53 2Ø 41 2Ø 4E 45 57
$1Ø7Ø: ØD 2Ø 2Ø 2Ø 4F 55 54 5Ø
$1Ø78: 55 54 2D 46 49 4C 45 2C
$1Ø8Ø: 2Ø 4E 41 4D 45 44 2Ø 27
$1Ø88: 4E 45 57 2E 5Ø 52 4F 44
$1Ø9Ø: 4F 53 27 3A ØD ØD ØD 2Ø
$1Ø98: 2Ø 2Ø 54 48 45 2Ø 2Ø 27
$1ØAØ: 4E 45 57 2Ø 2E 5Ø 52 4F
$1ØA8: 44 4F 53 27 2Ø 2Ø 41 43
$1ØBØ: 43 45 5Ø 54 ØD 2Ø 2Ø 2Ø
$1ØB8: 44 49 53 4B 2D 44 52 49
$1ØCØ: 56 45 53 2Ø 55 5Ø 2Ø 54
$1ØC8: 4F 2Ø 2Ø 32 2Ø 2A 2Ø 38
$1ØDØ: 3Ø 2Ø 54 52 41 43 4B 53
$1ØD8: ØD 2Ø 2Ø 2Ø 59 4F 55 2Ø
$1ØEØ: 43 41 4E 2Ø 4D 49 58 2Ø
$1ØE8: 44 49 53 4B 2Ø 44 52 49
$1ØFØ: 56 45 53 2Ø 41 4D 4F 4E
$1ØF8: 47 ØD 2Ø 2Ø 2Ø 28 2Ø 31
$11ØØ: 2Ø 2A 2Ø 33 35 2Ø 29 2Ø
$11Ø8: 2D 2Ø 28 2Ø 32 2Ø 2A 2Ø
$111Ø: 38 3Ø 2Ø 29 2Ø 2Ø 54 52
$1118: 41 43 4B 53 2E ØD ØØ ØD
$112Ø: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2D 2D
$1128: 2D 2Ø 5Ø 52 45 53 53 2Ø
$113Ø: 3C 45 53 43 3E 2Ø 54 4F
$1138: 2Ø 45 58 49 54 2Ø 2D 2D
$114Ø: 2D ØØ 49 2F 4F 2Ø 2D 2Ø
$1148: 45 52 52 4F 52 ØØ 44 49
$115Ø: 53 4B 2Ø 57 52 49 54 45
$1158: 2Ø 5Ø 52 4F 54 45 43 54
$116Ø: 45 44 ØØ 5Ø 41 54 48 2Ø
$1168: 4F 52 2Ø 46 49 4C 45 2Ø
$117Ø: 4E 4F 54 2Ø 46 4F 55 4E
$1178: 44 ØØ 4E 4F 2Ø 52 4F 4F
$118Ø: 4D 2Ø 4F 4E 2Ø 56 4F 4C
$1188: 55 4D 45 ØØ 2D 2D 2Ø 5Ø
$1190: 52 4F 44 4F 53 2Ø 45 52
$1198: 52 4F 52 2Ø 2D 2D ØØ
```

Z80-Assembler-Programmierung unter CP/M

von Dr. H. Kersten



Viele Apple-II-Systeme sind mit der Z80-Softcard nachgerüstet oder haben den Z80-Prozessor (vor allem bei neueren Nachbauten) bereits auf dem Motherboard integriert. Dieser Prozessor ist Voraussetzung dafür, daß das CP/M-Betriebssystem beim Apple II lauffähig ist. Im folgenden liegt stets die CP/M-Version 2.20 (56K) zugrunde. (Aber auch die Version 2.23 (60K) kann für die Testbeispiele verwendet werden.) Über dieses Betriebssystem und das hierunter laufende MBASIC ist im Peeker schon berichtet worden. Der vorliegende Artikel möchte den Leser nun mit der Assembler-Programmierung des Z80-Prozessors unter CP/M vertraut machen. Beabsichtigt ist dabei kein allgemeiner Assembler-Kurs, sondern eine schnelle Einführung für solche Leser, die schon Kenntnisse in irgendeiner Assembler-Sprache besitzen; für Apple-Besitzer wird das der 6502-Assembler sein, für den U. Stiehl in Peeker, Heft 7/85 eine Einführung gegeben hat. Dieser Prozessor wird hier oft zu Vergleichszwecken herangezogen.

Neben der Beschreibung der Registerstruktur, einem Überblick (mit Beispielen) über die Z80-Befehle und Adressierungsarten werden auch Übersetzer und Debugger beschrieben. Danach werden die System-Routinen (und deren Ansteuerung) für Console- und File-I/O dargestellt. Nach Durcharbeiten dieses Aufsatzes wird der Leser bereits über eine gut bestückte Makro-Bibliothek verfügen, die die Programmierung erheblich vereinfacht.

Zuvor einige Bemerkungen zum leidigen Problem der Z80- und 8080-Normierungen. Der 8080-Prozessor hat einen Befehlssatz, der eine Untermenge des Z80-Befehlsvorrats darstellt, d.h. der Z80-Prozessor kann alle 8080-Befehle ausführen, enthält aber noch weitere Befehle. Selbst bei gemeinsamen Befehlen sind beim Z80 oft noch weitergehende Adressierungsarten möglich. Das CP/M-Betriebssystem samt Assembler und Debugger (ASM.-COM/DDT.COM) ist nun in 8080-Normierung erstellt worden. Fazit: die exklusiven Z80-Befehle sind weder beim Assembler noch beim Debugger direkt verfügbar. Man kann sie höchstens als Hex-Bytes eingeben – was natürlich der Lesbarkeit nicht gerade zuträglich ist. Abhilfe schaffen hier nur ein Z80-Assembler wie z.B. M80/MACRO80 und ein Z80-Debugger wie ZSID oder DDTZ. Ein weiteres Problem kommt hinzu: Selbst die beiden Prozessoren gemeinsamen Befehle werden unterschiedlich geschrieben, d.h. besitzen sehr verschiedene Mnemonics. Man ist bei der Standard-Software fast gezwungen, auch noch 8080-Assembler zu lernen. Dieser Weg soll in diesem Artikel nicht beschritten werden. Grundlage ist stets die Z80-Normierung. Minimalvoraussetzung für die sinnvolle Nutzung dieses Artikels ist deshalb ein Z80-Assembler. Empfehlenswert ist in jedem Fall ein Z80-Debugger.

1. Die Z80-Registerstruktur

In **Abb. 1** ist der Registersatz des Z80-Prozessors dargestellt. Die Abbildung enthält die Registerbezeichnungen und die Registerlänge in Bits. Wer als gestandener 6502-Programmierer (gewöhnt an 6 Register) zum ersten Mal diese Vielfalt (je nach Zählweise 11-20 Register) sieht, sollte nicht verzweifeln oder gar dieses Heft zur Seite legen.

Die rechte Bildhälfte (Hintergrund-Register/Cache-Register) erledigt sich sehr schnell. Außerdem muß man in der Einstiegsphase ja nicht gleich mit allen Registern simultan arbeiten. Im Laufe der Zeit lernt man dann automatisch den Komfort einer größeren Registerzahl zu schätzen.

Beim Z80 wird das A-Register auch als Akkumulator bezeichnet und hat aufgrund spezieller Adressierungsarten eine Sonderstellung. Die allgemeinen Register B-C-D-E-H-L haben wie das A-Register eine Länge von 8 Bits, lassen sich aber paarweise (wie aus Abb. 1 ersichtlich) zu 16-Bit-Registern zusammenschalten. In der Assembler-Programmierung werden dann die Registerbezeichnungen BC-DE-HL benutzt. Das F-Register des Z80 entspricht von der Aufgabe her dem P-Register des 6502 – enthält also die Status-Bits. Die Bedeutung der einzelnen Bits (Bit 5 und 3 werden nicht benutzt) ist in **Abb. 2** wiedergegeben.

Die Indexregister IX und IY ähneln dem X- und Y-Register beim 6502. Sie besitzen jedoch hier eine Länge von 16 Bits. Das

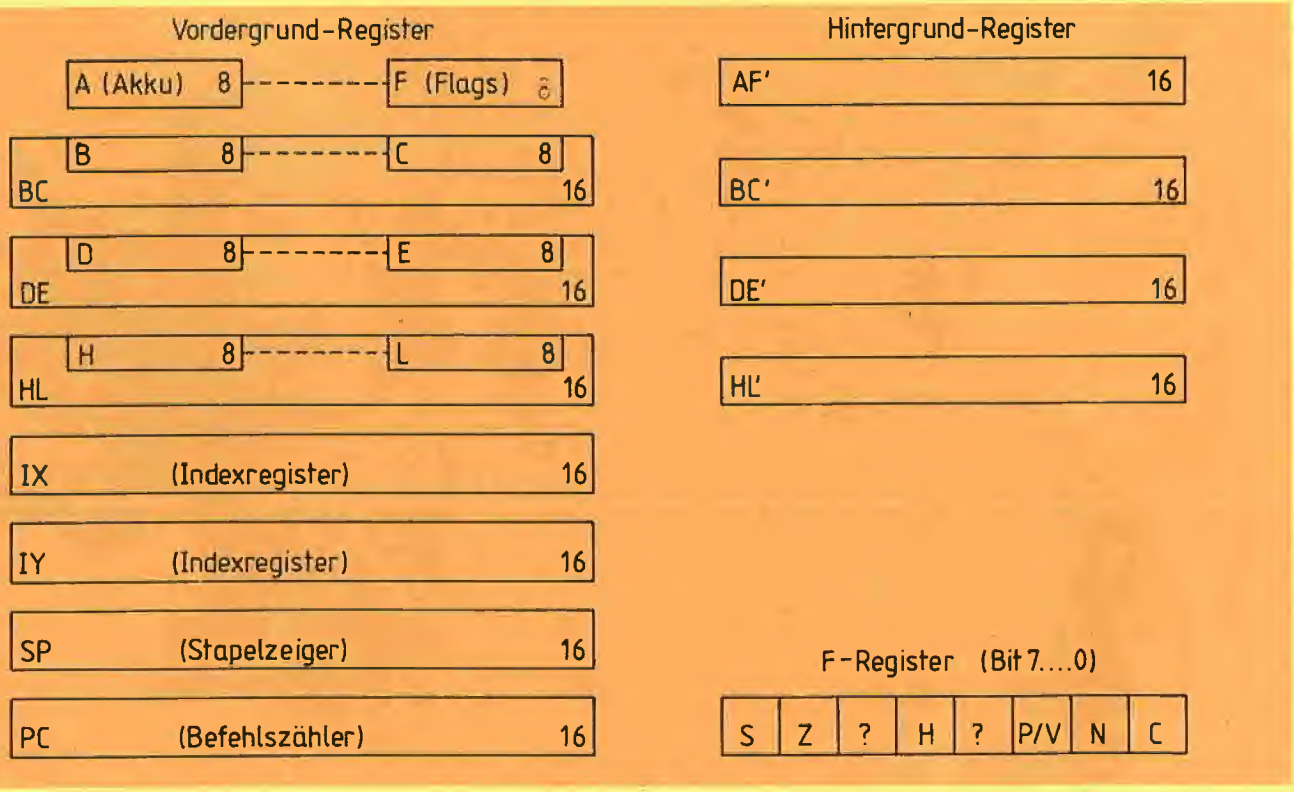


Abb. 1

Sign-Flag	Zero-Flag	Half-Carry	Parity/Overflow		N-Flag	Carry
1: negativ 0: positiv	1: Resultat = 0 0: Resultat <> 0	1: Nibble Übertrag 0: kein Übertrag	1: Parity gerade 0: Par. ungerade	1: Überlauf 0: kein Überlauf	1: nach Subtraktion 0: nach Addition	1: Übertrag 0: kein Übertr.

Abb. 2

gleiche gilt im Vergleich für das SP-Register (Stackpointer/Stapelzeiger beim Z80) und das S-Register des 6502. Der Befehlszähler (PC-Register) darf bei beiden Prozessoren als funktionsgleich angenommen werden. Bleiben noch die Hintergrund-Register in Abb. 1 zu erläutern. Sinn dieser Register ist es, eine schnelle Rettung der Vordergrund-Register (ohne Strich ') zu ermöglichen. Diese Hintergrund-Register und ebenso die Indexregister sind beim 8080-Prozessor nicht vorhanden. Vorteil: Das CP/M-Betriebssystem benutzt diese Register nicht. Man kann hier sehr schön auch über längere Zeit Registerinhalte retten.

2. Programmierung der Hintergrund-Register

Für diesen Registersatz AF' und BC'-DE'-HL' gibt es nur 2 Operationstypen:
a) Austauschen („exchange") von A mit A' und gleichzeitig F mit F' – in Z80-Assembler:

```
EX AF,AF'
```

b) Austauschen der 16-Bit-Register BC mit BC', DE mit DE', HL mit HL' gleichzeitig – Assembler-Befehl:

```
EXX
```

Ein zweimaliges Austauschen mit EX bzw. EXX stellt wieder den Originalzustand her. Für den EX-Befehl werden das A-Register und das F-Register als 16-Bit-Kombination betrachtet (ebenso für die PUSH/POP-Befehle), es handelt sich hier aber um kein echtes 16-Bit-Register.

3. Die Stack-Bearbeitung

Der Stapelzeiger hat 16 Bits Länge, kann also theoretisch einen 64K-Stack-Bereich verwalten. Beim 6502 sind nur 256 Bytes direkt verfügbar. Während dort der Stack byte-weise bearbeitet wird, liegt beim Z80 eine wortweise (16 Bits) Organisation vor. Der Befehl

```
PUSH reg (reg = AF-BC-DE-HL-IX-IY)
```

schiebt 2 Bytes des jeweiligen 16-Bit-Registers auf den Stack (und erniedrigt den Stapelzeiger sinngemäß um 2), der Befehl

```
POP reg
```

bewerkstelligt den umgekehrten Vorgang. Zweiter Unterschied zum 6502: Beim Z80 kann der Stack-Bereich an beliebiger Stelle im Speicher definiert werden – da ja SP insgesamt 64K adressieren kann. Beim Start eines Programms zeigt SP stets auf den CP/M-System-Stack, der laut CP/M-Beschreibung dem Benutzer einschließlich Rücksprungadresse ins Betriebssystem insgesamt 8 freie 16-Bit-Stack-Plätze zur Verfügung stellt. Wenn man im eigenen Programm nur eine sehr geringe Tiefe in der Unterprogrammschachtelung hat, wird man hiermit auskommen – zumal alle Aufrufe für I/O-Bearbeitung über einen „local stack" innerhalb des Systembereichs abgewickelt werden. Unter Umständen muß man sich allerdings einen eigenen Stack definieren. Hierzu lädt man beim Programmstart das SP-Register mit der Endadresse des reservierten Bereichs:

```
START: LD SP,STACK
...
       DS 32              ; 16 Einträge
STACK  EQU $-1
```

Der LD-Befehl (= Load) lädt den Stackpointer mit der Endadresse des Stack-Bereichs, der hier eine Länge von 32 Bytes = 16 Einträge hat (DS = Define Storage). (6502-Kenner: Irgendein Problem mit der LD-Zeile? Noch etwas Geduld!) Das $-Zeichen stellt den momentanen Stand des Adreßzählers dar, zeigt also auf das 33. Byte.

4. Beenden eines Programms

Hat man den System-Stack benutzt, so reicht (ähnlich wie unter DOS) ein RET(urn) zum Beenden eines Programms. Genauso gut, aber im Fall eines eigenen Stacks praktisch zwingend ist es, mit „JP 0" einen CP/M-Warmstart auszuführen (wie JMP $3D0 unter DOS).

5. Die Adressierungsarten

Der in Z80-Assembler-Programmen wohl am häufigsten vorkommende Befehl ist der LD-Befehl = Lade-Befehl. Grund: Er vereinigt in sich alle 6502-Transferfunktionen wie LDA/LDX/LDY, STA/STX/STY, TXA/TAX, TYA/TYA, usw... Die beim Z80 vorkommenden Adressierungstypen sollen deshalb an diesem Befehl erläutert werden – und zwar ausgehend von der Programmier-Praxis. Eine wichtige Normierung vorweg: Die Transfer-Richtung ist beim LD-Befehl durch LD Ziel,Quelle gegeben. Sinngemäß gilt diese Normierung auch für alle anderen Befehle.

5.1. Laden von Konstanten (8/16 Bits) in Register („immediate")

Mit

```
LD A,27h
```

wird A mit hexadezimal 27 geladen. Statt A kann natürlich jedes andere 8-Bit-Register A-L verwendet werden. Im Z80-Assembler werden Hex-Zahlen durch Anhängen von „h" (oder „H") gekennzeichnet (entspricht also dem vorangestellten #$ beim 6502). Um Verwechslungen mit Adreßbezeichnungen zu vermeiden, wird z. B. die Hex-Zahl FF als 0FFh notiert. Das Voranstellen von „0" verhindert, daß der Assembler FF oder FFh als Namen einer Speicherzelle interpretiert.
Konstanten mit 16 Bits Länge können in BC-DE-HL-SP-IX-IY geladen werden. Beispiel:

```
LD HL,0FF3Ch
LD IX,73Ah        ; = 073Ah
```

5.2. Register 1 nach Register 2 schieben

Beispiel:

```
LD A,H
```

Hier wird H nach A transferiert – und nicht umgekehrt. (Haben Sie sich schon daran gewöhnt? Nebenbei bemerkt entspricht diese Definitionsrichtung auch derjenigen beim CP/M-Betriebssystem. Denken Sie an PIP oder REN.) Jede andere Kombination von 8-Bit-Registern ist möglich.

Die 16-Bit-Register kann man *nicht* auf diese Weise transferieren. Ausnahme: das SP-Register kann von HL-IX-IY geladen werden. Korrekt ist also

```
LD SP,HL
LD SP,IX ; oder IY
```

Es existiert aber kein Befehl, der etwa BC nach HL lädt. Hierfür könnte man aber z.B. die folgenden Anweisungen benutzen:

```
PUSH BC
POP HL
oder
LD H,B
LD L,C
```

Speziell für die Register DE und HL kann man auch den EX(change)-Befehl ausnutzen:

```
EX DE,HL
```

Hierdurch werden die Registerinhalte ausgetauscht (was nicht ganz der gestellten Aufgabe entspricht, aber oft sehr nützlich ist). Andere Kombinationen der Vordergrund-Register sind bei EX nicht möglich.

5.3. Transfer Register → Speicher und umgekehrt

Hier gibt es eine ganze Reihe von Alternativen.

5.3.1. Absolute Adressierung

Nach dem Schema LD (Adresse),A wird das A-Register an die angegebene Adresse transferiert. (Umgekehrter Fall: LD A,(Adresse)). Für die anderen 8-Bit-Register existieren *keine* entsprechenden Befehle. Beispiel:

```
LD (ZELLE1),A ; schiebt A nach ZELLE1
```

Alle 6502-Anwender werden spätestens jetzt nachdenklich (oder schon oben bei dem Stack-Beispiel?). Beim 6502 bedeutet das Einklammern der Adresse nämlich eine indirekte Adressierung, d.h. A würde nicht nach ZELLE1, sondern (zumindest theoretisch) in die Speicherzelle transferiert, auf die ZELLE1 zeigt. Anders beim Z80: Das Einklammern „(ZELLE1)" ist gleichbedeutend mit „Inhalt von ZELLE1". Ohne Klammerung bedeutet „ZELLE1" nunmehr „Adresse von ZELLE1". (Beim 6502 wird dieser Fall durch Voranstellen von # oder / behandelt.) Hierbei handelt es sich um eine Immediate-Adressierung, da „ZELLE1" vom Assembler ja einen (16-Bit-)Wert zugeordnet bekommt.

Bei den 16-Bit-Registern hat man mehr Möglichkeiten bei der Registerauswahl. Der Formalismus ist hier LD (Adresse),r16 bzw. LD r16,(Adresse) (mit r16 = BC-DE-HL-SP-IX-IY). Vergleichen Sie jetzt noch einmal das Beispiel bei der Stack-Bearbeitung mit LD SP,STACK. Hier wird die Adresse (!) von STACK nach SP geladen.

5.3.2. Indirekte Adressierung (mit Zeigern)

Sofort ein Beispiel: Hat das Register HL den Inhalt 0179h, so bewirkt der Befehl

```
LD A,(HL)
```

daß der Inhalt der Speicherzelle 0179h nach A transportiert wird. HL enthält also den Adreßzeiger. Bei Registern bedeutet also (...) tatsächlich indirekte Adressierung (ein wenig verwirrend?). Das Beispiel entspricht etwa dem 6502-Fall „LDA (ZEIGER),X", wenn die Zellen ZEIGER/ZEIGER+1 die Inhalte $79, $01 haben und X mit 0 geladen ist.
Die Indexregister IX und IY können ebenfalls als Zeiger verwendet werden (mit Offset 0, vgl. 5.3.3). Als Zeiger können beim Z80 auch die Register BC und DE dienen, sofern das Ziel oder die Quelle das A-Register ist. Mit

```
LD (BC),A
```

wird der Inhalt von A nach der Zelle transferiert, auf die BC zeigt. Für alle anderen 8-Bit-Register dürfen nur HL-IX-IY als Zeiger verwendet werden.
Ein indirektes Laden von 16-Bit-Werten (2 Speicher-Bytes) etwa nach BC (vom Typ „LD BC,(HL)") ist prinzipiell nicht möglich – muß also in 2 Transfers (je ein Byte) aufgelöst werden. Ausnahme ist die Verwendung von SP im Rahmen eines EX-Befehls. Die Anweisung

```
EX (SP),HL
```

tauscht HL mit dem Speicherplatz, auf den SP zeigt, aus. Statt HL sind wiederum auch IX-IY möglich.

5.3.3 Indizierte Adressierung (mit IX/IY-Register)

Die Indexregister IX und IY lassen sich nicht direkt mit dem X- und Y-Register des 6502 vergleichen – eher stellen sie einen Ersatz für die Zero-Page-Adressierung dar. Die Befehlsfolge

```
LD IX,200h
LD A,(IX)
```

lädt den Inhalt der Speicherzelle 0200h nach A. Damit liegt praktisch die indirekte Adressierung wie unter 5.3.2 beschrieben vor. Von der Ausführungsgeschwindigkeit her sind in jedem Fall die Register BC-DE (sofern zulässig) und HL vorzuziehen (vgl. die **Operationstabellen** im Anhang)! Interessant werden die Indexregister erst durch die Möglichkeit, noch Offsets einbauen zu können. Die formale Schreibweise lautet (für das Beispiel oben) LD A,(IX+d). Will man etwa den Inhalt von 0210h nach A laden, so verfährt man wie folgt:

```
LD IX,200h
LD A,(IX+10h)
```

Der Offset 10h wird vor der Ausführung des 2. Befehls zu IX addiert (intern, ohne IX zu verändern) und adressiert somit die Zelle 0210h. Das gleiche Resultat erzielt man aber auch mit dem einzigen Befehl LD A,(210h). Wozu also das ganze? Nehmen wir an, 0200h ist die Anfangsadresse einer Tabelle, die im folgenden intensiv bearbeitet wird. Mit LD IX,200h setzt man das Indexregister auf den Tabellenanfang. Von nun an braucht man nur noch mit den relativen Adressen (den Offsets) zu arbeiten – Beispiel:

```
LD IX,TAB ; Adresse (!) von TAB
LD A,(IX+5)
LD B,(IX+27)
LD C,(IX+118)
```

Wird die eigentliche Verarbeitung (ab Zeile 2) als Unterprogramm geschrieben, so ist man in der Lage, durch Umladen von IX mehrere Tabellen auf die gleiche Art und Weise zu bearbeiten. Statt IX kann überall auch IY verwendet werden. Der Offset darf Werte zwischen 0 und 255 annehmen. Transferiert man Elemente einer Tabelle 1 in eine andere Tabelle 2, so kann IX die Startadresse von Tabelle 1, IY die von Tabelle 2 enthalten:

```
LD IX,TAB1
LD IY,TAB2
LD A,(IX+17)
LD (IY+32),A ; TAB1(17) → TAB2(32)
LD A,(IX+99)
LD (IY+33),A ; TAB1(99) → TAB2(33)
```

6502-Kenner werden bei der Aufzählung der Adressierungsarten die echte Zero-Page-Adressierung vermißt haben. Eine solche existiert beim Z80 nicht. (Eine gewisse Ausnahme bildet der hier nicht behandelte Interrupt-Befehl RST.)

6. Arbeiten mit Z80-Debuggern

Jetzt wird es ernst. Die bis hierhin graue Theorie soll am Rechner verifiziert werden. Vielleicht hatten Sie schon die Idee, die vorgestellten Assembler-Zeilen einzu-

tippen und auszuprobieren. Genau darum geht es in diesem Abschnitt. Den einfachsten Zugang bieten Debugger wie DDTZ oder ZSID (die mit DDT in der Bedienung praktisch übereinstimmen). Hiermit ist man in der Lage, Programme schrittweise auszuführen und zu überwachen (ähnlich wie beim alten 6502-Monitor mit STEP/TRACE-Betrieb bzw. dem exzellenten BUGBYTER, der ja auch unter ProDOS verfügbar ist). Die nachfolgenden Details gelten explizit für ZSID.

6.1. Eingabe von Befehlszeilen

Nach dem Start des Debuggers geben Sie A (wie „Assemblieren") und eine Adresse ein. Vorschlag: A100. (Auf die Speicherverwaltung und die freien Bereiche kommen wir später zu sprechen.) Im Bild erscheint jeweils links der aktuelle Adreßzähler. Man gibt nun etwa die obigen Assembler-Zeilen ein. Dabei müssen Sie alle symbolischen Namen (TAB1, TAB2, ...) durch feste Adressen ersetzen. Wählen Sie hierfür glatte Adreßwerte in der Größenordnung 1000h, 1100h, ... Eine Leerzeile beendet die Eingabe.

6.2. Kontrolle der Befehlszeilen

Das Gegenstück zum Assemblieren ist das Disassemblieren. Wie beim 6502-Monitor ist hierfür „L" das Schlüsselzeichen. Geben Sie L100 ein und kontrollieren Sie Ihre ersten Z80-Befehle.

6.3. Registerinhalte ein-/ausgeben

Der Formalismus ist Xreg<Return>. Dabei ist „reg" die Registerbezeichnung in abgekürzter Form: P(C), (I)X, (I)Y, B(C), D(E), H(L), A, ... – der eingeklammerte Buchstabe entfällt. Der jeweilige Registerinhalt wird angezeigt, man kann nun den gewünschten Wert eingeben. Mit <Return> übernimmt man den alten Wert. Mit X<Return> erhält man alle Register auf einen Blick. Die Status-Bits werden mit Xs<Return> abgerufen/eingegeben. Dabei ist für s zu setzen: C = Carry, M = Sign-Flag, Z = Zero, E = Parity/Overflow und I = Half-Carry. Das N-Flag wird nicht separat behandelt.

6.4. Eingabe/Ausgabe von Daten oder einzelnen Bytes

mit Dadr <Return> bzw. Dadr1,adr2 <Return> lassen sich Hex-Dumps ausgeben. Eingabe einzelner Bytes geht wie folgt: Sadr <Return> (S = Substitute). Angezeigt wird die laufende Adresse (beginnend ab „adr") und der alte Speicherinhalt. Danach geben Sie das neue Byte ein. Dieses Frage- und Antwort-Spiel wiederholt sich solange, bis ein Punkt als Ende-Kriterium eingegeben wird.

6.5. Programmstart/Haltepunkte

Geben sie beim Aufruf des Debuggers sofort einen File-Namen an (Beispiel: ZSID TEST.COM), so wird das entsprechende Programm in den Speicher geladen. Geben Sie nun als erstes (im Beispiel) ITEST.COM ein. Hiermit wird der File-Name TEST.COM als Workfile festgelegt. Ist nämlich während des Testbetriebs das Programm zerstört worden, so läßt es sich nunmehr mit R<Return> neu laden. Mit Gadr <Return> oder G <Return> wird das Testprogramm ab adr oder beim PC-Stand (= 0100h nach dem Laden) gestartet. Soll das Programm bei bestimmten Haltepunkten hp1, hp2, ... stoppen, so gibt man Gadr,hp1,hp2,...<Return> oder G,hp1,hp2,...<Return> ein. Die Programmausführung wird unterbrochen, sobald eine der Teststopp-Adressen hp1, hp2, ... erreicht ist.

6.6. TRACE-Betrieb

Die Anweisung T <Return> (= „TRACE") führt jeweils einen Befehl aus. Angezeigt werden alle Registerinhalte vor (!) der Befehlsausführung, sowie der ausgeführte Befehl. Wie unter 6.3 beschrieben, kann man jetzt die aktuellen Registerinhalte inspizieren. Einfacher ist es meist, mit T(RACE) fortzufahren und die neuen Registerinhalte im nächsten Schritt zu sehen. Durch T20 <Return> können Sie beispielsweise 20 Befehle ohne Unterbrechung durchlaufen und bekommen fortlaufend alle Register angezeigt.
Falls der Leser noch keine Erfahrungen mit Debuggern hat, könnten sich die „sample sessions" mit DDT in der CP/M-Beschreibung als sehr nützlich erweisen. In jedem Fall ist ein gewisses Training mit einem Z80-Debugger sehr empfehlenswert.

7. Spezielle Z80-Befehle

Nach den Beispielen zur Adressierung und möglicherweise ersten Testerfahrungen nun ein Blick auf die **Operationstabellen** im Anhang. In der 1. Gruppe sind Befehle aufgeführt, die von der Struktur her 16-Bit-Befehle sind. Danach folgen die Verzweigungsbefehle. In Gruppe 3 sind die 8-Bit-Befehle dargestellt. Der LD-Befehl und einige andere tauchen naturgemäß in der 8- und 16-Bit-Gruppe auf. Die Tabellen zeigen, daß der Z80-Prozessor Befehle mit einer Länge zwischen 1 und 4 Bytes kennt. Aus Gründen des Aufwands kann nur ein Teil der Operationen ausführlicher besprochen werden.

7.1. ADD/SUB und ADC/SBC

Im Gegensatz zum 6502 existieren hier mit ADD und SUB auch Befehle, die bei der Ausführung ein evtl. gesetztes Carry-Bit nicht berücksichtigen. Beide Typen von Operationen setzen aber das C-Flag nach Ausführung entsprechend dem Ergebnis. Einfache Additionen (OP1) + (OP2) → (OP3) sehen aus, wie in **Abb. 3** wiedergegeben.

```
8-Bit Operanden:          16-Bit Operanden:

LD   A,(OP1)              LD  HL,(OP1)
LD   HL,OP2               LD  BC,(OP2)
ADD  A,(HL)               ADD HL,BC
LD   (OP3),A              LD  (OP3),HL
```

Abb. 3: Additionsbeispiel

Die 16-Bit-Addition läßt sich natürlich auch in 2 * 8 zerlegen. Am Beispiel ist zu erkennen, daß HL die Rolle eines 16-Bit-Akkumulators hat. Für Subtraktionen geht man analog vor. Ein SUB existiert für 16-Bit-Subtraktion nicht – nur SBC.

7.2. INC und DEC

Bei den 8-Bit-Operanden ist die Funktion von INC/DEC wie beim 6502. Wichtig bei 16-Bit-Operanden ist aber, daß die Flags unbeeinflußt bleiben. Steht HL etwa auf 1, so ist nach Ausführung von DEC HL das Z-Flag nicht (!) gesetzt. Zum Test auf 0 muß vielmehr ein CP-(Compare-)Befehl verwendet werden.

7.3. DAA

Beim 6502 ist man gewöhnt, durch SED/CLD die BCD-Arithmetik zu aktivieren oder auszuschalten. Beim Z80-Prozessor geht man etwas anders vor. Hier wird nach der jeweiligen Operation der Anpassungsbefehl DAA benutzt. Das N-Flag im Status-Register ist nach vorausgegangener Addition auf 0, bei Subtraktion auf 1 gesetzt. Ergibt sich im Register bei der arithmetischen Operation ein Übertrag vom unteren Nibble zum oberen, so wird das H-Flag (Half Carry) gesetzt. Mit Hilfe dieser Flags (evtl. auch noch C-Flag aus vorausgegangener Operation) korrigiert der DAA-Befehl das arithmetische Ergebnis im Sinne der BCD-Logik. Einfaches Beispiel:

```
LD   A,27h
ADD  A,38h  ; Ergebnis: 5Fh
DAA         ; Ergebnis: 65h
```

Dieser mächtige DAA-Befehl wirkt allerdings nur bei 8-Bit-Arithmetik.

7.4. AND, XOR und OR

Die Funktionsweise dieser 8-Bit-Befehle dürfte hinlänglich bekannt sein. Wichtige Anmerkung: Nach Ausführung dieser Operationen ist das Carry-Bit auf 0 gesetzt. Da der Z80 keinen Befehl zum Löschen des Carry-Bits kennt, wird hierzu häufig etwa XOR A (löscht gleichzeitig A-Register), AND A oder OR A verwendet.

7.5. CP

Mit CP läßt sich das A-Register mit jedem anderen Register, einem Immediate-Wert oder einer Speicherzelle vergleichen (im Sinne einer internen Subtraktion). Für die möglichen Adressierungen vergleiche man die Tabellen. Ein wesentlicher Unterschied zum 6502: Das Carry-Bit wird genau entgegengesetzt verwendet. Die Befehle LD A,5 und CP 6 setzen beim Z80 das Carry-Flag zu 1 ((A) < Operand). Beim 6502 ergibt LDA #$05 und CMP #$06 gerade C = 0.

7.6. JP und JR

Neben dem Standard-Sprungbefehl JP gibt es hier ähnlich wie beim 65C02 einen relativen Jump JR. Die Sprungdistanz muß effektiv zwischen -126 und +129 liegen. Sowohl JP als auch JR (mit Einschränkungen) können noch von Bedingungen abhängig gemacht werden – wie die Tabelle im Anhang zeigt. Die Kürzel haben dabei (wie auch für CALL/RET) folgende Bedeutung: NZ (Not Zero, Z-Flag = 0), Z (Zero, Z = 1), NC (No Carry, C = 0), C (Carry, C = 1), PE (Parity Even/gerade Anzahl von 1en, P/V = 1), PO (Parity Odd/ungerade, P/V = 0), P (Positive, S-Flag = 0), M (Minus, S = 1).
Ein indirekter Sprung (wie beim 6502 z.B. JMP (KSW)) ist beim Z80 über Register möglich, und zwar mit HL-IX-IY als Adreßzeiger, also z.B. JP (IX).

Eine spezielle Z80-Befehlsgruppe bilden die Blockbefehle. Man erkennt sie daran, daß den Mnemonics LD und CP noch die Kürzel I, D, IR oder DR angehängt sind. Mit diesen Befehlen ist es möglich, durch einen Befehl Speicherblöcke zu verschieben oder nach einem festen Byte abzusuchen. Das I bedeutet „Inkrement", D „Dekrement" und R „Repeat".

7.7. LDIR/LDDR (und LDI/LDD)

Enthält HL den Zeiger auf die Quelle, DE den Zeiger auf das Ziel, so werden soviele Bytes von Quelle nach Ziel transferiert, wie das Zählregister BC angibt. Bei den I-Befehlen werden die Zeiger dabei inkrementiert (Transfer vorwärts), bei D dekrementiert (Transfer rückwärts). Beispiel:

```
LD BC,100h ; 256 Bytes
LD HL,TAB1
LD DE,TAB2
LDIR
```

Hier wird der Bereich TAB1 + 0 bis TAB1 + 255 nach TAB2 verschoben.
Die Befehle LDI/LDD sind als Vorstufe zu LDIR/LDDR anzusehen. Ihnen fehlt das „Repeat", sie inkrementieren/dekrementieren aber die Zeiger HL/DE/BC nach jedem (Einzel-)Transfer.

7.8. CPIR/CPDR (und CPI/CPD)

Hiermit läßt sich ein Speicherblock nach einem Byte durchsuchen. Das Ergebnis (gefunden/nicht gefunden) zeigt sich im Parity-Flag. Beispiel: Durchsuchen von TAB (Länge 4K) nach „*".

```
LD HL,TAB    ; Startadresse
LD A,"*"     ; Muster
LD BC,1000h; Anzahl Bytes
CPIR         ; Blockvergleich
JP PE,FOUND
JP PO,NOTFND
```

Beim Ausgang FOUND enthält HL die um 1 erhöhte Adresse des gefundenen Bytes (hier „*"). Die Befehle ohne Repeat-Funktion sind CPI/CPD.

7.9. LD-Nachtrag

Zu den LD-Befehlen sind zwei wichtige Nachträge erforderlich:
a) Dieser Befehl setzt keine Flags! Die beim 6502 übliche Programmierung indirekter Sprünge (z.B. LDA #$00 und BEQ ...) ist beim Z80 nicht möglich, da LD A,0 kein Z-Flag setzt. Als Ersatz stehen aber für das erwähnte Problem einfachere relative Sprungbefehle zur Verfügung.
b) Das 16-Bit-LD – z.B. LD (ZELLE),HL – schreibt den Registerinhalt im Low/High-Format in den Speicher, bzw. interpretiert den Speicherinhalt als Low/High beim Transfer in ein Register. Hat HL den Inhalt 55FFh, so steht im Beispiel ab ZELLE FFh, 55h. Mit LD HL,(ZELLE) würde in HL wieder 55FFh stehen.

7.10. Schiebebefehle

Bei den Schiebe- und Rotationsbefehlen reicht es, dem Assembler-Kenner die folgenden Skizzen zu präsentieren:

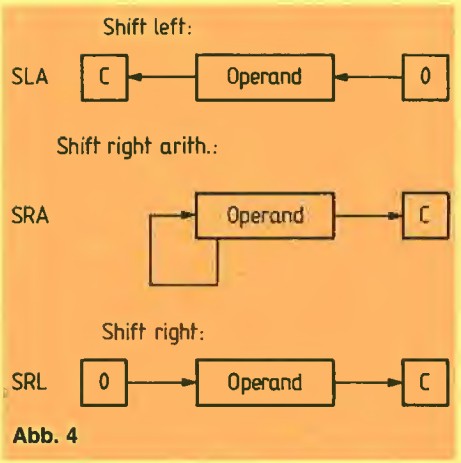


Abb. 4

Beim arithmetischen Rechtsschieben SRA wird das Vorzeichen-Bit reproduziert, beim SRL dagegen eine 0 nachgezogen.

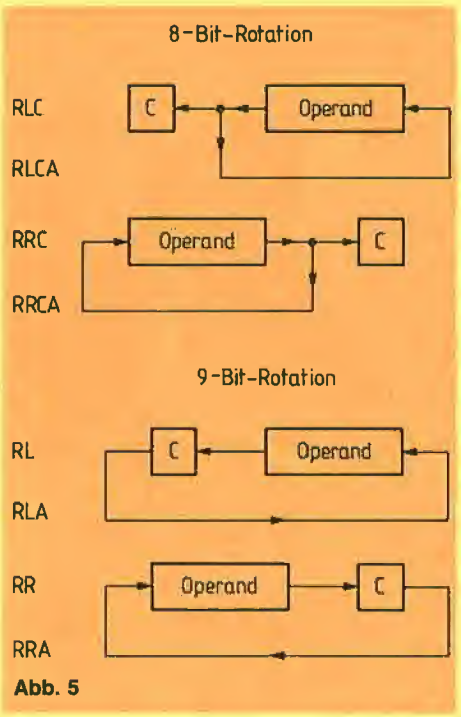


Abb. 5

Die Rotationsbefehle unten wirken praktisch auf 9 Bits (8 Register-Bits + Carry-Bit), oben auf 8 Bits. Die Befehle RLA, RRA, RLCA, RRCA adressieren (implizit) nur das A-Register, während die anderen Befehle auf verschiedene Operanden wirken können.

7.11. RLD/RRD

Über die im vorausgehenden Abschnitt behandelten Verschiebe-Befehle hinaus hat der Z80-Prozessor noch 2 Leckerbissen parat: Nibble-Rotation RLD und RRD. Diese 4-Bit-Verschiebe-Befehle sind hervorragend im Rahmen der BCD-Arithmetik (aber nicht nur dort!) einsetzbar.

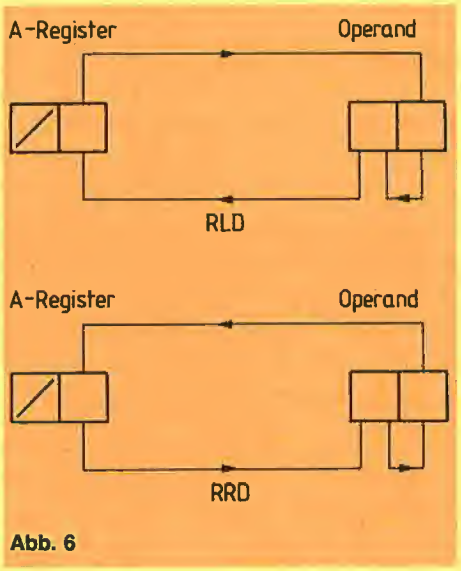


Abb. 6

Die Befehlsfolge

```
        LD HL,BCD
        LD A,(ASCII)     ; A:39 BCD:??
        RLD              ; A:3? BCD:?9
        LD A,(ASCII+1)   ; A:31 BCD:?9
        RLD              ; A:3? BCD:91
        ...
ASCII:  DB 39h,31h       ; = ASCII '91'
BCD:    DS 1
```

wandelt die ASCII-Zeichenfolge „91" in die gepackte BCD-Darstellung um. Hierzu muß HL auf die Adresse zeigen, in der der BCD-Wert abgelegt werden soll. Mit einem äußeren Loop lassen sich auch längere Strings von ASCII-Ziffern in BCD-Form umwandeln. (Versuchen Sie das doch einmal selbst! Vorschlag: DE zeigt auf ASCII, Loop-Counter in B, ...; lesen Sie noch die Erläuterungen zu DJNZ)

7.12. BIT, SET und RES

Die Befehle SET und RES setzen im Operanden ein Bit oder löschen es. Der BIT-Befehl testet den Operanden auf ein konkretes Bit und setzt das Z-Flag nach folgendem Schema:

Bit im Operand = 1 → Z-Flag = 0
Bit im Operand = 0 → Z-Flag = 1

Beispiel:

```
MUSTER:  DB 17h
         LD HL,MUSTER
         BIT 4,(HL)
```

Die Bit-Numerierung beginnt mit 0 von rechts. Im Beispiel wird das Z-Flag auf 0 gesetzt.

7.13. CALL und RET

Diese Befehle sind funktionsgleich mit JSR/RTS beim 6502. Beim Z80 kann man die Ausführung der Befehle aber noch von Bedingungen abhängig machen. Mit

```
CALL NZ,SUB
```

wird das Unterprogramm SUB nur dann durchlaufen, wenn das Ergebnis der vorausgegangenen Operation <> 0 war. Entsprechende Möglichkeiten sind auch beim RET-Befehle gegeben.

7.14. DJNZ

Dieser Spezialbefehl (Decrement and Jump if non-zero) erleichtert die Programmierung von Schleifen ungemein. Beispiele:

```
        LD B,COUNT
LOOP:   ...
        ...
        DJNZ LOOP
```

oder

```
        LD B,COUNT1
LOOP1:  PUSH BC
        LD B,COUNT2
LOOP2:  ...
        ...
        DJNZ LOOP2
        ...
        POP BC
        DJNZ LOOP1
```

Das B-Register muß stets den Schleifenzähler enthalten. DJNZ dekrementiert den Zähler und verzweigt nach der angegebenen Adresse, wenn der Zähler <> 0 ist. Der Startwert in B gibt also exakt die Anzahl der Schleifendurchläufe an. Im oberen Beispiel liegt eine einfache Schleife vor, darunter ein Beispiel mit einer Schachtelung. Der jeweils äußere Schleifenzähler wird auf dem Stack zwischengespeichert. Auf diese Weise lassen sich auch Schachtelungen mit größerer Tiefe leicht programmieren.

8. Der MACRO80-Assembler

Wie schon in der Einleitung gesagt, ist der CP/M-eigene Assembler ASM.COM ein 8080-Assembler, d.h. er verarbeitet nur Programme in 8080-Normierung. Dies wäre tragbar, wenn er zumindest makro-fähig wäre. Man könnte dann die Z80-Opcodes in Form von Makros definieren. Im Gegensatz dazu ist der MACRO80-Assembler (von Microsoft) ein Cross-Assembler, der (fast) keine Wünsche offen läßt. Quellprogramme können in 8080-, 6502-, Z80-Normierung geschrieben sein (sogar gemischt im gleichen Programm). Assembler-Listings können in Files ausgegeben werden. Cross-Reference ist möglich (Zusatzprogramm CREF80). Der Object-Code wird in Form von verschieblichen (relokativen) Modulen hergestellt. Es ist klar, daß hier nicht alle Features in voller Breite behandelt werden können. Die nachfolgende Auswahl orientiert sich an den Erfordernissen des Z80-Einsteigers, bringt aber darüber hinaus die Makro-Behandlung und (in Beispielen) weiteres Material, das Ausgangspunkt für eigene Experimente sein soll.

8.1. Aufruf des Assemblers

Die Start-Anweisung lautet:

```
MACRO80 object,list=source/switch
```

Hierin ist „object" der File-Name des erzeugten Code-Files. Die Default-Extension ist „.REL". Die Angabe „list" bezieht sich auf die Ausgabe des Assembler-Listings: Üblicherweise ist hierfür entweder „TTY:" (Ausgabe auf Bildschirm), „LST:" (Ausgabe auf Drucker) oder ein File-Name (Default-Extension „.PRN") zu schreiben. Während „object" und „list" optional sind, muß für „source" stets ein File-Name (Default-Extension „.MAC") angegeben werden. Läßt man die Laufwerksbezeichnungen weg, so wird natürlich stets das aktuelle Laufwerk angesprochen.
Ist beispielsweise das Quellprogramm TEST.MAC zu assemblieren, so bewirkt MACRO80 =TEST das Ablegen des Object-Codes in TEST.REL. Ein Listing wird nicht erzeugt. Mit MACRO80 TEST,TTY:=TEST wird Object-Code in TEST.REL und das Listing auf den Bildschirm ausgegeben. Die Sequenz MACRO80 ,TTY:=TEST unterdrückt den Object-Code.
Von den diversen Switches in der Kommando-Zeile sollen nur die folgenden behandelt werden:
/C erzeugt einen speziellen File für das CREF80-Programm.
/Z legt den Assembler beim Start auf Z80-Code fest. (Manche Versionen sind beim Start sonst auf 8080-Normierung festgelegt. Ausprobieren!)
Beispiel für die Schreibweise:

```
MACRO80 TEST,LST:=TEST/Z/C
```

8.2. Die Quellprogramm-Normierung

Die verschiedenen Felder einer Assembler-Zeile werden durch mindestens ein Leerzeichen oder Tab-Zeichen getrennt.

```
            .Z8Ø                    ; kann ggf. entfallen
            .COMMENT                * Dies ist ein Anweisungsteil
                                      mit den gebräuchlichsten Definitionen
                                      und Datentypen *
                                    ; DB = Define Byte(s)    DC = Def. Char.
                                    ; DS = Def. Storage      DW = Def. Word
            DB  "ASCII-Zeichenkette" ; statt " kann auch ' verwendet werden
            DB  "Text1","Text2"     ; mehrere Einträge möglich
            DB  'QU''OTE'           ; ergibt QU'OTE
            DB  22,27               ; Standard-Basis ist 1Ø
            DB  ØFFh,7Fh            ; Hexadezimal
            DB  "TEST"              ; ergibt 54 45 53 54
            DC  "TEST"              ; ergibt 54 45 53 D4 (high Bit!)
            DW  77FFh               ; ergibt im Speicher FF 77
BDOS        EQU ØØØ5h               ; EQU wie beim 65Ø2
WARMST      EQU BDOS-5              ; = ØØØØh
PLATZ:      DS  512                 ; reserviert 512 Bytes
NEXT        EQU $                   ; $ = Wert des Adreßzählers (PLATZ + 512)
ENDE        EQU $-1                 ; ENDE = Adresse PLATZ + 511
BYTE1       EQU LOW NEXT            ; = Low-Order-Byte der Adresse von NEXT
BYTE2       EQU HIGH NEXT           ; = High-Order-Byte
HEX         EQU 113 MOD 64          ; = 49
```

Abb. 7

Kommentare werden durch Voranstellen von „;“ gekennzeichnet (Ausnahme: spezielle COMMENT-Anweisung). Alle Labels (= Adreßdefinitionen) müssen durch „:“ abgeschlossen werden. Ausnahmen bilden nur die EQU- und ASET-Anweisungen. Beispiel:

```
DATA:  DS 5      ; reserviert
                          5 Speicherplätze
XP     EQU DATA
Z      ASET 4  ; ASET wirkt wie EQU
       ...
START: LD A,DATA+2
       ...
MOVE:  LD (DATA),A
Z      ASET 7   ; Redefin. bei ASET
                                erlaubt
               ; nicht aber bei EQU
       END
```

Mit diesem 7-Zeilen-Dummy-Programm (ohne die Zwischenraum-Punkte) können Sie Ihren MACRO80 ausprobieren, insbesondere die Z80-Voreinstellung überprüfen. Sollten Sie die 8080-Version haben, werden insbesondere die beiden LD-Zeilen als fehlerhaft gemeldet. Verwenden Sie dann beim Start von MACRO80 den /Z-Switch oder im Quellprogramm den Pseudo-Op „.Z80“. (Wollen Sie einmal doch 8080-Programme übersetzen – oder gar 6502-Programme –, so ist dem betreffenden Programmteil „.8080“ oder „.6502“ voranzustellen.)

8.3. Vereinbarungsteil

Statt vieler Worte ein selbsterklärendes Beispiel für einen Vereinbarungsteil in **Abb. 7**

Trotz gegenteiliger Aussage in der MACRO-80-Beschreibung, sollte man nur Labels mit 6 oder weniger Zeichen definieren. Probleme, insbesondere bei Makro-Ersetzungen, sind sonst unvermeidlich (zumindest bei der im Test verwendeten Version 3.36 von MACRO80).
Noch eine Bemerkung zu DW: Die Assembler-Darstellung High/Low entspricht der Registerdarstellung, während im Speicher ja nach Low/High-Muster abgelegt wird.

8.4. Die Makro-Technik

Die im Apple-6502-Bereich vorhandenen Assembler besitzen teilweise die Makro-Fähigkeit (z.B. Merlin/BIG MAC, MACRO-SOFT, ORCA, ... ; nicht aber LISA). Der MACRO80-Assembler für Z80-Code erlaubt es, Makros vorab im Programm zu definieren oder/und eine Makro-Bibliothek (oder auch mehrere) von einem Disk-File einzulesen. Der Formalismus ist hierbei folgender:

Makros programmintern

```
MACNR1  MACRO dummy1,dummy2,...
        ...
        ENDM
MACNR2  MACRO dummy1,dummy2,...
        ...
        ENDM
        ...
        (Beginn des Programms)
```

Makro-Bibliothek

```
        IF1
        INCLUDE MACROS.LIB
        ENDIF
        (Beginn des Programms)
        ...
```

Die Pseudo-Operation „MACRO“ definiert den links stehenden Namen (also MACNR1, MACNR2) als Makro, rechts stehen die formalen Parameter, für die später aktuelle Werte eingesetzt werden müssen. Die Anweisung „ENDM“ schließt die jeweilige Makro-Definition ab. Die Pseudo-Operation „INCLUDE“ benennt den rechts stehenden File-Namen als Makro-Bibliothek. Dieser File wird während der Assemblierung eingelesen – und zwar normalerweise in jedem Assembler-Pass.

Die oben stehenden Anweisungen IF1 .... ENDIF beschränken diesen Vorgang auf den ersten Assembler-Pass. Es bedeutet „IF1“ nämlich „falls im 1. Pass“. (Dies ist ein Beispiel für eine bedingte Assemblierung, hierzu später mehr.)

Diese Methodik hat noch zwei weitere Vorteile: Sie vermeidet die zeitaufwendige Redefiniton der Makros im 2. Pass und unterdrückt die Auflistung der (evtl. langen) Makro-Bibliothek im Assembler-Listing (alle Listings werden im 2. Pass erstellt!).

Nun zu praktischen Beispielen für Makros. Betrachten wir den Fall, daß der Inhalt von SZ1 nach SZ2 transferiert werden soll. Normales Coding wäre z.B.

```
        LD A,(SZ1)
        LD (SZ2),A
```

Die Makro-Definition

```
MOVE MACRO ZIEL,QUELLE
        LD A,(QUELLE)
        LD (ZIEL),A
        ENDM
```

liefert mit dem Aufruf „MOVE SZ2,SZ1“ das gleiche Resultat. Weitere einfache Makro-Beispiele enthält die **Makro-Liste** am Ende des Artikels. Zum Teil wird dabei die LOCAL-Anweisung verwendet. Sie bewirkt, daß die folgenden Labels nur innerhalb des betreffenden Makros bekannt sind.

Im Rahmen der Makro-Technik sind auch die bedingte Assemblierung und die Repeat-Funktion (Wiederholung von Assembler-Zeilen) von Bedeutung. Hierzu ein Beispiel, das Ausgangspunkt für weitere eigene Experimente sein soll. Problemstellung: Ab Adresse OP1 stehen N Bytes, die eine Binär-Zahl darstellen (beginnend mit den Low-Order-Bytes); ab OP2 stehen weitere N Bytes einer 2. Zahl; diese Zahl soll zur ersten addiert werden („lange Addition“); es soll möglichst wenig Rechenzeit verbraucht werden. Wegen der letzten Forderung verbieten sich Schleifen, viel-

mehr muß der Source-Code ausgeschrieben werden. Bei der Programm-Entwicklung sei aber die Länge N nicht bekannt.

Eine Lösung zeigt **Abb. 8**

Mit dem Pseudo-Op „IFB" (= IF BLANK) wird zunächst abgefragt, ob N im Aufruf angegeben wurde – ersatzweise wird NN zu 1 gesetzt, im anderen Fall zu N. (Hier muß eine neue Variable NN eingeführt werden, da N auch im Label-Feld durch den aktuellen Wert ersetzt würde.) Der Zähler X wird zu 0 gesetzt. Der Anweisungsblock zwischen „REPT" (= REPEAT) und dem ersten ENDM wird nun NN mal bei der Assemblierung (!) durchlaufen. (Das zweite „ENDM" unten schließt die Makro-Definition ab.) Die Variable X wird nach jedem Schritt um 1 erhöht. Im Anweisungsblock wird mit IFE/ELSE/ENDIF der Fall X =0 von X <> 0 getrennt. (Liest sich fast wie ein Pascal-Programm.) „IFE X" bedeutet „falls X = 0".

Da MACRO80 die Makros im Assembler-Listing ausschreibt, ist es leicht möglich, sich bei eigenen Experimenten von der Richtigkeit des Ansatzes zu überzeugen. Eine Schachtelung von Makros (Nesting Macros) ist erlaubt. Ein kurzes Beispiel hierzu: Es sei ein Makro DISPLAY vorhanden zur Ausgabe von Strings auf den Bildschirm (die Startadresse des Strings sei der Parameter im Makro-Aufruf). Es soll ein Makro FMELD (Fehlermeldung) geschrieben werden, das abhängig von einer Fehlernummer K= 1, 2, 3, ... einen Text ab Adresse TEXT1, TEXT2, TEXT3, ... ausgibt.

```
FMELD MACRO K
      DISPLAY TEXT&K
      ENDM
```

Der Parameter K wird mit dem Namen TEXT durch das Zeichen & verkettet. Aus FMELD 8 macht der Assembler DISPLAY TEXT8. Das Makro DISPLAY selbst kann intern noch auf weitere Makros zurückgreifen.

Die wenigen hier dargestellten Möglichkeiten stellen nur einen Ausschnitt dar, in jedem Fall lohnt es sich, die MACRO80-Beschreibung durchzuarbeiten und an eigenen Beispielen auszuprobieren.

9. Der Linker L80

Die von MACRO80 übersetzten Programme werden in Form von verschieblichen (relokativen) Modulen ausgegeben. Diese müssen noch vom Linker behandelt werden, bevor man startfähige Programme erhält. Auf die speziellen Features von MACRO80/L80 in diesem Bereich soll nicht weiter eingegangen werden. (Die sich hier bietenden Möglichkeiten stehen kaum zurück hinter den Eigenschaften von Assemblern/Linkern bei großen Mainframes.) Der Standard-Aufruf für L80 lautet:

L80 modul,comfile/N/E

Hierin ist „modul" der Name des von MACRO80 ausgegebenen REL-Files, „comfile" der File-Name für den zu erzeugenden COM-File. („.REL" und „.COM" brauchen nicht angegeben zu werden.) Die Switches /N und /E bedeuten: Abspeichern des COM-Files auf Disk und Beenden des Linkers. Als Startadresse nimmt L80 normalerweise 0100h, es sei denn, in der END-Anweisung des Assembler-Programms ist ein Adreßausdruck angegeben – z.B. END LAB7. Dann setzt L80 vor das Programm einen JP LAB7. Die erste Assembler-Zeile muß also nicht unbedingt die erste auszuführende Anweisung beinhalten. Insbesondere kann komplikationslos der Vereinbarungsteil vor dem Anweisungsteil liegen.

Ausgeschrieben:

```
LD   HL,OP2
LD   A,(OP1)
ADD  A,(HL)
LD   (OP1),A

INC  HL
LD   A,(OP1+1)
ADC  A,(HL)
LD   (OP1+1),A

...

INC  HL
LD   A,(OP1+N-1)
ADC  A,(HL)
LD   (OP1+N-1),A
```

Makro-Definition:

```
LADD  MACRO  OP1,OP2,N
      IFB    <N>
NN    ASET   1
      ELSE
NN    ASET   N
      ENDIF
X     ASET   Ø
      REPT   NN
      IFE    X
      LD     HL,OP2
      LD     A,(OP1)
      ADD    A,(HL)
      LD     (OP1),A
      ELSE
      INC    HL
      LD     A,(OP1+X)
      ADC    A,(HL)
      LD     (OP1+X),A
      ENDIF
X     ASET   X+1
      ENDM
      ENDM
```

Abb. 8

10. SUBMIT-File

Am einfachsten ist es, sich einen SUBMIT-File (z.B. mit dem Zeileneditor ED.COM) herzustellen, der sowohl den Assembler MACRO80 als auch den Linker L80 aufruft:

```
MACRO80 %1,%1=%1/Z
TYPE %1.PRN
L80 %1,%1/N/E
ERA %1.REL
```

Je nach Bedarf könnte man noch ZSID %1.COM oder einfach %1 anfügen (Debugger-Aufruf oder direkter Programm-Start). Speichert man diese Zeilen in dem File RUN.SUB ab, so kann man beispielsweise mit dem Aufruf

SUBMIT RUN TEST

das Assembler-Programm TEST.MAC assemblieren und linken. Das Listing wird auf dem Bildschirm ausgegeben, der überflüssige REL-File gelöscht.
Wichtig: SUBMIT kann nur dann aufgerufen werden, wenn A: das aktuelle Laufwerk ist. Arbeiten Sie z.B. mit 2 Laufwerken und haben B: als aktuelles Laufwerk, so funktioniert SUBMIT nicht (ausprobieren!). Grund (vgl. die Erläuterungen unter 11. und 12.): Der für SUBMIT wichtige File $$$.SUB wird auf das aktuelle Laufwerk gelegt, müßte aber stets auf A: liegen (wovon CP/M den CCP nachlädt). Mit folgender Änderung klappt es immer. Geben Sie der Reihe nach ein:

```
DDT SUBMIT.COM
S5BB <Return> 01 <Return> . <Return>
↑C
SAVE 5 SUBMIT.COM
```

Vielleicht ändern Sie für die gepatchte Version den Namen ab. Die Änderung 01 (gegenüber 00) in 5BB setzt im FCB den Drive-Parameter auf A:.

11. Speicherverwaltung unter CP/M 2.2

Die folgenden Erläuterungen beziehen sich stets auf einen 64K-Speicherausbau

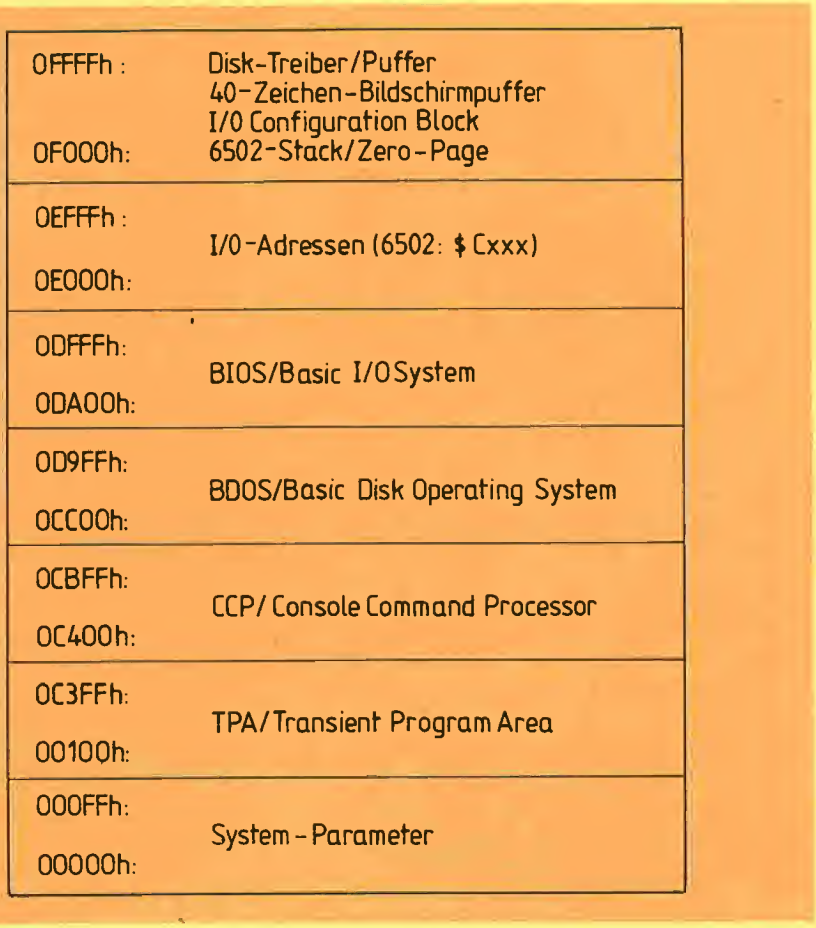


Abb. 9

(48K + 16K-Karte). In diesem Fall ist die Speicheraufteilung unter CP/M 2.2 wie in **Abb. 9** gegliedert.

6502-Experten werden sich jetzt wundern, wieso im Bereich Cxxx System-Programme liegen können, da dieser Adreßbereich doch u.a. für die ROM-Routinen der Controller-Karten reserviert ist. Die Lösung des Rätsels ist eine hardware-mäßig durchgeführte Adreßcodierung. Beim 6502 ist der nutzbare Speicherplatz durch den CX-Bereich unterbrochen. Ohne DOS 3.3 stehen dem Benutzer z. B. die Bereiche $800-$BFFF und (RAM-Karte) $D000-$FFFF (mit 2 Banks) zur freien Verfügung. Um die Unterbrechung zu beseitigen, wird bei der Z80-Softcard eine Adressenumrechnung nach folgenden Schema vorgenommen:

6502-Bereich	->	Z80-Bereich
$1000-$BFFF	(- 1000)	00000h-0AFFFh
$D000-$FFFF	(- 2000)	0B000h-0DFFFh
$C000-$CFFF	(+ 2000)	0E000h-0EFFFh
$0000-$0FFF	(- 1000)	0F000h-0FFFFh

Abb. 10: Adreßcodierung

Das hat zur Folge, daß nunmehr unter CP/M lückenlos der Bereich 0000h-DFFFh genutzt werden kann. Die Umrechnungstabelle muß allerdings nur dann beachtet werden, wenn
a) Kopplungen zwischen 6502- und Z80-Prozessor vorgesehen sind,
b) ROM-Routinen des 6502 auch unter Z80 benutzt werden sollen.
Im Normalfall kann die Adreßcodierung getrost vergessen werden. Gewöhnungsbedürftig ist und bleibt allerdings, daß die vertrauten $Cxxx-Adressen hier mit „E" beginnen.
Nun zur Speicherverwaltung selbst: Ganz grob (!) läßt sich sagen, daß der CCP dem Command-Interpreter unter DOS 3.3 entspricht, BDOS dem File-Manager, BIOS der RWTS (und anderen I/O-Routinen). Die TPA ist der für den Anwender zur Verfügung stehende Speicherbereich. Wesentlich für den Assembler-Programmierer ist nun, daß der CCP von Programmen überschrieben werden darf, sofern am Ende des Programms ein Warmstart (JP 0000h) ausgeführt wird. CP/M lädt dann den CCP von der Diskette nach. Bleibt jedoch der CCP intakt, und wird das eigene Programm über RET verlassen, braucht nichts nachgeladen zu werden. In diesem Fall stehen dem Benutzer durchgehend etwa 49K zur Verfügung, im anderen Fall 2K mehr. (Vergleich: unter DOS 3.3 nur ca. 37K in einem Stück.) Beim CP/M 2.23 (60K) wird auch noch die Bank 1 der RAM-Karte für das System benutzt, so daß hier die TPA noch entsprechend größer ist.

12. I/O-Funktionen unter CP/M

Das Betriebssystem besitzt 37 von „außen" ansprechbare Einzelfunktionen für die I/O-Abwicklung. Diese Funktionen sind durch Nummern von 0-36 (00h-24h) gekennzeichnet. Hiervon wird in diesem Aufsatz nur ein Teil ausführlich besprochen, und zwar Console-Eingabe/Ausgabe einzelner Zeichen/Strings, sowie File-I/O vom sequentiellen Typ. Da die Prozedur fast überall gleich ist, kann für alle anderen Funktionen auf das CP/M Reference Manual verwiesen werden. Die prinzipielle Vorgehensweise ist dabei folgende: In das C-Register wird die Funktionsnummer geladen und danach BDOS aufgerufen (BDOS-Vektor steht bei 0005h). Bei manchen Funktionen muß vorher noch im DE-Register eine Adresse bereitgestellt werden (z.B. Puffer-Adresse). Das ist alles! Da bei der Ausführung der BDOS-Funktionen Register zerstört werden, muß der Anwender ggf. seine Register vor dem BDOS-Aufruf retten.

12.1. Einfache I/O-Funktionen (Console, Drucker)

Beispiel 1: Einlesen eines Zeichens von Tastatur. Durch

```
LD    C,1    ; 1 = Console Input
CALL BDOS ; BDOS = 0005h
```

wird ein Zeichen im A-Register übergeben (Low-ASCII, Bit 7 = 0). Die BDOS-Routine wartet dabei auf das Betätigen der Tastatur und verzögert die weitere Programmausführung.
Beispiel 2: Einlesen einer Zeile von Tastatur. Hierzu ist

```
LD (BUFFER),MAX  ; MAX = max.
                                Zeilenlänge
LD C,10                  ; 10 = Console String
                                Input
LD DE,BUFFER      ; String-Puffer
CALL BDOS
```

anzugeben. BDOS erwartet, daß die erste Position im Puffer die maximale vom Benutzer zugelassene Zeilenlänge enthält. Das garantiert die 1. Assembler-Zeile. DE enthält die BUFFER-Adresse. Nach Ausführung des BDOS-Aufrufs legt die Input-Routine in der 2. Position in BUFFER die tatsächlich eingegebene Zeichenzahl ab, danach erst die eigentlichen Daten. BUFFER muß also 2 Bytes länger sein als der erwartete String. Die Zeilenende-Zeichen CR/LF (Carriage Return/Line Feed) werden nicht in den BUFFER übergeben – ebensowenig wie die Edit-Zeichen (z.B.

Ctrl-H). Zeichen erscheinen stets in Low-ASCII.
Beispiel 3: String-Ausgabe auf Bildschirm. Der auszugebende String muß durch ein „$" abgeschlossen sein. Davor sind ggf. noch CR/LF zu setzen – wie im folgenden Fall:

```
        LD C,9 ; 9 = Display String
        LD DE,STRING
        CALL BDOS
        ...
STRING: ASC "Testzeile"
        DB 0Ah,0Dh,"$"
```

Der auszugebende String darf natürlich kein „$" als Datenzeichen enthalten. In einem solchen Fall muß der String zeichenweise ausgegeben werden; hierzu dient das Makro DISPC in der **Makro-Liste** im Anhang. Hiermit könnte der Leser ein Makro definieren, das z. B. das gesetzte Bit 7 als Ende-Zeichen betrachtet. Der String würde dann mit DC definiert.
Die Makro-Liste enthält noch weitere elementare I/O-Funktionen, z. B. für die Drucker-Ansteuerung.

12.2. Sequentielles File-I/O

Maßgebend ist hierfür der sogenannte FCB = File Control Block (in der Funktion vergleichbar mit der File-Manager-Parameter-Liste unter DOS). Dieser Block hat eine Länge von 36 Bytes und ist nach dem in **Abb. 11** wiedergegebenen Schema vorzubelegen.

Ähnlich wie unter DOS, kann man mit einem eigenen FCB arbeiten oder den System-FCB benutzen (liegt bei Adresse 005Ch-007Fh). Arbeitet man mit mehr als einem File simultan, so sollte man stets mehrere und damit selbst definierte FCBs vorsehen. Ein spezielles Makro für den Vereinbarungsteil (s. Makro-Liste) erleichtert die Definition/Initialisierung des FCB. Die jeweilige FCB-Adresse ist stets im DE-Register bereitzustellen.
Nun zum Problem des Daten-Puffers. Hier ist zunächst festzustellen, daß CP/M den Datentransfer mit einer festen Record-Länge von 128 Bytes abwickelt. Daten können somit nur 128-byte-weise gelesen/geschrieben werden. Der System-Puffer liegt bei 0080h-00FFh. Entscheidet man sich für einen eigenen Daten-Puffer, so muß das dem System mitgeteilt werden. Hierfür existiert eine separate BDOS-Funktion, die wie folgt abgerufen wird:

```
        LD C,26 ; 26 = Set DMA
                       Address
        LD DE,DATABUF
        CALL BDOS
        ...
DATABUF: DS 80h
```

Die neue Puffer-Adresse kann nur durch erneuten Aufruf der Funktion 26, durch Warmstart oder die BDOS-Funktion 0 (DISK RESET) wieder verändert werden.

12.2.1. File-Eröffnung

Bei der Eröffnung eines Disk-Files muß man unterscheiden zwischen Neueröffnung (MAKE FILE) und einer Wiedereröffnung eines schon bestehenden Files (OPEN FILE). Jetzt Achtung: Bei einem MAKE FILE kontrolliert BDOS nicht (!), ob ein File gleichen Namens schon vorhanden ist, sondern legt u.U. einen namensgleichen File an (evtl. File vorher löschen). BDOS übergibt nach Ausführung der Eröffnungen einen Status-Code im A-Register: 0-3 bedeutet „alles O.K.", OFFh heißt Fehler (MAKE FILE: Disk Full, OPEN FILE: File not found).

12.2.2. Record lesen/schreiben

Für das Lesen und Schreiben sind die BDOS-Funktionen 20 und 21 zuständig. Ist nach Ausführung dieser Funktionen das A-Register <> 0, so ist beim Schreiben „Disk Full" aufgetreten, beim Lesen das File-Ende erreicht worden. Die Makros READRC und WRITRC in der Makro-Liste berücksichtigen diese Fälle. Schreibt man Daten in einen bestehenden File und überschreitet das Dateiende, so wird zwanglos die Datei vom System erweitert. Eine separate APPEND-Funktion ist überflüssig. Bei ASCII-Dateien ist es üblich, das Dateiende (sofern innerhalb eines Records) durch CTRL-Z (1Ah) abzuschließen. So verfährt der ED.COM und auch WORDSTAR.

12.2.3. Datei schließen

Am Ende der Verarbeitung ist die CLOSE-Funktion auszuführen (BDOS-Funktion Nr.16)
Das Beispiel in **Abb. 12** simuliert das TYPE-Kommando von CP/M und zeigt den Umgang mit den verschiedenen BDOS-Funktionen. Es wird angenommen, daß der File-Name bereits im unten definierten FCB eingetragen wurde oder von vorneherein feststeht. Die Makro-Namen beziehen sich auf die Makro-Liste.

Versuchen Sie doch einmal, auf ähnliche Art und Weise ein simples File-Copy-Programm zu erstellen. (Zuerst mit fest vorgegebenen File-Namen. Programmieren Sie dann im nächsten Schritt ein kleines Menü für die Eingabe der File-Namen und der zugehörigen Laufwerke. Vielleicht eine Pause für Diskettenwechsel?)
Noch eine wichtige Bemerkung zum Debuggen solcher Programme, die File-I/O beinhalten: Die Debugger setzen beim Start den Stackpointer auf 0100H – also auf das Ende des System-I/O-Puffers. Wird dieser benutzt, muß in jedem Fall ein eigener Stack-Bereich verwendet werden.

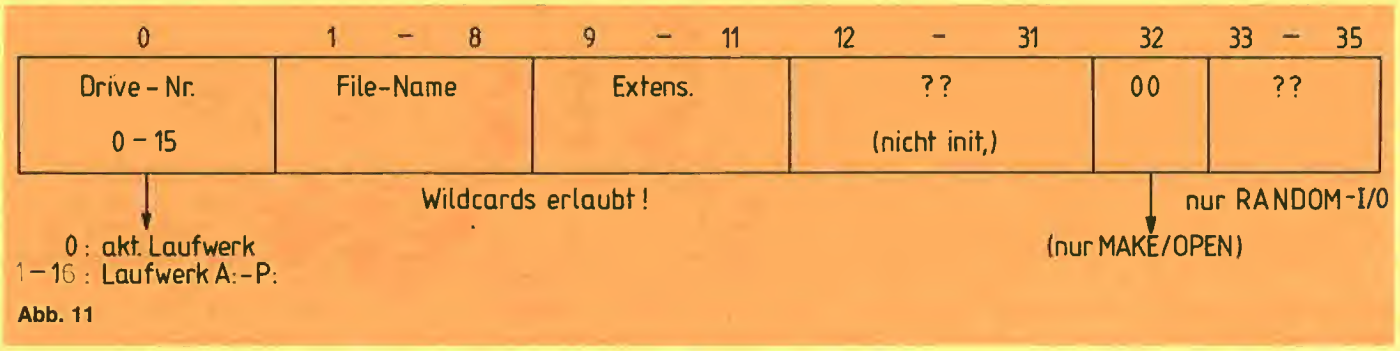


Abb. 11

```
          ...
          SETBUF BUFFER          ; eigener Puffer für File-I/O
          OPEN   FCB,NOTFND      ; File eröffnen (vorhanden?)
LOOP:     READRC FCB,EOF         ; Record lesen
          CALL   DISPLAY         ; Record -> Bildschirm
          JR     LOOP            ; nächster Record
EOF:      CLOSE  FCB             ; End-of-File: File schließen
          JP     0               ; Programmende
NOTFND:   DISPS  MESS            ; File nicht vorhanden/melden
          JP     0

DISPLAY:  LD     HL,BUFFER
L:        LD     A,(HL)          ; Zeichen aus Puffer
          CP     1Ah             ; Ctrl-Z ?
          JR     Z,EOF           ; End-of-File erreicht
          PUSH   HL              ; HL vor BDOS retten
          LD     E,A             ; umladen für DISPC
          DISPC

          POP    HL
          INC    HL              ; Zeiger: + 1
          PUSH   HL              ; HL sichern
          AND    A               ; lösche Carry
          LD     BC,BUFFER+0080h
          SBC    HL,BC           ; HL = Pufferende + 1?
          POP    HL              ; HL restaurieren
          JR     NZ,L            ; nächstes Zeichen, falls <> 0
          RET                    ; Pufferende erreicht

FCB:      FCB    "A","FILENAME","EXT"  ; File-Control-Block
                                       ; für "A:FILENAME.EXT"
BUFFER:   DS     80h             ; I/O-Puffer
MESS:     CRLF                   ; CR LF
          DB     "File nicht vorhanden"
          DELIM                  ; CR LF $
          END
```

Abb. 12

Z80-Makro-Liste (unter CP/M 2.2)

```
BDOS      EQU       0005H

READC     MACRO
; Zeichen/Tastatur -> A-Register
          LD        C,1
          CALL      BDOS
          ENDM


READS     MACRO     BUFFER,MAXZ
; Zeile/Tastatur -> BUFFER
; BUFFER + 0: max. Zeichenzahl MAXZ
; BUFFER + 1: tatsächl. Zeichenzahl
; BUFFER + 2: Eingabestring ohne CR/LF
          LD        A,MAXZ
          LD        (BUFFER),A
          LD        C,10
          LD        DE,BUFFER
          CALL      BDOS
          ENDM


DISPC     MACRO
; Zeichen/E-Register -> Bildschirm
          LD        C,2
          CALL      BDOS
          ENDM


DISPS     MACRO     BUFFER
; String begrenzt durch $ -> Bildschirm
; BUFFER: auszugebende Zeichenkette
          LD        C,9
          LD        DE,BUFFER
          CALL      BDOS
          ENDM


CRLF      MACRO
; setzt Characters CR/LF ein
          DB        0DH,0AH   ; CR/LF
          ENDM


DELIM     MACRO
; schließt String durch CR/LF/$ ab
          CRLF                ; CR/LF
          DB        "$"
          ENDM

PRINTC    MACRO
; Zeichen/E-Register -> Drucker/LST:
          LD        C,5
          CALL      BDOS
          ENDM
```

```
PRINTS    MACRO     BUFFER
; String bis $ -> Drucker/LST:
; BUFFER: zu druckender String
; IX-Register wird benutzt!
          LOCAL     LOOP,EXIT
          LD        IX,BUFFER
LOOP:     LD        E,(IX)
          INC       IX
          LD        A,"$"
          CP        E
          JR        Z,EXIT
          PRINTC
          JR        LOOP
EXIT:
          ENDM


FCB       MACRO     DRV,FILE,EXT
; File-Control-Block
; Parameter in "..." angeben
; FILE: File-Name
; EXT:  File-Erweiterung
; DRV:  Drive (leer = akt. Drive)
          LOCAL     FL,FLE,X,XE
          IFB       <DRV>
          DB        0
          ELSE
          DB        DRV-64
          ENDIF
FL:
          DB        "        "
FLE:
          ORG       FL
          DB        FILE
          ORG       FLE
X:
          DB        "   "
XE:
          ORG       X
          DB        EXT
          ORG       XE
          DB        0,0,0,0,0,0,0,0,0,0
          DB        0,0,0,0,0,0,0,0,0,0
          DB        0,0,0,0
          ENDM

SETBUF    MACRO     BUFFER
; setzt Buffer für File-I/O
; BUFFER: Daten-Puffer für Record
;         (128 Bytes)
          LD        C,26
          LD        DE,BUFFER
          CALL      BDOS
          ENDM
```

```
MAKE      MACRO     FCBADR,ERROR
; File neu eröffnen , ERROR optional
; FCBADR: Adresse des FCB
          XOR       A
          LD        (FCBADR+32),A
          LD        C,22
          LD        DE,FCBADR
          CALL      BDOS
          IFNB      <ERROR>
          CP        4
          JP        NC,ERROR  ; Disk Full
; ERROR: Ausgang bei 'Disk Full'
          ENDIF
          ENDM


OPEN      MACRO     FCBADR,NOTFND
; vorhand. File eröffnen, NOTFND opt.
; FCBADR: Adresse des FCB
          XOR       A
          LD        (FCBADR+32),A
          LD        C,15
          LD        DE,FCBADR
          CALL      BDOS
          IFNB      <NOTFND>
          CP        4
          JP        NC,NOTFND
; NOTFND: Ausgang bei 'File not found'
          ENDIF
          ENDM


READRC    MACRO     FCBADR,EOF
; Read sequ. Record, EOF optional
; FCBADR: Adresse des FCB
          LD        C,20
          LD        DE,FCBADR
          CALL      BDOS
          IFNB      <EOF>
          CP        0
          JP        NZ,EOF
; EOF: Ausgang bei 'End of File'
          ENDIF
          ENDM


WRITRC    MACRO     FCBADR,ERROR
; Write sequ. Record, ERROR optional
; FCBADR: Adresse des FCB
          LD        C,21
          LD        DE,FCBADR
          CALL      BDOS
```

```
          IFNB      <ERROR>
          CP        Ø
          JP        NZ,ERROR
; ERROR: Ausgang bei 'Disk Full'
          ENDIF
          ENDM


CLOSE     MACRO     FCBADR,NOTFND
; File schließen , NOTFND optional
; FCBADR: Adresse des FCB
          LD        C,16
          LD        DE,FCBADR
          CALL      BDOS
          IFNB      <NOTFND>
          CP        4
          JP        NC,NOTFND
; NOTFND: Ausgang bei 'File not found'
          ENDIF
          ENDM

DELETE    MACRO     FCBADR,NOTFND
; File löschen, NOTFND optional
; FCBADR: Adresse des FCB
          LD        C,19
          LD        DE,FCBADR
          CALL      BDOS
          IFNB      <NOTFND>
          CP        4
          JP        NC,NOTFND
; NOTFND: Ausgang bei 'File not found'
          ENDIF
          ENDM

SAVX      MACRO
; Register retten -> Cache
          EX        AF,AF'
          EXX
          ENDM

LADX      MACRO
; Register laden <- Cache
          SAVX
          ENDM

SAVR      MACRO     X
; Register auf Stack retten
; X: falls nicht leer, auch
;    Indexregister retten
          PUSH      HL
          PUSH      DE
          PUSH      BC
          PUSH      AF
          IFNB      <X>
          PUSH      IY
          PUSH      IX
          ENDIF
          ENDM

LADR      MACRO     X
; Register von Stack laden
; X: falls nicht leer, auch
;    Indexregister laden
          IFNB      <X>
          POP       IX
          POP       IY
          ENDIF
          POP       AF
          POP       BC
          POP       DE
          POP       HL
          ENDM

; bedingte Sprünge bei Größenvergleich
; 8-Bit-Zahlen (ohne Vorzeichen)
; ADR: Sprungadresse bei erfüllter
;      Bedingung

JLT       MACRO     ADR  ; JP, IF <
          JP        C,ADR
          ENDM
JLE       MACRO     ADR  ; JP, IF <=
          JP        C,ADR
          JP        Z,ADR
          ENDM
JE        MACRO     ADR  ; JP, IF =
          JP        Z,ADR
          ENDM
JNE       MACRO     ADR  ; JP, IF <>
          JP        NZ,ADR
          ENDM
JGE       MACRO     ADR  ; JP, IF >=
          JP        NC,ADR
          ENDM
JGT       MACRO     ADR  ; JP, IF >
          LOCAL     EXIT
          JR        C,EXIT
          JR        Z,EXIT
          JP        ADR
EXIT:
          ENDM

MOVE      MACRO     DEST,SOURCE ; Byte
          LD        A,(SOURCE)
          LD        (DEST),A
          ENDM

MOVE2     MACRO     DEST,SOURCE  ; Wort
          LD        HL,(SOURCE)
          LD        (DEST),HL
          ENDM

MOVBL     MACRO     TAB2,TAB1,N
; N Bytes von TAB1 -> TAB2 (Inkrement)
          LD        BC,N
          LD        HL,TAB1
          LD        DE,TAB2
          LDIR
          ENDM

FILL      MACRO     TAB,N,VALUE
; N Bytes ab TAB mit 'VALUE' füllen
          LOCAL     LOOP
          LD        HL,TAB
          LD        B,N
          LD        A,VALUE
LOOP:
          LD        (HL),A
          INC       HL
          DJNZ      LOOP
          ENDM

SEARCH    MACRO     TAB,N,VALUE,FOUND,NOTFND
; N Bytes ab TAB nach 'VALUE' durchsuchen
; falls vorh. -> FOUND, sonst -> NOTFND
; FOUND/NOTFND optional
; HL zeigt ggf. auf gefundenes Byte + 1
          LD        HL,TAB
          LD        BC,N
          LD        A,VALUE
          CPIR
          IFNB      <FOUND>
          JP        PE,FOUND
          ENDIF
          IFNB      <NOTFND>
          JP        PO,NOTFND
          ENDIF
          ENDM
```

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

Bei der Version 2.0 der MMU's sind die Utilities teilweise so umgeschrieben worden, daß sie sowohl unter DOS 3.3 als auch unter ProDOS lauffähig sind. Da dies nicht immer möglich war, sind zusätzlich zu den reinen DOS-Hilfsprogrammen, speziell den RAM-Disk-Drivern, einige reine ProDOS-Utilities aufgenommen worden. 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.

Im einzelnen umfaßt „MMU 2.0"

- Drei RAM-Disk-Driver für DOS 3.3: „INIT 62" benutzt nur die 64K-Karte als RAM-Disk, „INIT 78" benutzt zusätzlich die Motherboard-LC als RAM-Karte und „DOSMOVER. INIT 62" gilt für den Fall, daß sich das DOS selbst in der Motherboard-LC befindet.
- Eine sehr nützliche Pseudo-Coprocessor-Utility, die das Hin- und Herschalten zwischen zwei Programm-Modulen ermöglicht, von denen sich das eine Modul auf der 64K-Karte befindet.
- Zwei schnelle Kopierprogramme (für DOS 3.3 und ProDOS).
- Mehrere Move-Programme zum Verschieben von Daten auf die 64K-Karte sowie auf die Language Card und umgekehrt.
- Mehrere Hilfsprogramme zum Untersuchen und Löschen bestimmter Speicherbereiche der 64K-Karte und der LC, zur Ermittlung des Softswitch-Status usw.
- Zwei Simulator-Programme zum Simulieren von Apple II und Apple II Plus auf dem Apple IIe.

Gerätevoraussetzung: Apple IIe mit 64K-Karte oder IIc

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

Z80-Operationstabelle

Bei der Taktfrequenz von ca. 2 MHz dauert ein Speicherzyklus ca. 500 nsec (= 0.5 × 10^{-6} sec). Bei den Speicherzyklen sind z. T. „von-bis"-Angaben vorhanden, hier ist die Ausführungszeit von Registerinhalten abhängig. Sofern dem Befehl ein Adreß- oder 16-Bit-Immediatewert folgt, ist dieser im Low/High-Format anzuhängen. Bei den Flags bedeuten: X = Bit kann verändert werden; – = Bit wird nicht geändert; 0 = Bit wird zu 0 gesetzt; 1 = Bit wird zu 1 gesetzt; ? = Bit ist unbestimmt.

Unter der Spaltenüberschrift 8080-M sind die 8080-Mnemonics eingetragen – sofern entsprechende Befehle beim Intel-8080 vorhanden sind. Die Mnemonics sollen allerdings nur als grobe Stütze dienen. Die Schreibweise der 8080-Befehle weicht noch in anderen Details von der Z80-Normierung ab.

In den Tabellen fehlen die Z80-Befehle, die sich auf die Port- und Interrupt-Programmierung beziehen. Grund: Die Z80-Port-Leitungen sind bei der Softcard nicht herausgeführt. In der Standard-Betriebsart sind auch die Interrupt-Befehle nicht ausnutzbar. Beim Apple II unter CP/M kann man diese beiden Befehlsgruppen zunächst vergessen.

Befehle mit 16-Bit-Struktur:

Operation	Operand → Funktion	AF	BC Op/Cy	DE Op/Cy	HL Op/Cy	SP Op/Cy	IX Op/Cy	IY Op/Cy	(Adr) Op/Cy	Imm/16 Op/Cy	Impliz. Op/Cy	Flags SZ–H–PNC	8080-M
Arithmetik													
ADC HL,	HL+Operand+Carry		ED4A/15	ED5A/15	ED6A/15	ED7A/15						XX?X?X0X	
ADD HL,	HL+Operand		09/11	19/11	29/11	39/11						––?X?–0X	DAD
ADD IX,	IX+Operand		DD09/15	DD19/15		DD39/15	DD29/15					––?X?–0X	
ADD IY,	IY+ Operand		FD09/15	FD19/15		FD39/15		FD29/15				––?X?–0X	
SBC HL,	HL– Operand–Carry		ED42/15	ED52/15	ED62/15	ED72/15						XX?X?X1X	
DEC	Dekrementieren		0B/6	1B/6	2B/6	3B/6	DD2B/10	FD2B/10				––?–?–––	DCX
INC	Inkrementieren		03/6	13/6	23/6	33/6	DD23/10	FD23/10				––?–?–––	INX
Transfer-Befehle													
EX DE,	DE ↔ Operand				EB/4							––?–?–––	XCHG
EX (SP),	(SP) ↔ Operand				E3/19		DDE3/23	FDE3/23				––?–?–––	XTHL
EX AF, AF '	AF ↔ AF										08/4	––?–?–––	
EXX	Austausch/Cache										D9/4	––?–?–––	
LD BC,	BC ← Operand								ED4B/20	01/10		––?–?–––	LXI
LD DE,	DE ← Operand								ED5B/20	11/10		––?–?–––	LXI
LD HL,	HL ← Operand								ED6B/20	21/10		––?–?–––	LHLD/LXI
"									2A[1]/16				
LD SP,	SP ← Operand				F9/6		DDF9/10	FDF9/10	ED7B/20	31/10		––?–?–––	LXI/SPHL
LD (Adr),	Adr ← Operand		ED43/20	ED53/20	ED63/20	ED73/20	DD22/20	FD22/20				––?–?–––	SHLD
"					22[1]/16								
LD IX,	IX ← Operand								DD2A/20	DD21/14		––?–?–––	
LD IY,	IY ← Operand								FD2A/20	FD21/14		––?–?–––	
POP	Stack → Operand	F1/10	C1/10	D1/10	E1/10		DDE1/14	FDE1/14				––?–?–––	POP
PUSH	Operand →Stack	F5/11	C5/11	D5/11	E5/11		DDE5/15	FDE5/15				––?–?–––	PUSH

[1] 2. Operationscode für den gleichen Befehl

Bedingte/Unbedingte Programmverzweigung:

Operation	Bedingung → Funktion	ohne Bed. Op/Cy	NZ Op/Cy	Z Op/Cy	NC Op/Cy	C Op/Cy	PO Op/Cy	PE Op/Cy	P Op/Cy	M Op/Cy	Flags SZ–H–PNC	8080-M
Jump-Call-Return												
CALL	Unterprogrammsprung[3]	CD/17	C4/17-10	CC/17-10	D4/17-10	DC/17-10	E4/17-10	EC/17-10	F4/17-10	FC/17-10	––––?–?––	CALL/Cbed[2]
DJNZ	Dekrem./Sprung NZ[3]		10/13-8								––––?–?––	
JP	Sprungbefehl	C3/10	C2/10	CA/10	D2/10	DA/10	E2/10	EA/10	F2/10	FA/10	––––?–?––	JMP/Jbed[2]
JP (HL)	indirekt. Sprung	E9/4									––––?–?––	PCHL
JP (IX)	indirekt. Sprung	DDE9/8									––––?–?––	
JP (IY)	indirekt. Sprung	FDE9/8									––––?–?––	
JR	Sprung/kurz[3]	18/12	20/12-7	28/12-7	30/12-7	38/12-7					––––?–?––	
RET	Rückspr. aus UP[3]	C9/10	C0/11-5	C8/11-5	D0/11-5	D8/11-5	E0/11-5	E8/11-5	F0/11-5	F8/11-5	––––?–?––	RET/Rbed[2]

[2] „bed" ist einer der Schlüssel NZ, Z, NC, C, . . .
[3] Bei den Cycles ist z.B. in 17-10 die erste Zahl bei erfüllter Bedingung, die zweite Zahl bei nicht erfüllter Bedingung gültig.

Befehle mit 8-Bit-Struktur:

Um die Tabelle nicht zu breit werden zu lassen, sind Befehle mit mehr als einem Byte Operationscode nach einem besonderen Schema aufgeführt. Beispiel:

1.) CPD
a) Spaltenüberschrift ED
b) Tabelleneintrag A9/16
Insgesamt: ED A9. Ausführungszeit des Befehls: 16 Speicherzyklen

2.) ADC A, (IX+d)
a) Spaltenüberschrift DDxx+
b) Tabelleneintrag 8E/19
Es steht xx für den 1. Wert unter b), an die Stelle von + ist der Offset d einzutragen. Für bspw. d=7 erhält man insgesamt: DD 8E 07. Ausführungszeit beträgt 19 Speicherzyklen.
Die Spaltenüberschrift gilt bis zur nächsten Überschrift oder bis zur durchgezogenen Linie.

Operation	Operand → Funktion	Impliz. Op/Cy	A Op/Cy	B Op/Cy	C Op/Cy	D Op/Cy	E Op/Cy	H Op/Cy	L Op/Cy	(HL) Op/Cy	(BC) Op/Cy	(DE) Op/Cy	(IX+d) Op/Cy	IY+d) Op/Cy	Imm/8 Op/Cy	(Adr) Op/Cy	Flags SZ–H–PNC	8080-M
Arithmetik																		
													DDxx+	FDxx+				
ADC A,	A+Operand+Carry		8F/4	88/4	89/4	8A/4	8B/4	8C/4	8D/4	8E/7			8E/19	8E/19	CE/7		XX?X?X0X	ADC/ACI
ADD A,	A+Operand		87/4	80/4	81/4	82/4	83/4	84/4	85/4	86/7			86/19	86/19	C6/7		XX?X?X0X	ADD/ADI
SBC A,	A–Operand–Carry		9F/4	98/4	99/4	9A/4	9B/4	9C/4	9D/4	9E/7			9E/19	9E/19	DE/7		XX?X?X1X	SBB/SBI
SUB	A–Operand		97/4	90/4	91/4	92/4	93/4	94/4	95/4	96/7			96/19	96/19	D6/7		XX?X?X1X	SUB/SUI
INC	Inkrement		3C/4	04/4	0C/4	14/4	1C/4	24/4	2C/4	34/11			34/23	34/23			XX?X?X0–	INR
DEC	Dekrement		3D/4	05/4	0D/4	15/4	1D/4	25/4	2D/4	35/11			35/23	35/23			XX?X?X1–	DCR
DAA	BCD–Anpass./A	27/4															XX?X?X–X	DAA
		ED																
NEG	A:= – A	44/8															XX?X?X 1 X	
Logische/Boolesche Operationen																		
													DDxx+	FDxx+				
AND	A.UND.Operand		A7/4	A0/4	A1/4	A2/4	A3/4	A4/4	A5/4	A6/7			A6/19	A6/19	E6/7		XX?1?X00	ANA/ANI
XOR	A.XOR.Operand		AF/4	A8/4	A9/4	AA/4	AB/4	AC/4	AD/4	AE/7			AE/19	AE/19	EE/7		XX?0?X00	XRA/XRI
OR	A.ODER.Operand		B7/4	B0/4	B1/4	B2/4	B3/4	B4/4	B5/4	B6/7			B6/19	B6/19	F6/7		XX?0?X00	ORA/ORI
CPL	A-Reg. kompl.	2F/4															–– ?1? –1–	CMA
CP	Vergl. A/Operand		BF/4	B8/4	B9/4	BA/4	BB/4	BC/4	BD/4	BE/7			BE/19	BE/19	FE/7		XX?X?X1X	CMP/CPI
		ED																
CPD	Vergl./Dekrement	A9/16															XX?X?X1–	
CPDR	Blockvergleich –	B9/16-21															XX?X?X1–	
CPI	Vergl./Inkrement	A1/16															XX?X?X1–	
CPIR	Blockvergleich +	B1/16-21															XX?X?X1–	
Transfer-Befehle																		
													DDxx+	FDxx+				
LD A,	A ← Operand		7F/4	78/4	79/4	7A/4	7B/4	7C/4	7D/4	7E/7	0A/7	1A/7	7E/19	7E/19	3E/7	3A/13	––?–?–––	MOV/MVI
LD B,	B ← Operand		47/4	40/4	41/4	42/4	43/4	44/4	45/4	46/7			46/19	46/19	06/7		––?–?–––	MOV/MVI
LD C,	C ← Operand		4F/4	48/4	49/4	4A/4	4B/4	4C/4	4D/4	4E/7			4E/19	4E/19	0E/7		––?–?–––	MOV/MVI
LD D,	D ← Operand		57/4	50/4	51/4	52/4	53/4	54/4	55/4	56/7			56/19	56/19	16/7		––?–?–––	MOV/MVI
LD E,	E ← Operand		5F/4	58/4	59/4	5A/4	5B/4	5C/4	5D/4	5E/7			5E/19	5E/19	1E/7		––?–?–––	MOV/MVI
LD H,	H ← Operand		67/4	60/4	61/4	62/4	63/4	64/4	65/4	66/7			66/19	66/19	26/7		––?–?–––	MOV/MVI
LD L,	L ← Operand		6F/4	68/4	69/4	6A/4	6B/4	6C/4	6D/4	6E/7			6E/19	6E/19	2E/7		––?–?–––	MOV/MVI
LD (HL),	(HL) ← Operand		77/7	70/7	71/7	72/7	73/7	74/7	75/7						36/10		––?–?–––	MOV/MVI
LD (BC),	(BC) ← Operand		02/7														––?–?–––	STAX
LD (DE),	(DE) ← Operand		12/7														––?–?–––	STAX
LD (Adr),	Adr ← Operand		32/13														––?–?–––	STA
			DDxx+	DDxx+	DDxx+	DDxx+	DDxx+	DDxx+	DDxx+						DDxx+			
LD (IX+d),	(IX+d) ← Operand		77/19	70/19	71/19	72/19	73/19	74/19	75/19						36/19		––?–?–––	
			FDxx+	FDxx+	FDxx+	FDxx+	FDxx+	FDxx+	FDxx+						FDxx+			
LD (IY+d),	(IY+d) ← Operand		77/19	70/19	71/19	72/19	73/19	74/19	75/19						36/19		––?–?–––	
		ED																
LDD	Laden/Dekrement	A8/16															––?0?X0–	
LDDR	Blockversch.–	B8/16-21															––?0?00–	
LDI	Laden/Inkrement	A0/16															––?0?X0–	
LDIR	Blockversch.+	B0/16-21															––?0?00–	

Befehle mit 8-Bit-Struktur (Fortsetzung)

Operation	Operand → Funktion	Impliz. Op/Cy	A Op/Cy	B Op/Cy	C Op/Cy	D Op/Cy	E Op/Cy	H Op/Cy	L Op/Cy	(HL) Op/Cy	(BC) Op/Cy	(DE) Op/Cy	(IX+d) Op/Cy	IY+d) Op/Cy	Imm/8 Op/Cy	(Adr) Op/Cy	Flags SZ–H–PNC	8080-M
Verschiebe-Befehle																		
			CB	CB	CB	CB	CB	CB	CB	CB			DDCB+xx	FDCB+xx				
RL	Rot. ← Op,9		17/8	10/8	11/8	12/8	13/8	14/8	15/8	16/15			16/23	16/23			XX?0?X0X	
RLA	Rot. ← A,9	17/4															––?0?–0X	RAL
RLC	Rot. ← Op,8		07/8	00/8	01/8	02/8	03/8	04/8	05/8	06/15			06/23	06/23			XX?0?X0X	
RLCA	Rot. ← A,8	07/4															––?0?–0X	RLC
RR	Rot. → Op,9		1F/8	18/8	19/8	1A/8	1B/8	1C/8	1D/8	1E/15			1E/23	1E/23			XX?0?X0X	
RRA	Rot. → A,9	1F/4															––?0?–0X	RAR
RRC	Rot. → Op,8		0F/8	08/8	09/8	0A/8	0B/8	0C/8	0D/8	0E/15			0E/23	0E/23			XX?0?X0X	
RRCA	Rot. → A,8	0F/4															––?0?–0X	RRC
SLA	Shift ← Operand		27/8	20/8	21/8	22/8	23/8	24/8	25/8	26/15			26/23	26/23			XX?0?X0X	
SRL	Shift → Operand		3F/8	38/8	39/8	3A/8	3B/8	3C/8	3D/8	3E/15			3E/23	3E/23			XX?0?X0X	
SRA	Shift arithm. →		2F/8	28/8	29/8	2A/8	2B/8	2C/8	2D/8	2E/15			2E/23	2E/23			XX?0?X0X	
		ED																
RLD	Rot. ← Nibble	6F/18															XX?0?X0–	
RRD	Rot. → Nibble	67/18															XX?0?X0–	
Bit-Manipulation																		
			CB	CB	CB	CB	CB	CB	CB	CB			DDCB+xx	FDCB+xx				
BIT 0,	Test Bit 0		47/8	40/8	41/8	42/8	43/8	44/8	45/8	46/12			46/20	46/20			?X?1??0–	
BIT 1,	Test Bit 1		4F/8	48/8	49/8	4A/8	4B/8	4C/8	4D/8	4E/12			4E/20	4E/20			?X?1??0–	
BIT 2,	Test Bit 2		57/8	50/8	51/8	52/8	53/8	54/8	55/8	56/12			56/20	56/20			?X?1??0–	
BIT 3,	Test Bit 3		5F/8	58/8	59/8	5A/8	5B/8	5C/8	5D/8	5E/12			5E/20	5E/20			?X?1??0–	
BIT 4,	Test Bit 4		67/8	60/8	61/8	62/8	63/8	64/8	65/8	66/12			66/20	66/20			?X?1??0–	
BIT 5,	Test Bit 5		6F/8	68/8	69/8	6A/8	6B/8	6C/8	6D/8	6E/12			6E/20	6E/20			?X?1??0–	
BIT 6,	Test Bit 6		77/8	70/8	71/8	72/8	73/8	74/8	75/8	76/12			76/20	76/20			?X?1??0–	
BIT 7,	Test Bit 7		7F/8	78/8	79/8	7A/8	7B/8	7C/8	7D/8	7E/12			7E/20	7E/20			?X?1??0–	
SET 0,	Setze Bit 0		C7/8	C0/8	C1/8	C2/8	C3/8	C4/8	C5/8	C6/15			C6/23	C6/23			––?–?–––	
SET 1,	Setze Bit 1		CF/8	C8/8	C9/8	CA/8	CB/8	CC/8	CD/8	CE/15			CE/23	CE/23			––?–?–––	
SET 2,	Setze Bit 2		D7/8	D0/8	D1/8	D2/8	D3/8	D4/8	D5/8	D6/15			D6/23	D6/23			––?–?–––	
SET 3,	Setze Bit 3		DF/8	D8/8	D9/8	DA/8	DB/8	DC/8	DD/8	DE/15			DE/23	DE/23			––?–?–––	
SET 4,	Setze Bit 4		E7/8	E0/8	E1/8	E2/8	E3/8	E4/8	E5/8	E6/15			E6/23	E6/23			––?–?–––	
SET 5,	Setze Bit 5		EF/8	E8/8	E9/8	EA/8	EB/8	EC/8	ED/8	EE/15			EE/23	EE/23			––?–?–––	
SET 6,	Setze Bit 6		F7/8	F0/8	F1/8	F2/8	F3/8	F4/8	F5/8	F6/15			F6/23	F6/23			––?–?–––	
SET 7,	Setze Bit 7		FF/8	F8/8	F9/8	FA/8	FB/8	FC/8	FD/8	FE/15			FE/23	FE/23			––?–?–––	
RES 0,	Reset Bit 0		87/8	80/8	81/8	82/8	83/8	84/8	85/8	86/15			86/23	86/23			––?–?–––	
RES 1,	Reset Bit 1		8F/8	88/8	89/8	8A/8	8B/8	8C/8	8D/8	8E/15			8E/23	8E/23			––?–?–––	
RES 2,	Reset Bit 2		97/8	90/8	91/8	92/8	93/8	94/8	95/8	96/15			96/23	96/23			––?–?–––	
RES 3,	Reset Bit 3		9F/8	98/8	99/8	9A/8	9B/8	9C/8	9D/8	9E/15			9E/23	9E/23			––?–?–––	
RES 4,	Reset Bit 4		A7/8	A0/8	A1/8	A2/8	A3/8	A4/8	A5/8	A6/15			A6/23	A6/23			––?–?–––	
RES 5,	Reset Bit 5		AF/8	A8/8	A9/8	AA/8	AB/8	AC/8	AD/8	AE/15			AE/23	AE/23			––?–?–––	
RES 6,	Reset Bit 6		B7/8	B0/8	B1/8	B2/8	B3/8	B4/8	B5/8	B6/15			B6/23	B6/23			––?–?–––	
RES 7,	Reset Bit 7		BF/8	B8/8	B9/8	BA/8	BB/8	BC/8	BD/8	BE/15			BE/23	BE/23			––?–?–––	
Weitere Befehle (ohne Interrupt- und Port-Anweisungen)																		
CCF	Kompl. Carry	3F/4															––?X?–0X	CMC
SCF	Carry: = 1	37/4															––?0?–01	STC
NOP	Nulloperation	00/4															––?–?–––	NOP

Print-Using

Rechtsbündige Zahlenformatierung

von Ulrich Stiehl

Unter Print-Using versteht man die Formatierung einer Zahl, z.B. die rechtsbündige und auf x Stellen nach dem Dezimalpunkt gerundete Zahlenausgabe mit einer vordefinierten Feldlänge. Da der in anderen Basic-Dialekten vorhandene Print-Using-Befehl in Applesoft fehlt, wurde ich schon wiederholt gebeten, eine kurze und flexible Print-Using-Routine zu veröffentlichen. In „Apple Assembler", S. 192ff., war zwar bereits eine Print-Using-Version publiziert worden, doch war diese erstens auf 1-3 Dezimalstellen begrenzt und außerdem war das Programm nicht relokativ. Das nachfolgende neue PRINT.USING hat folgende erweiterte Features:

1. Die Routine ist völlig relokativ, kann also mit einer beliebigen Startadresse in den Speicher geladen werden, z.B.
BLOAD PRINT.USING, A768
BLOAD PRINT.USING, A$9500
usw.

2. Die Routine funktioniert sowohl unter Interpreter- wie auch unter Compiler-Applesoft. Bei Interpreter-Applesoft erfolgt der Aufruf wahlweise mit CALL oder USR, bei Compiler-Applesoft (TASC, HAYDEN usw.) nur mit USR. CALL hat die Syntax
CALL A, F, D, N
und USR die Syntax
N = USR (N)

3. Es können 4 Parameter übergeben bzw. (bei USR) gepokt werden. Die Buchstaben A, F, D und N stehen für Variablennamen, die durch beliebige andere Variablennamen sowie durch Formelausdrücke und Konstanten ersetzt werden können.

3.1. A = *Anfangsadresse*: Diese entspricht der BLOAD-Anfangsadresse.

3.2. F = *Feldlänge*: Die Feldlänge sollte im Bereich 11 bis 15 liegen. Wenn z.B. die Feldlänge 11 beträgt und eine formatierte Zahl, z.B. „123.45", einschließlich Dezimalpunkt 6 Stellen einnehmen würde, so werden 11 minus 6 = 5 Leertasten *vor* der Zahl eingefügt, damit eine Feldlänge von 11 erreicht wird. Die denkbar längste Zahl, nämlich „-1.23456789E-12", nimmt 15 Stellen ein; daher die maximale Feldlänge 15. Bei Nicht-Exponentialzahlen genügt eine Feldlänge von 11, weil „-1234567.89" 11 Stellen belegt. Man beachte, daß unter Applesoft die Mantisse maximal 9 Stellen einnimmt. Ferner ist ein mögliches Minus („-") sowie ein möglicher „." zu berücksichtigen. Wenn Sie wissen, daß die von Ihnen benötigten Zahlen einen bestimmten Bereich nicht überschreiten, können Sie die Feldlänge entsprechend reduzieren, z.B. auf F = 3 für natürliche Zahlen im Bereich 0 bis 999.

3.3. D = *Dezimalstellen*: Die Anzahl der Dezimalstellen (empfohlen 1–3) hängt von der Feldlänge ab. Ferner muß man berücksichtigen, daß bei mehr als 3 Dezimalstellen oft keine Rundung mehr möglich ist. Beispielsweise läßt sich 123456789 nur noch „runden", indem man Nullen anhängt, z.B. „123456789.0000". Die Rundung erfolgt dadurch, daß die Zahl N zunächst mit 10 ↑ D multipliziert, dann 0.5 addiert, dann der Nachkommarest abgehackt und schließlich durch 10 ↑ D geteilt wird, also bei N = 123.456 und D = 2:
123.456 * 100 = 12345.6
12345.6 + 0.5 = 12346.1
INT (12346.1) = 12346
12346 / 100 = 123.46
PRINT.USING gestattet auch die rechtsbündige Formatierung von Ganzzahlen, indem man D auf 0 setzt.

3.4. N = Number = Zahl: Dies ist die zu rundende Fließkommazahl.

3.5. POKE A + 1, 1 oder 0: Dies ist das *FAC-Flag* (= Exponentialzahl-Flag). Nach POKE A + 1, 1 wird die Zahl N, wenn sie sich als Exponentialzahl (mit E, z.B. 1E20) nicht runden läßt, trotzdem ohne Rücksicht auf die Feldlänge ausgegeben. Nach POKE A + 1, 0 werden Exponentialzahlen unterdrückt und statt dessen durch Leerzeichen ersetzt. Auf diese Weise wird der mehrspaltige Ausdruck von Zahlenkolonnen stets korrekt ausgeführt.

4. CALL-Syntax: CALL A, F, D, N
A steht für Anfangsadresse, z.B. 768
F steht für Feldlänge, z.B. 11
D steht für Dezimalstellen, z.B. 2
N steht für Number = zu rundende Zahl

5. USR-Syntax: N = USR (N)
N steht für zu rundende Zahl. Der Wert von N geht durch Print-Using nicht verloren. Die Werte für F und D müssen bei USR gepokt werden. Wenn z.B. die Anfangsadresse A = 768 beträgt, dann gilt:
POKE A + 1, FAC-Flag
POKE A + 5, F
POKE A + 9, D
POKE A + 12, 56
FAC-Flag, F und D müssen nur einmal gepokt werden. Ferner muß der USR-Vektor gesetzt und durch POKE A + 12, 56 der Print-Using-Routine mitgeteilt werden, daß USR aktiv ist. Da dies alles etwas umständlich ist, sollte USR nur bei compilierten Applesoft-Programmen verwendet werden.

Um PRINT.USING zu installieren, können Sie entweder das Maschinenprogramm mit BLOAD PRINT.USING laden oder die DATA-Statements in das eigene Applesoft-Programm einbauen (siehe PRINT.USING.DATA). Das Demonstrationsprogramm PRINT.USING.DEMO veranschaulicht im Detail die Anwendung.
Die Print-Using-Routine formatiert *positive* Zahlen, die größer/gleich 0 und kleiner als 1 sind, in der Form „0.XX" (hier als Beispiel 2 Dezimalstellen). *Negative* Zahlen in demselben Bereich werden in der Form „-.XX" ausgewiesen. Wer hier die Form „-0.XX" wünscht (also „0" zwischen „-" und „."), kann die verbesserte und um einige Bytes gestraffte Version PRINT.USING.G benutzen, die sich auf der Peeker-Sammeldisk befindet. Diese Verbesserung wurde von Harald Grumser vorgenommen.

PRINT.USING.DEMO

```
100 REM PRINT.USING.DEMO
110 REM A = Startadresse
120 REM F = Feldlänge (1-15)
130 REM D = Dezimalstellen (0-8)
140 REM N = Zahl(enausdruck)
150 HOME : PRINT CHR$ (4)"BLOAD PRINT.USING,A"768
160 PRINT "PRINT.USING.DEMO": PRINT : PRINT "1.
    Input.Demo": PRINT : PRINT "2. Random.Demo": PRINT :
    PRINT "3. USR.Demo": PRINT : PRINT "0. Ende": PRINT :
    PRINT
170 GET X$: ON X$ = "1" GOTO 190: ON X$ = "2" GOTO 290:
    ON X$ = "3" GOTO 400: ON X$ < > "0" GOTO 170: END
180 REM Input.Demo
190 A = 768
200 POKE A + 12,24: REM CLC (sicherheitshalber wegen USR)
210 F = 11
220 D = 2
230 INPUT "Zahl:";N$: IF N$ = "" THEN END
240 N = VAL (N$)
250 CALL A,F,D,N: REM Print.Using!
260 PRINT " formatiert": PRINT N;" unformatiert"
270 GOTO 230
280 REM Random.Demo
290 A = 768:F = 10
300 POKE A + 12,24: REM CLC (sicherheitshalber wegen USR)
310 Z = 1000
320 FOR D = 5 TO 0 STEP - 1
330 FOR R = 1 TO 100
340 N = RND (1) * Z
350 CALL A,F,D,N: REM Print.Using!
360 NEXT R: NEXT D
370 END
380 REM USR.Demo
390 REM Vektor $0300=768
400 POKE 10,76: POKE 11,0: POKE 12,3:A = 768
410 POKE A + 12,56: REM SEC!
420 POKE A + 1,0: REM FACFLAG
430 POKE A + 5,10: REM FELDLEN
440 Z = - 10000
450 FOR D = 3 TO 0 STEP - 1
460 POKE A + 9,D: REM DEZLEN
470 FOR R = 1 TO 100
480 N = RND (1) * Z
490 N = USR (N): REM Print.Using!
500 NEXT R: NEXT D
510 END
```

PRINT.USING.DATA

```
100 REM PRINT.USING.DATA
110 DATA 169,0,133,252,169,11,133,253,169,2
120 DATA 133,254,24,176,16,32,76,231,134,253
130 DATA 32,76,231,134,254,32,190,222,32,103
140 DATA 221,162,16,160,1,32,43,235,164,254
150 DATA 132,255,240,7,32,57,234,198,255,208
160 DATA 249,165,162,72,70,162,32,160,231,32
170 DATA 35,236,104,16,3,32,208,238,32,52
180 DATA 237,162,0,189,0,1,240,61,201,69
190 DATA 240,30,232,208,244,189,0,1,232,157
200 DATA 0,1,202,202,16,245,160,48,201,45
210 DATA 240,5,140,0,1,208,220,140,1,1
220 DATA 240,215,56,165,252,208,6,166,253,32
230 DATA 74,249,24,169,16,160,1,32,249,234
240 DATA 144,6,32,46,237,24,144,241,96,228
250 DATA 254,144,198,240,196,228,253,176,219,164
260 DATA 254,240,1,232,138,168,196,253,176,6
270 DATA 32,87,219,200,208,246,169,0,157,0
280 DATA 1,164,254,240,19,202,202,189,0,1
290 DATA 72,169,46,157,0,1,104,232,157,0
300 DATA 1,136,208,237,185,0,1,240,179,32
310 DATA 92,219,200,208,245
320 RESTORE :A = 768: FOR X = A TO A + 204: READ Y: POKE
    X,Y: NEXT
330 F = 11:D = 2:N = SIN (1): CALL A,F,D,N: PRINT : PRINT
    "Print.Using installiert"
```

Hinweis:

PRINT.USING läuft unter DOS 3.3 und ProDOS.

PRINT.USING

BSAVE PRINT.USING, A768, L205
BLOAD PRINT.USING mit beliebiger Startadresse

```
1                ORG  768
2     *
3     * PRINT.USING
4     * ============
5     *
6     * von U.Stiehl/28.03.85
7     *
8     * Diese relokative Utility
9     * mit beliebiger Startadresse
10    * oder ersatzweise mit
11    * BLOAD PRINT.USING, A 768
12    * laden und dann aufrufen mit
13    *
14    * CALL A, F, D, N
15    *
16    * A = Adresse (z.B. 768)
17    * F = Feldlänge (1-15)
18    * D = Dezimalstellen (0-8)
19    * N = zu formatierende Zahl
20    *
21    * Die Namen der Variablen können
22    * beliebig gewählt werden.
23    * Sie sind jedoch durch Kommas
24    * abzugrenzen.
25    *
26    * FACFLAG
27    * -------
28    *
29    * POKE 768 + 1, 1 = ja, 0 = nein
30    * POKE   A + 1, 1 = ja, 0 = nein
31    *
32    * Exponentielle und nicht-rundbare
33    * Zahlen werden unformatiert
34    * ausgedruckt, wenn das FACFLAG
35    * auf 1 gesetzt ist. Andernfalls
36    * wird das Feld mit Spaces gefüllt.
37    *
38    * Es wird daher ein llstelliges Feld
39    * empfohlen, da dann meist gerundet
40    * werden kann. Beispiele:
41    *
42    * F = 11, D = 1  -12345678.9
43    * F = 11, D = 9  -.123456789
44    *
45    * D.h. 9stellige Mantisse sowie
46    * 2 weitere Stellen für "-" + "."
47    * D = 9 ist absoluter Grenzfall
48    * für N: 0 > /N/ < 1
49    * Warnung: Da die Parameter F und D
50    * nicht auf Legalität überprüft
51    * werden, "dreht" PRINT.USING "durch",
52    * wenn F > 15 oder D > 9 und
53    * D > F - 2 ist.
54    *
55    * PRINT.USING benötigt übrigens
56    * ca. 0.02 Sek. pro Zahl.
57    *-------------------------------
58    *
59    MINUS    EQU  $2D        ;"-"
60    PUNKT    EQU  $2E        ;"."
61    NULL     EQU  $30        ;"0"
62    EXPO     EQU  $45        ;"E"
63    *
64    FACFLAG  EQU  $00FC      ;252
65    FELDLEN  EQU  $00FD      ;253
66    DEZLEN   EQU  $00FE      ;254
67    DEZCNT   EQU  $00FF      ;255
68    *
69    FAC      EQU  $009D      ;-$00A2
70    FACSIGN  EQU  $00A2      ;Signum
71    STACKFAC EQU  $0100      ;-$010F
72    FACSAVE  EQU  $0110      ;-$0114
73    *
74    MUL10    EQU  $EA39      ;N=N*10
75    FADDH    EQU  $E7A0      ;N=N+0.5
76    INT      EQU  $EC23      ;N=INT(N)
77    NEGOP    EQU  $EED0      ;N=-N
78    FOUT     EQU  $ED34      ;N in Stack
79    FRMNUM   EQU  $DD67      ;N in FAC
80    COMBYTE  EQU  $E74C      ;Byte in X
81    MOVFM    EQU  $EAF9      ;MEM->FAC
82    MOVMF    EQU  $EB2B      ;FAC->MEM
```

```
                    83  CHKCOM   EQU  $DEBE       ;Komma-Check
                    84  *
                    85  PRNTFAC  EQU  $ED2E       ;Print FAC
                    86  OUTDO    EQU  $DB5C       ;wie COUT
                    87  OUTSPC   EQU  $DB57       ;wie PRBL2
                    88  PRBLK2   EQU  $F94A       ;X=Blanks
                    89  *-------------------------------
                    90  *
                    91  * Parameter für USR festlegen
                    92  *
Ø3ØØ: A9 ØØ         93  ADRESSE  LDA  #Ø          ;A+1
Ø3Ø2: 85 FC         94           STA  FACFLAG
Ø3Ø4: A9 ØB         95           LDA  #11         ;A+5
Ø3Ø6: 85 FD         96           STA  FELDLEN
Ø3Ø8: A9 Ø2         97           LDA  #2          ;A+9
Ø3ØA: 85 FE         98           STA  DEZLEN
                    99  *
                    1ØØ * Wenn SEC, dann USR statt CALL
                    1Ø1 * CLC = $18; SEC = $38 (dez. 56)
                    1Ø2 *
Ø3ØC: 18            1Ø3 USRFLAG  CLC              ;A+12
Ø3ØD: BØ 1Ø         1Ø4          BCS  USER
                    1Ø5 *
                    1Ø6 * COMBYTE überspringt Komma und
                    1Ø7 * wertet die Variablen F und D aus.
                    1Ø8 * Danach befindet sich der
                    1Ø9 * Wert von F bzw. D im X-Register.
                    11Ø *
Ø3ØF: 2Ø 4C E7      111          JSR  COMBYTE     ;F
Ø312: 86 FD         112          STX  FELDLEN
Ø314: 2Ø 4C E7      113          JSR  COMBYTE     ;D
Ø317: 86 FE         114          STX  DEZLEN
                    115 *
                    116 * CHKCOM überspringt Komma.
                    117 * FRMNUM überträgt Wert
                    118 * der Fließkommazahl N in
                    119 * den Floating-Point-Accumulator.
                    12Ø *
Ø319: 2Ø BE DE      121          JSR  CHKCOM
Ø31C: 2Ø 67 DD      122          JSR  FRMNUM      ;N
                    123 *
                    124 * N = USR (N)
                    125 * ------------
                    126 *
                    127 * Statt mit CALL A, F, D, N
                    128 * kann PRINT USING auch mit
                    129 * N = USR (N)
                    13Ø * aufgerufen werden.
                    131 * Zu diesem Zweck muß
                    132 *
                    133 * (1) der USR-Vektor auf A gesetzt
                    134 *     (ØØØA: 4C ØØ Ø3) und
                    135 * (2) FACFLAG  A +  1 (Ø-1)
                    136 *     FELDLEN  A +  5 (1-15)
                    137 *     DEZLEN   A +  9 (Ø-8)
                    138 *     USRFLAG  A + 12 ($38=dez.56)
                    139 *     gepokt werden.
                    14Ø *
Ø31F: A2 1Ø         141 USER     LDX  #<FACSAVE  ;FAC
Ø321: AØ Ø1         142          LDY  #>FACSAVE  ;retten
Ø323: 2Ø 2B EB      143          JSR  MOVMF
                    144 *-------------------------------
                    145 *
                    146 * Integer-Zahl (DEZLEN = Ø)?
                    147 *
Ø326: A4 FE         148          LDY  DEZLEN
Ø328: 84 FF         149          STY  DEZCNT
Ø32A: FØ Ø7         15Ø          BEQ  INT1        ;Integer
                    151 *
                    152 *  N = N * (10 ↑ D)
                    153 *
Ø32C: 2Ø 39 EA      154 MULTIPLY JSR  MUL1Ø
Ø32F: C6 FF         155          DEC  DEZCNT
Ø331: DØ F9         156          BNE  MULTIPLY
                    157 *
                    158 * N = INT (/N/ + Ø.5)
                    159 *
Ø333: A5 A2         16Ø INT1     LDA  FACSIGN
Ø335: 48            161          PHA              ;Save +-
Ø336: 46 A2         162          LSR  FACSIGN     ;N=/N/
Ø338: 2Ø AØ E7      163          JSR  FADDH       ;N=N+Ø.5
Ø33B: 2Ø 23 EC      164          JSR  INT         ;Integer
                    165 *
                    166 * N = /N/ plus Signum (+/-)
                    167 *
Ø33E: 68            168          PLA              ;Load +-
Ø33F: 1Ø Ø3         169          BPL  STRING
Ø341: 2Ø DØ EE      17Ø          JSR  NEGOP
                    171 *
                    172 * N als String im Stack ablegen
                    173 *
Ø344: 2Ø 34 ED      174 STRING   JSR  FOUT        ;N->Stack
                    175 *-------------------------------
                    176 *
                    177 * STRLEN = Länge des N-Strings = X
                    178 *
Ø347: A2 ØØ         179 STRLEN1  LDX  #Ø
Ø349: BD ØØ Ø1      18Ø STRLEN2  LDA  STACKFAC,X
Ø34C: FØ 3D         181          BEQ  ZERO
Ø34E: C9 45         182          CMP  #EXPO
Ø35Ø: FØ 1E         183          BEQ  EXPONENT
Ø352: E8            184          INX
Ø353: DØ F4         185          BNE  STRLEN2     ;stets
                    186 *
                    187 * Bei STRLEN < DEZLEN
                    188 * N-String nach rechts rücken
                    189 * und "Ø" voranstellen.
                    19Ø * Bei Negativzahl NACH "-"
                    191 *
Ø355: BD ØØ Ø1      192 ADJUST1  LDA  STACKFAC,X
Ø358: E8            193          INX
Ø359: 9D ØØ Ø1      194          STA  STACKFAC,X
Ø35C: CA            195          DEX
Ø35D: CA            196          DEX
Ø35E: 1Ø F5         197          BPL  ADJUST1
Ø36Ø: AØ 3Ø         198          LDY  #NULL
Ø362: C9 2D         199          CMP  #MINUS
Ø364: FØ Ø5         2ØØ          BEQ  ADJUST2
Ø366: 8C ØØ Ø1      2Ø1          STY  STACKFAC
Ø369: DØ DC         2Ø2          BNE  STRLEN1
Ø36B: 8C Ø1 Ø1      2Ø3 ADJUST2  STY  STACKFAC+1
Ø36E: FØ D7         2Ø4          BEQ  STRLEN1
                    2Ø5 *
                    2Ø6 * Entweder Exponentialzahl
                    2Ø7 * oder Feld zu kurz für N-String.
                    2Ø8 *
Ø37Ø: 38            2Ø9 EXPONENT SEC              ;Carry!
Ø371: A5 FC         21Ø          LDA  FACFLAG
Ø373: DØ Ø6         211          BNE  FACMOVE2
                    212 *
                    213 * Nur Spaces ausgeben
                    214 *
Ø375: A6 FD         215          LDX  FELDLEN
Ø377: 2Ø 4A F9      216          JSR  PRBLK2
                    217 *
                    218 * Wenn FACFLAG nicht Ø ist,
                    219 * dann Exponentialzahl oder
                    22Ø * nicht in das Feld passende
                    221 * Zahl trotzdem ausgeben.
                    222 *
                    223 * MOVFM ändert nicht Carry-Flag!
                    224 *
Ø37A: 18            225 CARRY    CLC
Ø37B: A9 1Ø         226 FACMOVE2 LDA  #<FACSAVE
Ø37D: AØ Ø1         227          LDY  #>FACSAVE
Ø37F: 2Ø F9 EA      228          JSR  MOVFM
Ø382: 9Ø Ø6         229          BCC  FACMOVE3
Ø384: 2Ø 2E ED      23Ø FACOUT1  JSR  PRNTFAC
Ø387: 18            231          CLC
Ø388: 9Ø F1         232          BCC  FACMOVE2
                    233 *
                    234 * Beim Exit befindet sich N
                    235 * wieder im FAC.
                    236 *
Ø38A: 6Ø            237 FACMOVE3 RTS              ;Exit
                    238 *
                    239 * STRLEN < = DEZLEN?
                    24Ø * Tritt auf für N:   Ø > N < 1
                    241 *
Ø38B: E4 FE         242 ZERO     CPX  DEZLEN
Ø38D: 9Ø C6         243          BCC  ADJUST1
Ø38F: FØ C4         244          BEQ  ADJUST1
                    245 *
                    246 * STRLEN + "." > Feld?
                    247 *
Ø391: E4 FD         248          CPX  FELDLEN
Ø393: BØ DB         249          BCS  EXPONENT
Ø395: A4 FE         25Ø          LDY  DEZLEN
Ø397: FØ Ø1         251          BEQ  INT2
                    252 *
                    253 * X = X + 1 wegen "."
                    254 *
```

```
0399: E8        255           INX
                256  *
                257  * Feldanfang mit Spaces auffüllen
                258  *
039A: 8A        259  INT2     TXA
039B: A8        260           TAY
039C: C4 FD     261  SPACES   CPY  FELDLEN
039E: B0 06     262           BCS  ENDMARK
03A0: 20 57 DB  263           JSR  OUTSPC
03A3: C8        264           INY
03A4: D0 F6     265           BNE  SPACES
                266  *
                267  * Endmarker um 1 nach oben setzen
                268  *
                269  * Beispiel: 100e -> 100ee
                270  *
03A6: A9 00     271  ENDMARK  LDA  #$00       ;Endmarker
03A8: 9D 00 01  272           STA  STACKFAC,X
03AB: A4 FE     273           LDY  DEZLEN
03AD: F0 13     274           BEQ  FACOUT2    ;Integer
                275  *
                276  * Dezimalpunkt einfügen
                277  *
                278  * Beispiel: 100ee  (1*100)
                279  *           100.e
                280  *           10.0e
                281  *           1.00e  (1.00)
                282  *
03AF: CA        283  DEZPUNKT DEX
03B0: CA        284           DEX
03B1: BD 00 01  285           LDA  STACKFAC,X
03B4: 48        286           PHA
03B5: A9 2E     287           LDA  #PUNKT
03B7: 9D 00 01  288           STA  STACKFAC,X
03BA: 68        289           PLA
03BB: E8        290           INX
03BC: 9D 00 01  291           STA  STACKFAC,X
03BF: 88        292           DEY
03C0: D0 ED     293           BNE  DEZPUNKT
                294  *
                295  * Stack bis Endmarker ausgeben
                296  *
03C2: B9 00 01  297  FACOUT2  LDA  STACKFAC,Y
03C5: F0 B3     298           BEQ  CARRY
03C7: 20 5C DB  299           JSR  OUTDO
03CA: C8        300           INY
03CB: D0 F5     301           BNE  FACOUT2
                302  *
                303  * Wie PRINT.USING funktioniert,
                304  * soll anhand der Routinen
                305  * MULTIPLY, ADJUST und DEZPUNKT
                306  * für DEZLEN = 2 (2 Durchläufe)
```

```
                307  * durch Grenzfälle erläutert
                308  * werden (e = Endmarker)
                309  *
                310  * Zahl "1": 1 * 100 = "100"
                311  *
                312  *  100e  N-String
                313  *  10.0e DEZPUNKT 1. Durchlauf
                314  *  1.00e DEZPUNKT 2. Durchlauf
                315  *
                316  * Zahl "-1": -1 * 100 = "-100"
                317  *
                318  *  -100e  N-String
                319  *  -10.0e DEZPUNKT 1. Durchlauf
                320  *  -1.00e DEZPUNKT 2. Durchlauf
                321  *
                322  * Zahl ".1": .1 * 100 = "10"
                323  *
                324  *  10e   N-String
                325  *  010e  ADJUST   1. Durchlauf
                326  *  01.0e DEZPUNKT 1. Durchlauf
                327  *  0.10e DEZPUNKT 2. Durchlauf
                328  *
                329  * Zahl "-.1": -.1 * 100 = "-10"
                330  *
                331  *  -10e  N-String
                332  *  -1.0e DEZPUNKT 1. Durchlauf
                333  *  -.10e DEZPUNKT 2. Durchlauf
                334  *
                335  * Zahl ",01": .01 * 100 = "1"
                336  *
                337  *  1e    N-String
                338  *  01e   ADJUST   1. Durchlauf
                339  *  001e  ADJUST   2. Durchlauf
                340  *  00.1e DEZPUNKT 1. Durchlauf
                341  *  0.01e DEZPUNKT 2. Durchlauf
                342  *
                343  * Zahl "-.01": -.01 * 100 = "-1"
                344  *
                345  *  -1e   N-String
                346  *  -01e  ADJUST   1. Durchlauf
                347  *  -0.1e DEZPUNKT 1. Durchlauf
                348  *  -.01e DEZPUNKT 2. Durchlauf
                349  *
                350  * Zahl "0": 0 * 100 = "0"
                351  *
                352  *  0e    N-String
                353  *  00e   ADJUST   1. Durchlauf
                354  *  000e  ADJUST   2. Durchlauf
                355  *  00.0e DEZPUNKT 1. Durchlauf
                356  *  0.00e DEZPUNKT 2. Durchlauf
205 Bytes
```


Für IIc und IIe mit 64K-Karte

SUPERPLOT

Double-Hires-Utility

von Karl-Walter Bott, 1984, Programmdiskette und Manual, DM 48,–

SUPERPLOT ist eine neue, ungewöhnlich kompakte und schnelle Ampersand-Utility für Double Hires, die einschließlich eines vollständigen ASCII-Shape-Zeichensatzes wahlweise in Bank 1 oder Bank 2 der Language Card liegt und damit sowohl unter ProDOS als auch unter DOS 3.3, falls letzteres in die LC-Bank geschoben wurde, benutzt und in eigene Applesoftprogramme integriert werden kann. SUPERPLOT unterstützt die üblichen HGR-Befehle, denen lediglich ein & vorangestellt werden muß, also z. B. & HPLOT 500, 100 TO 500, 150 usw. SUPERPLOT ist speziell für das Plotten von beschrifteten wissenschaftlichen Funktionskurven mit hoher Auflösung gedacht und weniger für HGR-Spiele.

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

peeker Börse

Gelegenheitsanzeigen

Sie können unter dieser Rubrik zu einem besonders günstigen Preis

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

Musteranzeige privat

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

Beispiel:

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

nur DM 17,10 inkl. MwSt.

Musteranzeige gewerblich

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

Beispiel:

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

nur DM 137,94 inkl. MwSt.

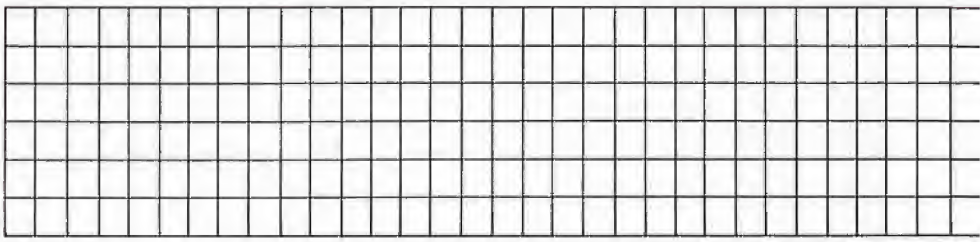
peeker

peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

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

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



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

peeker

Produkt-Karte

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

über _____

_____

bitte ich um detaillierte Information.

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

Menge	Produkt und Bestellnummer	à DM	gesamt DM

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

_____

Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

peeker

Info-Karte

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

_____

_____

_____

_____

_____

_____

peeker-Börse

Vorname, Name

Beruf

Straße

Wohnort

PLZ

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

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

Datum Unterschrift

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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

Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

ANTWORTKARTE

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

POSTKARTE

Firma

Straße

PLZ/Ort

POSTKARTE

peeker
Redaktion
Postfach 10 28 69
6900 Heidelberg 1

Produkt-Karte

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

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

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

Damit erleichtern Sie dem Hersteller eine gezielte Beantwortung Ihrer Anfrage.

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

peeker
MAGAZIN FÜR APPLE-COMPUTER

Referenztest

Zeilenverweise in Applesoft-Programmen

von Ludger T. Engbert

Bei der Entwicklung umfangreicher Applesoft-Programme mit vielen Zeilenverweisen (GOTO, GOSUB) ist es z.T. schwierig nachzuvollziehen, ob alle Referenzen erfüllt sind. Manche „UNDEFINED STATEMENT"-Meldung erscheint erst nach längerem Gebrauch, weil ein selten benutzter Programmteil beim Test nicht berücksichtigt wurde.
Abhilfe schafft das Programm **REF.TEST**, das alle Zeilenverweise verifiziert und Sprünge auf nicht existierende Zeilen anzeigt. Dabei werden nicht nur alle GOTO- und GOSUB-Befehle überprüft, sondern auch Zeilennummern nach THEN sowie Aufzählungen (z.B. ON I GOSUB 100, 200, 300) berücksichtigt.

Erläuterung zum Programmablauf

Beim ersten Aufruf durch „BRUN REF.-TEST" wird der Ampersand-Vektor initialisiert, wodurch die Routine immer wieder durch Eingabe von „&" gestartet werden kann.
Beim eigentlichen Aufruf wird zunächst der aktuelle Textpointer gesichert, da dieser von LINGET (s.u.) bei der Auswertung der Zeilennummer benötigt wird.
Einer Applesoft-Programmzeile sind immer 4 Bytes vorangestellt: Die ersten beiden Bytes enthalten einen Zeiger auf die nachfolgende Programmzeile, die nächsten beiden Bytes beinhalten die Zeilennummer (Low-Byte zuerst).
Die Routine NXTLIN (Next Line) testet, ob das Programmende erreicht ist. In diesem Fall sind beide Link-Bytes Null (Das Low-Byte kann unabhängig vom Programmende Null sein). Wurde das Programmende erreicht, wird nach END gesprungen, um den Textpointer wieder herzustellen. Andernfalls überträgt das Programm die derzeitige Zeilennummer nach CURLIN (Current Line).
Die Routine NXTBYT (Next Byte) testet das Zeichen, auf das der Textpointer zeigt. Im Falle, daß es sich um
GOTO,
GOSUB oder
THEN, gefolgt von einer Ziffer
handelt, wird die Routine LINE? abgearbeitet, die überprüft, ob eine entsprechende Zeile im Applesoft-Programm existiert. Ist dieser Test negativ, wird ein Fragezeichen, die gesuchte Zeile und die Zeilennummer der fehlerhaften Anweisung ausgegeben.
In jedem Fall wird in der Routine FOUND getestet, ob nach der derzeitigen Zeilennummer ein Komma folgt. Auch diese weiteren Zeilennummern werden von der Routine LINE? untersucht, bis kein weiteres Komma mehr folgt.

Erklärung der aufgerufenen Applesoft-Routinen

CHRTST – (Character test) testet das Zeichen (Character) im Akku. Handelt es sich dabei um eine Ziffer ($30-$39), so wird das Carry-Flag gelöscht. Bei End-of-Statement ($00 oder „:") wird das Zero-Flag gesetzt.
FNDLIN – (Find Line) durchsucht den BASIC-Programmtext nach der Zeilennummer, die in LINNUM ($0050, $0051) abgelegt ist. Existiert die Zeile, wird das Carry-Flag gesetzt.
LINGET – (Line Get) wertet den Ausdruck (Zahl als ASCII-Folge), auf den der Textpointer ($00B8, $00B9) verweist, aus und trägt das Ergebnis in LINNUM ($0050, $0051) ein. Beim Einsprung müssen Akku und Carry-Flag durch CHRGET (bzw. CHRTST) initialisiert sein.
CRDO – (Carriage Return do) sendet ein CR zum aktuellen Ausgabegerät.
OUTSP – (Out Space) gibt ein Leerzeichen ($20) aus.
OUTQST – (Out Question Mark) gibt ein Fragezeichen ($3F) aus.
LINPRT – (Line Print) gibt die Zahl dezimal aus, die sich im Akku (Low-Byte) und X-Register (High-Byte) befindet.

Kurzhinweise

1. Zweck:
Hilfsprogramm zur Überprüfung von Zeilenverweisen
2. Konfiguration:
Apple II+, IIe, IIc; wahlweise DOS 3.3 oder ProDOS
3. Test:
BRUN REF.TEST
Beliebiges Applesoft-Programm laden
&
4. Sammeldisk:
REF.TEST
(Maschinenprogramm)
T.REF.TEST
(Big-Mac-Quelltext)

REF.TEST

```
                        1   **************************
                        2   *                        *
                        3   *       REF.TEST         *
                        4   *                        *
                        5   *   Applesoft-BASIC-     *
                        6   *       Utility          *
                        7   *                        *
                        8   *  zeigt alle            *
                        9   *  UNDEFINED STATEMENTS  *
                        10  *                        *
                        11  *  L.T. Engbert, Jan. 84 *
                        12  *                        *
                        13  **************************
                        14  *
                        15  * BSAVE REF.TEST, A$300, L$93
                        16  *
                        17  * Installieren mit BRUN REF.TEST
                        18  * Aufruf mit "&"
                        19  *
                        20          ORG  $300
                        21
                        22  LINNUM  EQU  $50
                        23  TXTTAB  EQU  $67
                        24  CURLIN  EQU  $75
                        25  TXTPTR  EQU  $B8
                        26  CHRTST  EQU  $BA
                        27  AMPER   EQU  $3F5
                        28  FNDLIN  EQU  $D61A
                        29  LINGET  EQU  $DA0C
                        30  CRDO    EQU  $DAFB
                        31  OUTSP   EQU  $DB57
                        32  OUTQST  EQU  $DB5A
                        33  LINPRT  EQU  $ED24
                        34
                        35  ** Ampersand-Vektor auf  **
                        36  ** Programmanfang setzen **
                        37
0300: A9 4C             38          LDA  #$4C       ;JMP
0302: 8D F5 03          39          STA  AMPER      ;&-Adresse
0305: A9 10             40          LDA  #<REFS
0307: 8D F6 03          41          STA  AMPER+1    ;Lo-Byte
030A: A9 03             42          LDA  #>REFS
030C: 8D F7 03          43          STA  AMPER+2    ;Hi-Byte
030F: 60                44          RTS
                        45
                        46  ** Melde UNDEFINED STATEMENTS **
                        47
0310: A5 B8             48  REFS    LDA  TXTPTR
0312: 48                49          PHA
0313: A5 B9             50          LDA  TXTPTR+1
0315: 48                51          PHA
0316: A5 67             52          LDA  TXTTAB     ;TXTTAB
0318: 85 B8             53          STA  TXTPTR     ;->
031A: A5 68             54          LDA  TXTTAB+1   ;TXTPTR
031C: 85 B9             55          STA  TXTPTR+1
031E: D0 03             56          BNE  NXTLIN1    ;unbedingt
                        57
0320: 20 88 03          58  NXTLIN  JSR  BYTGET     ;Lo-Link
0323: 20 88 03          59  NXTLIN1 JSR  BYTGET     ;Hi-Link
0326: F0 59             60          BEQ  END        ;Programmende
0328: 20 88 03          61          JSR  BYTGET     ;Lo-Line#
032B: 85 75             62          STA  CURLIN
032D: 20 88 03          63          JSR  BYTGET     ;Hi-Line#
0330: 85 76             64          STA  CURLIN+1
                        65
0332: 20 88 03          66  NXTBYT  JSR  BYTGET
0335: C9 00             67  NXTBYT1 CMP  #$00
0337: F0 E7             68          BEQ  NXTLIN
                        69
0339: C9 AB             70          CMP  #$AB       ;GOTO
033B: F0 11             71          BEQ  LINE?
033D: C9 B0             72          CMP  #$B0       ;GOSUB
033F: F0 0D             73          BEQ  LINE?
0341: C9 C4             74          CMP  #$C4       ;THEN
0343: D0 ED             75          BNE  NXTBYT     ;nein
0345: A0 01             76          LDY  #1
0347: B1 B8             77          LDA  (TXTPTR),Y
0349: 20 BA 00          78          JSR  CHRTST     ;Ziffer?
034C: B0 E4             79          BCS  NXTBYT     ;nein
                        80
034E: 20 88 03          81  LINE?   JSR  BYTGET
0351: 18                82          CLC
0352: 20 0C DA          83          JSR  LINGET
0355: 20 1A D6          84          JSR  FNDLIN
0358: B0 1D             85          BCS  FOUND      ;wenn gefunden
035A: 20 5A DB          86          JSR  OUTQST
035D: A5 51             87          LDA  LINNUM+1
035F: A6 50             88          LDX  LINNUM
```

```
0361: 20 24 ED          89          JSR  LINPRT
0364: 20 57 DB          90          JSR  OUTSP
0367: 20 57 DB          91          JSR  OUTSP
036A: 20 57 DB          92          JSR  OUTSP
036D: A5 76             93          LDA  CURLIN+1
036F: A6 75             94          LDX  CURLIN
0371: 20 24 ED          95          JSR  LINPRT
0374: 20 FB DA          96          JSR  CRDO       ;Line Feed
0377: A0 00             97  FOUND   LDY  #0
0379: B1 B8             98          LDA  (TXTPTR),Y
037B: C9 2C             99          CMP  #','       ;Aufzählung?
037D: F0 CF            100          BEQ  LINE?
037F: D0 B4            101          BNE  NXTBYT1
                       102
                       103  ** Routine abschließen **
                       104
0381: 68               105  END     PLA
0382: 85 B9            106          STA  TXTPTR+1
0384: 68               107          PLA
0385: 85 B8            108          STA  TXTPTR
0387: 60               109          RTS
                       110
                       111  ** Zeichen aus Programm holen **
                       112
0388: E6 B8            113  BYTGET  INC  TXTPTR
038A: D0 02            114          BNE  BYTGET1
038C: E6 B9            115          INC  TXTPTR+1
038E: A0 00            116  BYTGET1 LDY  #0
0390: B1 B8            117          LDA  (TXTPTR),Y
0392: 60               118          RTS

147 Bytes
```


Die ProDOS-Analyse

Version 1.0.1, 1.0.2, 1.1.1

von Arne Schäpers

1985, ca. 480 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. Dabei werden alle bisherigen Versionen von 1.0.1 bis 1.1.1 berücksichtigt. „Die ProDOS-Analyse“ beschreibt erstmals auch mehrere Programmierfehler, die bis in die neueste Version zu finden sind. 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. Durch die minutiöse theoretische Sezierung von ProDOS eröffnen sich völlig neue programmierpraktische Perspektiven.

Dr. Alfred Hüthig Verlag
6900 Heidelberg · Postfach 10 28 60

Tips und Tricks in Pascal

Teil 4: Die Compiler-Optionen

von Dieter Geiß

Ein mächtiges Werkzeug zur Programm-Optimierung stellen die Compiler-Optionen dar. Aber nicht nur zur Optimierung kann man sie gebrauchen, manche Konstruktionen werden durch sie erst möglich. Wer glaubt, er kenne alle Optionen genau, der sollte trotzdem weiterlesen, denn in diesem Artikel werden viele erstaunliche Tatsachen ans Licht gelangen, die man weder in den Handbüchern noch in anderen Artikeln finden kann.

Eine Compiler-Option tritt bekanntlich in einem Pascal-Programm in einer Kommentarklammer mit nachfolgendem „$" auf, also z.B. (*$G+*) oder {$G+}. Trifft der Compiler auf eine solche Option, wird ein spezieller Zustand eingenommen, in dem man z.B. ein um ein „Goto" erweitertes Pascal benutzen kann. Eine Option besteht meistens aus einem, manchmal auch aus mehreren Buchstaben mit einem darauf folgenden Parameter. Mehrere Optionen können mit Kommata getrennt werden, wenn der Parameter der Option kein Filename ist. Ein Leerzeichen zwischen dem „*" und dem „$" ist nicht erlaubt, ebensowenig wie zwischen der Option (z.B. „G") und dem Parameter (z.B. „+"). Genug zur Syntax, diese kann auch in den Handbüchern nachgelesen werden.

Die Compiler-Optionen werden in alphabetischer Reihenfolge mit konkreten Beispielen erklärt.

1. C – Die Comment-Option

Die C-Option kann benutzt werden, um einen Kommentar in den übersetzten Codefile zu bringen. Die Zeichenkette, die auf „C" folgt, wird in Block 0 des Codefiles geschrieben. Dafür sind genau 80 Bytes reserviert. Da es sich hierbei um einen String handelt, darf eine Länge von 79 Zeichen nicht überschritten werden (das erste Byte eines Strings beinhaltet die Länge). Die Option hat z.B. folgendes Aussehen:

(*C von Dieter Geiß, 5. August 1985*)

Die Leerzeichen, die direkt hinter dem „C" stehen, werden nicht übernommen. Die Option muß in der äußersten statischen Ebene auftreten, also im Hauptprogramm, sonst wird der Fehler 194 („Comment must appear at top of program") ausgegeben. Warum das so ist, soll für interessierte Leser ebenfalls erklärt werden.

Beim Übersetzen von Programmen wird vom Compiler eine dynamische Symboltabelle angelegt. In jeder Prozedur, die im Hauptprogramm oder in anderen Prozeduren auftritt, können dabei lokale Konstanten, Typen und Variablen erklärt werden. Diese sind außerhalb der Prozedur, in der sie definiert wurden, nicht mehr bekannt (Blockstruktur). Ist also die Übersetzung einer Prozedur beendet, kann der Compiler den Teil der Symboltabelle freigeben, in der die lokalen Größen der Prozedur definiert waren. Da die Symboltabelle auf dem Heap liegt, kann der Compiler leicht diesen Teil mit einem Release-Aufruf freigeben. Unglücklicherweise reserviert sich der Compiler aber oben auf dem Heap auch Platz für den String, der bei der C-Option erzeugt wird. Würde also kein Fehler 194 auftreten, wäre der String verloren, wenn die Option innerhalb einer Prozedur erklärt wurde.

Leider hat sich beim Compiler der Pascal-Version 1.2 ein Fehler eingeschlichen, der beim 1.1-Compiler noch nicht vorhanden war. Wenn der Compiler den Kommentar, der durch die C-Option erzeugt wurde, in den Codefile kopiert, dann vergißt er das Längenbyte des Strings, was zur Folge hat, daß der erste Buchstabe als Längenbyte interpretiert wird und dieser Buchstabe dann im Codefile fehlt. Läßt man sein Programm oder seine Unit mit dem auf der APPLE3-Diskette befindlichen LIBMAP-Programm ausgeben, so wird auf jeden Fall nicht der String ausgegeben, den man durch die C-Option eingegeben hat. Diesen Fehler kann man durch ein vorangestelltes „O" beheben. „O" hat den ASCII-Wert 79 (maximale Länge des Comment-Strings), also z.B.

(*$CO von Dieter Geiß, 5. August 1985*)

Das Längenbyte hat dann zwar immer die Länge 79, was aber nichts ausmacht, da die restlichen Bytes des Strings mit ASCII NUL aufgefüllt werden.

2. D – Die Debugger-Option

Diese Option wird in keinem Handbuch erklärt, da in der UCSD-Version II.1 (Apple Pascal 1.1 und 1.2) kein Debugger implementiert ist. Nichtsdestoweniger soll die Wirkung dieser Option erklärt werden. Eingeschaltet wird die Option mit (*$D+*), abgeschaltet mit (*$D-*). Ist die Option eingeschaltet, so nimmt der Compiler an, daß das Programm mit einem Debugger untersucht wird. In jeder Zeile, die überhaupt Code erzeugt (also keine Kommentarzeilen), wird ein P-Code-Befehl BPT (Breakpoint) mit einem anschließenden Big-Parameter erzeugt. Dieser Parameter gibt die Zeilennummer des Quellcodes an, in dem sich der Breakpoint befindet. Ein Debugger kann dann sogar die Zeilennummer im Quellcode angeben, in dem ein Breakpoint gesetzt wurde, obwohl normalerweise ein Pascal-Programm nicht zeilenorientiert ist. Damit der Programmierer sehen kann, wo seine Breakpoints gesetzt sind, wird in einem Compiler-Listing,

das durch eine L-Option (s.u.) erzeugt werden kann, der die Prozedur-Nummer und Verschachtelungstiefe trennende Doppelpunkt durch einen Stern ersetzt. Beispiel (bitte übersetzen lassen):

```
(*$D+,Q+,L #1:*)
program test;
begin
  repeat
  until false
end.
```

3. E – Die Extended-File-Option

Warum diese Option in keinem Handbuch zu finden ist, ist verwunderlich, da sie recht brauchbar ist. Eingeschaltet wird sie mit (*$E+*), abgeschaltet mit (*$E-*). Um deren Wirkung zu erklären, muß etwas weiter ausgeholt werden. Es geht um die Definitionen von Files. Jedesmal, wenn der Programmierer in seinem Programm einen File definiert, also z.B. mit

```
var F: file;
```

wird dieser vom Compiler in einer speziellen Liste eingetragen. Bevor der Code für eine Prozedur erzeugt wird, in der Files definiert wurden, wird vom Compiler Code eingefügt, um diese Files zu initialisieren. Dies ist nötig, da es verschiedene Arten von Files gibt (Interactive, Text, Untyped, Structured), aber alle Files die gleiche interne Datenstruktur haben (File Information Block). Diese Initialisierung kann vom Programmierer nicht unterbunden werden. Außerdem erzeugt der Compiler Code am Ende einer Prozedur, um alle Files wieder zu schließen. Dies ist für den Fall gedacht, daß der Programmierer es vergessen sollte. Es wird also ein implizites „close (normal)" durchgeführt. Würden Files nicht geschlossen werden, so würden sie auf Diskette als nicht geschlossene Files stehen, deren Jahr im Filedatum 100 beträgt, also eine illegale Zahl. Auf diese Files kann dann nicht mehr richtig zugegriffen werden (siehe auch „Pascal-Directory unter der Lupe", Peeker 1/85, Seite 65, Abschnitt Daterec).

Das oben Dargelegte ist auch der Grund, warum in den „Units" keine privaten Files definiert werden dürfen: Die implizite Close-Anweisung käme zu früh, da der Initialisierungsteil einer „Unit" nur einmal durchlaufen wird. Der Fehler 191 („No private Files") tritt auf, wenn ein File entweder im Implementierungsteil einer Unit oder in einer Prozedur der Unit lokal erklärt wurde. Wird er im Interface-Teil definiert, so wird bei dem Programm, das diese Unit benutzt, der Code für Initialisierung und Schließen des Files eingefügt.

Will man nun aber doch private Files in einer Unit definieren, so hält einen nichts davon ab, wenn man ein (*$E+*) in seinen Quelltext einfügt. Dann tritt kein Fehler 191 mehr auf, dafür wird aber mehr Verantwortung auf den Programmierer abgewälzt. Nach wie vor wird zwar Code für das Initialisieren der Files erzeugt, nicht jedoch für das Schließen. Dies muß dann der Programmierer übernehmen. Private Files werden z.B. in der neuen von mir erstellten „TurtleGraphics" benutzt, um Zeichensätze zu laden und Hires-Bilder zu laden und zu speichern. Ein Beispiel für private Files ist in **Listing 1** gezeigt.

4. F – Die Bytes-Flipped-Machine-Option

Wer sich beim Konfigurieren seines Systems mit Hilfe des Programms SETUP.CODE auf der APPLE3-Diskette Gedanken darüber gemacht hat, was die Frage „Has Byte Flipped Machine" zu bedeuten hat, findet hier die Antwort.

Die P-Maschine ist ein Pseudo-Maschine, die praktisch auf jedem Rechner mit einem beliebigen Prozessor emuliert werden kann. Nun zerfällt nicht bei jedem Prozessor die Adresse eines Bytes im Speicher in zwei 8-Bit-Komponenten, die sich in der Reihenfolge Low-High im Speicher befinden müssen. Das Format könnte genausogut im High-Low-Format vorliegen. Beim 6502 folgt auf einen JMP-Befehl z.B. immer zuerst der niederwertige Teil der Sprungadresse, dann der höherwertige Teil. Beim 68000 ist dies nicht der Fall. Eine Adresse wie z.B. $00FF3020 steht auch in dieser Reihenfolge im Speicher.

Um nun Programme für Maschinen zu erstellen, die ihre Adressen im High-Low-Format haben, wurde die F-Option entwikkelt. Schaltet man die Byte-Flipped-Machine-Option mit (*$F+*) ein, so wird Code erzeugt, der nur auf den entsprechenden Maschinen ablaufen kann. Beim 6502 muß diese Option ausgeschaltet sein (*$F-*), was der Normalzustand beim Compilieren ist. Läßt man sein Programm mit eingeschalteter Option übersetzen, so läuft es garantiert nicht mehr, da bei jeder Prozedur die Werte von Enter-IC, Exit-IC, Parameter-Size, Data-Size und die Jump-Table falsch interpretiert werden.

5. G – Die Goto-Option

Um zu verhindern, daß Programmierer die im Pascal-Sprachschatz enthaltene Goto-Anweisung verwenden, wird vom Compiler die Goto-Option auf (*$G-*) gestellt, also abgeschaltet. Die Verwendung von unbedingten Sprüngen ist normalerweise in Pascal, das eine Vielzahl von Anweisungen kennt, die den Programmfluß steuern, nicht nötig. Die Benutzung einer Goto-Anweisung zeugt von schlechtem Programmierstil und wird manchmal von BASIC-Programmierern benutzt, die mit der Pascal-Programmierung anfangen und noch nicht alle Anweisungen kennen oder eben einen Spaghetti-Code erzeugen wollen.

In Ausnahmefällen mag es angebracht sein, ein Goto zu benutzen, um aus einer tief verschachtelten Struktur entfliehen zu können. In diesem Fall setzt man ein (*$G+*) in sein Programm, sonst erhält man Fehler 6 („Illegal Symbol").

6. I – Die I/O-Checking- und Include-File-Option

Die I-Option wird in zwei Unteroptionen unterteilt. Die I/O-Checking-Option wird durch die Parameter +/-, die Include-File-Option durch einen Filenamen erkannt.

a) Nach jeder File-I/O-Operation, mit Ausnahme der direkten Unit-I/O-Operationen wie UNITREAD und UNITWRITE, also nach jedem READ, WRITE, RESET, CLOSE, SEEK, GET, PUT etc., wird vom Compiler der Code $9E $00 (CSP IOTEST) hinter die Operation gesetzt. Tritt bei einer I/O-Operation ein Fehler auf, so unterbricht der Interpreter den normalen Programmablauf und ruft die Exec-Error-Prozedur des Pascal-Systems auf, die eine entsprechende Fehlermeldung ausgibt. Will man dies unterbinden, so kann man das Einschieben des Codes für das I/O-Checking verhindern, indem man (*$I-*) in sein Programm einfügt. Dann sollte man auftretende Fehler aber selbst abfangen, weil sonst nicht vorhersehbare Dinge passieren könnten (Schreiben auf einen ungeöffneten File usw.). **Listing 2** zeigt eine Möglichkeit der Benutzung der I-Option.

b) Wenn ein Programm so lange wird, daß der Text nicht mehr in einem File unterzubringen ist, muß man das Programm in mehrere Teile aufspalten. In einem Hauptteil, den man dann übersetzen läßt, veranlaßt man den Compiler, einen anderen Textfile zu übersetzen, bevor er im Hauptfile weitermachen soll. Man erreicht dies durch die Angabe von (*$IFilename*), wobei „Filename" der Name des Files ist, welcher an dieser Stelle eingeschoben werden soll. Beispiel:

```
Program Test;
(*$I DECL.TEXT*)
(*$I PROCS1.TEXT*)
(*$I PROCS2.TEXT*)
begin
end.
```

Das Suffix „.TEXT" ist nicht nötig, der Compiler hängt es automatisch an. Verschachtelte Include-Files sind nicht möglich. Ebenso ist es nicht möglich, Include-Files im Interface-Teil einer Unit zu erklären. Dies liegt daran, daß der Interface-Teil blockweise in den Codefile kopiert wird. Eine Unterbrechung durch einen Include-

File mitten im Block erzeugt deswegen den fatalen Fehler 406 („Include File not legal"). Wenn man Include-Files benutzt, sollte man auch auf ein anderes Phänomen achten. Jedesmal, wenn ein File geöffnet wird, also auch bei der I-Option, muß das Directory der entsprechenden Diskette eingelesen werden. Dieses benötigt knapp zwei K und wird immer auf den Heap eingelesen. Hat der Compiler nur noch etwa 1000 Wörter Platz in der Symboltabelle, so steigt er mit einem unrühmlichen „Stack Overflow" aus. Um dies zu verhindern, sollte man auf der Diskette, die den Compiler enthält, einen File „machen" (M(ake vom Filer aus), der den Namen SYSTEM.SWAPDISK hat und vier Blöcke lang sein sollte. Wird dann der Include-File geöffnet, wird zuerst ein Teil des Heaps auf diesen File geschrieben, bevor das Directory eingelesen wird. Dies übernimmt, nicht wie im Handbuch angegeben, der Compiler, sondern geschieht in der Prozedur Fopen (entspricht Reset) im SYSTEM.PASCAL. Nach dem Öffnen des Files wird der Heap wieder auf den ursprünglichen Zustand gebracht, indem dessen Inhalt wieder aus dem File SYSTEM.SWAPDISK eingelesen wird.

7. L – Die List-Option

Um ein ausführliches Listing eines Programms zu erzeugen, welches vor allem zur Fehlersuche benutzt werden kann, muß man die L-Option benutzen, die wie die I-Option zwei verschiedene Arten von Parametern kennt.

a) (*$L+*) erzeugt einen Listing-File mit dem Namen „*SYSTEM.LST.TEXT". (*$L-*) schaltet die Ausgabe eines Listings aus, ein nachfolgendes (*$L+*) wieder ein.

b) (*$L Filename*) erzeugt einen Listing-File mit dem Namen „Filename". Ein Beispiel wäre (*$L LIST.TEXT*). Das „.TEXT" braucht nicht angehängt zu werden, dies übernimmt der Compiler. Auch hier kann man mit (*$L-*) die Ausgabe wieder abschalten, durch ein darauffolgendes (*$L+*) auch wieder einschalten. Das Listing geht dann wieder auf den zuerst angegeben Namen „Filename". Es gibt nur einen Listing-File pro Übersetzungslauf. Der Filename kann auch ein Gerät sein, wie z.B. PRINTER:, so daß ein (*$L PRINTER:*) ein Listing auf den Drucker ausgibt, ein (*$L CONSOLE:*) ein Listing auf den Bildschirm.

Das Listing hat verschiedene Spalten. In der ersten steht die Zeilennummer, dann die Segment-Nummer, die Prozedur-Nummer (vor dem Doppelpunkt bzw. dem „*"), die Befehlsverschachtelungstiefe (hinter dem Doppelpunkt bzw. dem „*") und schließlich der Wort- oder Byte-Offset innerhalb der Prozedur und der Quelltext. Erscheint anstelle der Verschachtelungstiefe ein „D", so gibt der Wort-Offset die Stelle des Speicherplatzes der Variablen an (relativ zum Markstack), ansonsten gibt der Byte-Offset das Codebyte des Befehls innerhalb einer Prozedur an. Mit diesen Werten kann ein Laufzeit-Fehler sicher gefunden werden. Ein Beispiel ist in **Listing 3** gegeben.

8. N – Die Noload-Option

Normalerweise werden Units, die von einem Programm benutzt werden, nach dem Starten des Programms sofort geladen und bleiben im Speicher, bis das Programm beendet ist. Um dies aus Speicherplatzgründen zu verhindern, muß man ein (*$N+*) direkt hinter das „begin" einer Prozedur oder des Programms gestellt werden. Dann werden Units wie Segment-Prozeduren behandelt, d.h. sie werden ausgelagert und erst dann geladen, wenn eine Prozedur dieser Unit aufgerufen wird.

9. NS – Die Next-Segment-Option

Die Segment-Nummer des Hauptprogramms ist 1, der ersten Segment-Prozedur oder der ersten normalen (Nicht-Intrinsic-)Unit ist 7. Weitere Segment-Nummern werden vom Compiler in aufsteigender Reihenfolge vergeben, d.h. nach 7 kommt 8 usw. bis 31 (Pascal 1.1) bzw. 63 (Pascal 1.2). Nehmen wir an, ein Programm habe 10 Segment-Prozeduren, nämlich von 7 bis 16. Außerdem benutze es eine Intrinsic-Unit mit der Segment-Nummer 16. Dies ist nicht möglich, da Segment-Nummern nicht doppelt verkommen dürfen. Die Lösung besteht darin, vor der letzten Segment-Prozedur die NS-Option einzusetzen:

```
(*$NS 17*)
segment procedure SegProc;
```

Die Segment-Prozedur bekommt dann die Nummer 17, der Konflikt wird vermieden.

10. P – Die Page-Option

Will man einen Seitenvorschub auf seinen Listing-File angeben, so schreibt man die parameterlose Option (*$P*) in den Programmtext. Diese Zeile erscheint dann als erste auf der neuen Seite.

11. Q – Die Quiet-Compile-Option

Wird mit (*$Q+*) die Q-Option eingeschaltet, so werden verschiedene Compiler-Meldungen unterdrückt. Dazu gehören die Zeilennummern mit den Punkten und die Namen der Prozeduren. Tritt ein Fehler auf, so wird nicht die Zeile „<sp> to continue..." ausgegeben. Diese Option wird gern in Verbindung mit einem Listing-File auf den Bildschirm in Verbindung gebracht, also (*$Q+,L CONSOLE:*), weil dann die sonstigen Meldungen des Compilers nicht die Ausgabe des Listing-Files stören. Man bekommt also ein übersichtlicheres Bild. Allerdings stoppt der Compiler nicht mehr bei einem Fehler, wenn bei eingeschalteter Q-Option ein Listing-File ausgegeben wird. Eine Ausnahme bilden die Fehler über 400, die als fatal angesehen werden. Beim Auftreten eines solchen Fehlers stoppt der Compiler immer die Übersetzung.

12. R – Die Range-Check- und Resident-Option

Wie die I-Option zerfällt auch die R-Option in zwei verschiedene Unteroptionen.

a) Normalerweise erzeugt der Compiler nach jedem Array- und String-Zugriff sowie nach Zuweisungen an Variablen eines Unterbereichs Code, der zur Laufzeit die Gültigkeit von Array- und String-Indizes bzw. den Wert eines Ausdrucks überprüft, der an eine Subset-Variable zugewiesen werden soll. Dieser Code kostet relativ viel Speicherplatz, macht aber das Programm vor unliebsamen Überraschungen sicher. Im Grunde kann man ja nie sicher sein, daß man nicht doch einmal auf das 101. Element eines 1 . . . 100-Arrays zugreift, dennoch kann es angebracht sein, das Range-Checking mit einem (*$R-*) auszuschalten. Ein Beispiel dafür sei folgende Aufgabe: Man erstelle eine String-Variable, die 132 „=" -Zeichen – etwa zum Unterstreichen – beinhaltet. Die Lösung ist in **Listing 4** dargestellt.

b) Die R-Option ist in den Handbüchern nicht vollständig erklärt. Diese Option ist wichtig für das Memory-Management zur Laufzeit. Damit kann man Programme erzeugen, die sich selbst ein- und auslagern, wie es auch der Compiler bei der S-Option tut.

Es geht wieder einmal um Segmente. Ein Segment ist z.B. das Hauptprogramm, alle seine Segment-Prozeduren, normale Units und Intrinsic-Units, die vom Hauptprogramm benutzt werden. Jedes Segment hat seine eigene Segment-Nummer, das Hauptprogramm die Nummer 1, Segment-Prozeduren meistens ab 7 aufwärts usw. Alle Segmente können ausgelagert werden, d.h. der Code eines Segments wird nur geladen, wenn dieses Segment aufgerufen wird. Während Units grundsätzlich bei Beginn des Programms geladen werden, sind Segment-Prozeduren immer ausgelagert. Das Einlagern der Units kann aber auch, wie schon oben erwähnt, durch die N-Option verhindert

werden. Angenommen, es sind jetzt alle Segmente ausgelagert: Dann kann mit der R-Option entschieden werden, welche Segmente eingelagert werden sollen. Dies kann auch in Abhängigkeit von vorhandenem Speicherplatz geschehen, wenn man geschickt programmiert. Auch der Compiler schickt sich selbst in den Swapping-Modus, wenn er weniger als 19900 Bytes freien Speicher vorfindet.

Um das Einlagern eines Segments zu erzwingen, stellt man ein (*$RSegment-Name*) an den Anfang des Rumpfes der Prozedur, die dieses Segment einlagern soll. Nach Ablauf der Prozedur wird dieses Segment wieder ausgelagert. Ein ausführliches Beispiel ist in **Listing 5** gegeben. Was nicht in den Handbüchern steht, ist eine andere Möglichkeit der Syntax der R-Option. Man muß nämlich nicht unbedingt den Segment-Namen angeben, es reicht schon die Segment-Nummer, also (*$R 7*), was völlig neue Möglichkeiten eröffnet. Nun können auch Segmente, deren Namen nicht bekannt sind, z.B. Segment-Prozeduren in Prozeduren von der äußersten Ebene eingelagert werden, und dies kann man dann natürlich auch mit Betriebssystem-Segmenten tun.

Ein Beispiel: Ein Programm soll mit Files arbeiten und unabhängig von der vom System ein- oder ausgeschalteten S(wapping-Option (Kommando-Ebene) arbeiten. Wie man weiß, muß bei eingeschalteter S(wapping-Option die Boot-Diskette immer im Boot-Laufwerk stecken, damit SYSTEM.PASCAL immer online ist. Das liegt daran, daß beim Einschalten des S(wapping Teile dieses SYSTEM.PASCAL ausgelagert werden. Es ist dies das Segment #6 (FILEPROCS) und bei Pascal 1.2 eventuell noch das Segment #2 (FIOPRIMS), da dort ein S(wapping der zweiten Stufe eingeschaltet werden kann. Da einem Programm nicht die Bezeichner der beiden Segmente bekannt sein können (außer, wenn man es als System-Programm übersetzen läßt), kann es nicht eine Option (*$R FILEPROCS*) anwenden, wohl aber eine Option (*$R 6*). Das Programm könnte also so aussehen wie in **Listing 6**. Wenn ein Segment, das bereits im Speicher ist, mit der R-Option noch einmal eingelagert wird, so macht das nichts aus, das System erkennt dies und lädt nicht noch einmal nach. Will man mehrere Segmente einlagern, kann man deren Namen oder Nummern durch Kommata trennen, also z. B. (*$R 2, 6*).

13. S – Die Swapping-Option

Wenn Programme mit vielen Deklarationen oder Units übersetzt werden sollen, dann wird der Speicherplatz für die Symboltabelle des Compilers knapp. Um dies zu verhindern, kann man den Compiler dazu veranlassen, Teile seiner Segmente auf Diskette auszulagern und bei Bedarf nachzuladen. Schaltet man mit (*$S+*), das am Anfang des Programms oder der Unit stehen muß, den Swapping-Modus des Compilers ein, so werden nicht alle der 15 Segmente des Compilers bei dessen Start eingelagert. Beim Einschalten von (*$S++*) wird sogar noch weniger eingelagert. Wieviel Platz damit im einzelnen gespart wird, hängt von der Version des Compilers ab. Aufschluß darüber gibt die folgende Tabelle:

	(*S+*)	(*S++*)
Pascal 1.1	5285	6743 words
Pascal 1.2	5311	6765 words

Eingelagert werden ohne Swapping die Segmente DECLARATIONPART, BODYPART, NUMSTRING, STATEMENT, CASESTATEMENT, FORSTATEMENT, BODY1, BODY3 und ROUTINE. Bei einfachem Swapping werden nur noch ROUTINE und STATEMENT eingelagert, bei Doppel-Swapping keine der Segmente. Bei der 128K-Version von Pascal 1.2 muß der Swapping-Zustand nie eingeschaltet werden, da damit kein weiterer Platz für die Symboltabelle freigegeben werden kann. Man hat dort ohnehin knapp 40K für die Symboltabelle zur Verfügung.

14. T – Die Tiny-Pascal-Option

Um weiteren Speicherplatz für die Symboltabelle während des Übersetzens einzusparen, wurde die T-Option eingeführt. Der Compiler, der selbst aus knapp 10000 Zeilen Pascal-Quelltext besteht, konnte auf den kleinen Maschinen von früher kaum übersetzt werden. Da der Compiler aber nicht alle Standardprozeduren benutzt, braucht er auch nicht alle zu kennen. Setzt man (*$T+*) an den Anfang eines Programms, so werden verschiedene Prozeduren gar nicht erst vom Compiler initialisiert, d.h. sie sind dann nicht bekannt. Tiny-Pascal ist sozusagen eine Untermenge von Pascal. Zu den nicht bekannten Prozeduren gehören: READLN, PRED, SQR, UNITREAD, INSERT, DELETE, COPY, POS, SEEK, GET, PUT, PAGE, STR, GOTOXY, UNITSTATUS. Ohne diese Prozeduren hat man weitere 165 Worte mehr Platz für die Symboltabelle.

15. U – Die User-Program- und Use-Library-Option

Auch diese Option zerfällt in zwei gänzlich verschiedene Unteroptionen.

a) Die Erklärung der User-Program-Option im Handbuch ist sehr unbefriedigend. Dort steht, man solle die Option nicht benutzen, wenn man nicht weiß, wie sie genau funktioniert. Doch schweigt sich das Handbuch über deren Anwendung aus.

Für die Erklärung dieser Option muß man etwas weiter ausholen. Jedes Programm wird in sog. Segmente zerlegt. Das Hauptprogramm hat immer die Segment-Nummer 1. Das System benutzt die Nummern 0 und 2 bis 6. Mit „dem System" ist das Programm im File SYSTEM.PASCAL gemeint, das beim Booten nach dem Einladen des Interpreters im File SYSTEM.APPLE geladen wird. Dieses System ist in Pascal geschrieben und hat die Segment-Nummer 0. Da es relativ groß ist, wurde es noch einmal aufgeteilt und belegt noch die Segment-Nummern 2 bis 6. Die Segment-Nummer 1 heißt USERPROGRAM und ist eigentlich das normale Hauptprogramm. Nur einmal nach dem Booten kann dieses USERPROGRAM aufgerufen werden, wenn man sofort den Befehl U(ser restart gibt. Sonst wird beim Ausführen eines normalen Programms immer die Segment-Tabelle des Interpreters gefüllt, so daß ein normales User-Programm immer in Segment 1 der Segment-Tabelle geladen wird. Das SYSTEM.PASCAL selbst und alle anderen System-Programme wurden aber mit einem (*$U-*) vor dem Programmkopf übersetzt. Dies zwingt den Compiler, nicht mit der Segment-Nummer 1 anzufangen, sondern mit der Segment-Nummer 0! Dadurch ist es möglich, ein eigenes Pascal-System (SYSTEM.PASCAL) zu schreiben, oder Programme zu schreiben, die auf Systemvariablen zugreifen. In **Listing 7** ist das Gerippe des Files SYSTEM.PASCAL dargestellt, das mit (*$U-*) übersetzt werden muß (siehe auch Apple Pascal Operating System Reference Manual, Seite 263 ff.). Zu beachten ist dabei folgendes: Die erste Variable im Pascal-System muß ein Zeiger auf den sog. Syscomrec (System Communication Area Record) sein, der vom Interpreter beim Booten automatisch gesetzt wird. Die erste Nicht-Segment-Prozedur muß die Exec-Error-Prozedur sein, die vom Interpreter angesprungen wird, wenn ein Laufzeitfehler auftritt (siehe Apple Pascal Operating System Reference Manual, Seite 226 ff.).

Listing 8 gibt ein Beispiel eines eigenen winzig kleinen Pascal-Systems. Es dürfen nur Standardprozeduren benutzt werden, also kein READ, READLN etc., denn diese müssen erst im SYSTEM.PASCAL definiert werden. Der Compiler übersetzt solche Prozeduren als CXP 0,n (Call eXternal Procedure Segment 0, Procedure n), der Compiler und das SYSTEM.PASCAL zie-

hen sich sozusagen gegenseitig hoch. Um das Listing 8 zum Laufen zu bringen, muß der erzeugte Codefile in SYSTEM.PASCAL umbenannt werden und dann frisch gebootet werden. Nach Ablauf des Hauptprogramms des Pascal-Systems wird immer ein Kaltstart ausgeführt (P-Code-Befehl XIT am Ende des Programms).
Wenn man mit (*$U-*) arbeitet, werden zusätzlich folgende Optionen gesetzt: (*$R-,I-,G+,V-*). Außerdem ist es erlaubt, nicht aufgelöste Forward-Referenzen zu benutzen. Die Units LONGINTIO (Long Integer I/O) und PASCALIO (Real I/O) werden beim Benutzen von Long Integers und Reals nicht automatisch eingelagert. Will man also in einem System-Programm Long Integers oder Real I/O benutzen, so muß man diese Segmente selbst mit Hilfe der R-Option für resident erklären. Durch (*$U-*) ist es außerdem möglich, Zeiger auf Files zu definieren, was sonst nicht erlaubt ist, also ein Konstrukt type Filep = ↑text. Variablen vom Typ Filep werden natürlich nicht wie gewöhnliche Files initialisiert. Man sollte mit solchen Zeigervariablen also nur auf schon initialisierte Files zugreifen, wie INPUT, OUTPUT oder die Workfiles.
Eine weitere Möglichkeit der Benutzung der U-Option ist das Zugreifen auf Systemvariablen. **Listing 9** zeigt, wie man von einem normalen Programm auf Systemvariablen zugreift. Dieses Programm simuliert die Umgebung des Pascal-Systems, läuft aber als normales User-Programm ab, das nach dem Übersetzen durch R(un bzw. X(ecute gestartet werden kann. Es gibt das Systemdatum aus, das im Filer mit dem D(ate-Kommando gesetzt werden kann.
Ähnlich kann man auch Units schreiben, die auf Systemvariablen zugreifen, ohne daß man dies von außen merkt. Die Unit CHAINSTUFF ist ein solches Beispiel. In diesem Fall muß man die Unit in eine Programm-Umgebung einbetten und das ganze mit (*$U-*) übersetzen lassen. Nur durch die U-Option ist es möglich, zuerst Variablen zu definieren und dann die Unit. Eine Unit, die auf das Systemdatum zugreift, ist in **Listing 10** gezeigt. Das „Daterec" ist dort deswegen im Interface-Teil nochmals definiert, damit es durch ein Uses in ein Programm übernommen werden kann.
b) Sollen Units benutzt werden, die sich nicht in der SYSTEM.LIBRARY befinden, so muß dies mit der U-Option dem Compiler bekannt gemacht werden. Dazu plaziert man ein (*$UFilename*) vor die Units in der Uses-Liste, also z.B.
uses (*$U #5:TURTLE.LIBRARY*)
TurtleGraphics, TurtleIO, TurtleRealIO;
Der Compiler sucht dann auf der Diskette in Laufwerk #5: nach dem File TURTLE.LIBRARY und lädt von dort die Interface-Texte der Units TurtleGraphics, TurtleIO und TurtleRealIO. Danach können keine Units mehr benutzt werden, die sich in der SYSTEM.LIBRARY befinden, es sei denn man gibt dann wieder ein explizites (*$U SYSTEM.LIBRARY*) an.

16. V – Die Var-Parameter-Option

Wenn der formale Parameter einer Prozedur ein String ist und die Prozedur mit einem aktuellen Parameter aufgerufen wird, dessen String-Länge kleiner als die des formalen Parameters ist, tritt der Fehler 175 („Actual parameter max string length < var formal max length") auf. Beispiel:

```
program Test;
var S: string [10];
procedure X (var T: string);
begin
   fillchar (T, size_of (T), ' ')
end;
begin
   S := 'ABC';
   X (S) (* Fehler 175 *)
end.
```

Warum der Fehler auftreten muß, ist klar. In der Prozedur X werden insgesamt 82 Bytes des Strings T mit einem Leerzeichen gefüllt. Der Parameter S, der die Stelle von T beim Aufruf X (S) übernimmt, ist aber nicht 82 Bytes, sondern nur 12 Bytes groß. Würde kein Fehler auftreten, würden einige Speicherzellen mit dem Wert 32 (ASCII SP) gefüllt, die adressenmäßig hinter dem String S liegen. Dies ist sicher nicht im Sinne des Programmierers. Trotzdem kann man dies erreichen, wenn man (*$V-*) in seinem Programm einfügt. Dann wird das Auftreten des Fehlers 175 unterdrückt. In dem oben genannten Beispiel würde man allerdings undefinierte Speicherzellen löschen. In anderen Fällen könnte aber die V-Option zum Vorteil eingesetzt werden. Wenn man in der Prozedur X z.B. über die Länge des Strings geht, die ja die definierte Länge nie überschreiten kann, also z.B.

```
procedure X (var T: String);
begin
   fillchar (T (1), length (T), ' ')
end;
```

Die Länge von T muß allerdings vorher bekannt sein. Im obigen Beispiel ist sie 3, da eine Zuweisung S := 'ABC' stattgefunden hat.

Weitere Compiler-Optionen sind in Pascal 1.1 und 1.2 nicht definiert. Im nächsten Teil wollen wir uns mit der Möglichkeit befassen, von Maschinensprache aus in Pascal Ein- und Angabeoperationen durchzuführen.

ProDOS-Editor 1.0

Applesoft-Editor
unter ProDOS-Betriebssystem

von U. Stiehl

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

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

Unter anderem sind folgende Features implementiert worden:

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

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

Gerätevoraussetzung: Apple II+, IIe oder IIc, 40 Zeichen/Zeile

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

Alle Optionen auf einen Blick

Die mit "*" gekenzeichneten Optionen sind die vom Compiler initialisierten Werte (Default values). Am Anfang einer Übersetzung hat der Compiler also folgende Optionen gesetzt:
(*$D-,E-,F-,G-,I+,L-,N-,Q-,R+,S-,T-,U+,V+*)

C <Text>	Der String <Text> wird in Block Ø des Codefiles gespeichert.
D+	Schaltet die Erzeugung von Breakpoints ein.
*D	Schaltet die Erzeugung von Breakpoints ab.
E+	Erlaubt private Files in Units.
*E-	Verbietet private Files in Units.
F+	Schaltet die Byte-Flipped-Machine ein (High vor Low).
*F-	Schaltet die Byte-Flipped-Machine aus (Low vor High).
G+	Erlaubt Goto-Befehle.
*G-	Verbietet Goto-Befehle.
*I+	Der Compiler erzeugt Code für I/O-Checking.
I-	Der Compiler erzeugt keinen Code für I/O-Checking.
I<Filename>	Fügt bei der Übersetzung den Quellcode <Filename> ein.
L+	Ein Listing-File mit Namen SYSTEM.LST.TEXT wird erzeugt.
*L-	Kein Listing-File wird erzeugt.
L<Filename>	Ein Listing-File mit Namen <Filename> wird erzeugt.
N+	Alle benutzten Units werden ausgelagert.
*N-	Die benutzten Units werden nach dem Starten sofort geladen.
NS <Nummer>	Das nächste zu erzeugende Segment hat die Nummer <Nummer>.
P	Ein Seitenvorschub wird im Compilerlisting erzeugt.
Q+	Gewisse Bildschirmmeldungen werden unterdrückt.
*Q-	Prozedur-Namen und Zeilennummern werden ausgegeben.
*R+	Der Compiler erzeugt Range-Checking-Code.
R-	Der Compiler erzeugt keinen Range-Checking-Code.
R<Filename>	Lädt ein Segment mit Namen <Filename>.
R<Nummer>	Lädt ein Segment mit Segment-Nummer <Nummer>.
S+	Der Compiler lagert einige seiner Segmente aus.
S++	Der Compiler lagert noch mehr Segmente aus.
*S-	Fast alle Segmente des Compilers werden eingelagert.
T+	Tiny-Pascal-Übersetzung.
*T	Normale Pascal-Übersetzung.
*U+	Es werdem User-Programme übersetzt.
U-	Es werden System-Programme übersetzt.
U<Filename>	Die folgenden Units werden in der Library <Filename> gesucht.
*V+	Checking für VAR-Parameter von Strings wird eingeschaltet.
V-	Kein Checking für VAR-Parameter von Strings.

Demoprogramme

(Listing 1)

```
{$S+,E+}
unit Private; intrinsic code 23 data 24;

interface

procedure Dummy;

implementation

var F : file; {Ohne E+ nicht moeglich}

procedure Dummy;

var G : file; {Ohne E+ nicht moeglich}

begin {Dummy}
  reset (G, 'Dummy');
  close (G, normal); {close notwendig}
end; {Dummy}

begin {Private}
  rewrite (F, 'Dummy');
  close (F, lock)    {close notwendig}
end {Private}.
```

{——}

(Listing 2)

```
program I_Test;

var IO : integer;
    F  : file;
    S  : string;

begin
  write ('Name: ');
  readln (S);
  {$I-}          {Kein I/O-Checking}
  reset (F, S);
  if IOresult <> Ø then {File nicht da,}
  begin          {vielleicht mit Suffix}
    insert ('.TEXT', S, length (S) + 1);
    reset (F, S)
    {$I+}        {wieder einschalten}
  end; {if}
  IO := IOresult;
  write ('Textfile ', S);
  if IO <> Ø then write (' nicht');
  writeln (' gefunden.');
  close (F)
end.
```

{——}

(Listing 3)

```
{$Q+,L CONSOLE:}
program L_Test;

var I : Ø..9;

begin
  writeln ('Gleich kommt ein Fehler');
  I := 1Ø;{Value Range Error S#1 P#1 I#48 (48 zw. 45 und 51)}
  writeln ('Hier komme ich nicht mehr hin')
end.
```

{——}

(Listing 4)

```
program R_Test;

const Leng = 132;

var   S    : string [Leng];

begin
  fillchar (S [1], Leng, '=');
  {$R-}
  S [Ø] := chr (Leng);
  {$R+}
  unitwrite (1, S [1], 8Ø);     {8Ø '=' auf Bildschirm}
  {unitwrite (6, S [1], 132)}   {132 '=' auf Drucker}
end.
```

{——}

(Listing 5)

```
program R_Test;

uses TurtleGraphics;

var C : char;

segment procedure Init;

{Initialisierungen}

begin {Init}
  InitTurtle
end; {Init}

segment procedure DoIt;
```

```
{Irgendwelche Aktionen}

var I : integer;

begin {DoIt}
  {$R TURTLEGRAPHICS} {sonst wird 41mal TurtleGraphics geladen}
  MoveTo (Ø, 1ØØ);
  for I := 1 to 4Ø do WChar ('X')
end; {DoIt}

procedure Main;

{Hauptprogramm}

begin {Main}
  Init;
  DoIt;
  TextMode;
  writeln (memavail, ' words')
end; {Main}

procedure SwapIn;

{Alle aufgezaehlten Prozeduren werden eingelagert}

begin {SwapIn}
  {$R TURTLEGRAPHICS}   {Alles einlagern}
  {$R INIT}
  {$R DOIT}
  Main
end; {SwapIn}

begin {R_Test}
  {$N+}          {Alles auslagern}
  write ('Einlagern (j/n)? ');
  read (C);
  writeln;
  if C in ['N', 'n']
  then Main
  else SwapIn
end {R_Test}.
```

{——}

(Listing 6)

```
program R_Test;

{Vor dem Ausfuehren dieses Programms bitte}
{S(wapping von der Kommandoebene einschalten.}

var F : file;

begin
  {$R 2, 6} {Hier Boot-Diskette noch in Boot-Laufwerk}
  writeln ('Bitte Boot-Diskette herausnehmen');
  writeln ('und andere Pascal-Diskette einschieben!');
  write ('Dann <return> druecken!');
  readln;
  rewrite (F, '#4:XXX'); {Ohne die R-Option wuerde...}
  close (F)              {...das Programm an dieser...}
end.                     {...Stelle abstuerzen.}
```

{——}

(Listing 7)

```
{$U-}
program Pascalsystem;

{Rumpf SYSTEM.PASCAL Version 1.1, vgl. Apple Pascal
 Operating System Reference Manual, Seite 263 ff.}

type Syscomrec = record
                   IOrslt  : integer;
                   XeqErr  : integer;
                   SysUnit : integer;
                   {usw. siehe Apple Pascal
                    Operating System Reference
                    Manual, Seite 225 ff.}
                 end; {Syscomrec}
     Fibp      = ↑Fib;
     Fib       = file; {normalerweise record...}

var  Syscom    : ↑Syscomrec;
     {usw.}

segment procedure Userprogram (input, output : Fibp);
{Ein normales Programm ist also nur eine Prozedur
 im Pascalsystem, deren Segmente in GetCmd in die
 Segment-Tabelle im Syscomrec gefuellt werden.}

begin {Userprogram}
end; {Userprogram}

segment procedure Debugger;

{Nicht implementiert in Pascal 1.1,
 in Pascal 1.2 FIOPRIMS}

begin {Debugger}
end; {Debugger}

segment procedure PrintError (XeqErr, IOrslt : integer);

{Fehlermeldungen wie "Value Range Error" etc.}

begin {PrintError}
end; {PrintError}

segment procedure Initialize;

{I(nitialize von Kommando-Ebene}

begin {Initialize}
end; {Initialize}

segment function GetCmd (LastSt : integer) : integer;

{Kommandozeile etc.}

begin {GetCmd}
end; {GetCmd}

segment procedure FileProcs {6 Parameter...};

{Segment zum Auslagern bei S(wapping von der
 Kommando-Ebene (reset, rewrite, close etc.)}

begin {FileProcs}
end; {FileProcs}

procedure ExecError;

{Muss Segment #Ø, Prozedur #2 sein}

begin {ExecError}
end; {ExecError}

begin {Pascalsystem}
  Initialize;
  {usw.}
end {Pascalsystem}.
```

{——}

(Listing 8)

```
{$U-}
program Pascalsystem;
{Ein eigenes kleines Pascalsystem. Ein wesentlich
 groesseres muesste gespalten werden, damit es auf
 die 16K-Karte passt.}

type Fibp    = ↑Fib;
     Fib     = file;

var  Syscom  : ↑integer;
     FF      : char;
     CR      : char;

procedure ExecError;
forward;

procedure Fwritestr (S : string);
forward;

procedure Freadln;
forward;

procedure ExecError;

{Muss Segment #Ø, Prozedur #2 sein}

begin {ExecError}
  Fwritestr ('Es ist ein Fehler aufgetreten!');
  Freadln
end; {ExecError}

procedure Fwritestr (S : string);

{'Fwritestr' ersetzt writeln, welches der Compiler
 als Prozedur 19 adressieren wuerde, die hier
 nicht existiert.}
```

```
begin {Fwritestr}
  unitwrite (1, S [1], length (S));
  unitwrite (1, CR, 1)
end; {Fwritestr}

procedure Freadln;

{'Freadln' ersetzt readln, welches der Compiler
 als Prozedur 21 adressieren wuerde, die hier
 nicht existiert.}

var C : char;

begin {Freadln}
  C := chr (Ø);
  repeat
    unitread (2, C, 1)
  until C = CR
end; {Freadln}

procedure Clearscreen;

begin {Clearscreen}
  unitwrite (1, FF, 1)
end; {Clearscreen}

begin {Pascalsystem}
  FF := chr (12);
  CR := chr (13);
  Clearscreen;
  Fwritestr ('Dies ist ein eigenes Pascalsystem');
  Fwritestr ('Bei Druecken von <return> wird ein');
  Fwritestr ('Kaltstart durchgefuehrt.');
  Freadln
end {Pascalsystem}.
```

{——}

(Listing 9)

```
{$U-}
program Pascalsystem;

type Daterec = packed record
                 Month : Ø..12;
                 Day   : Ø..31;
                 Year  : Ø..1ØØ
               end; {Daterec}

     Fibp    = ↑Fib;
     Fib     = file;

var  Fill    : array [1..66] of integer;
     TheDate : Daterec; {Das 67. Wort}

segment procedure Userprogram (input, output : Fibp);

{Programm kann mit X(ecute oder R(un gestartet werden}

begin {Userprogram}
  write ('Heute ist der ');
  with TheDate do write (Day, '. ', Month, '. 19', Year)
end; {Userprogram}

begin {Pascalsystem}
end {Pascalsystem}.
```

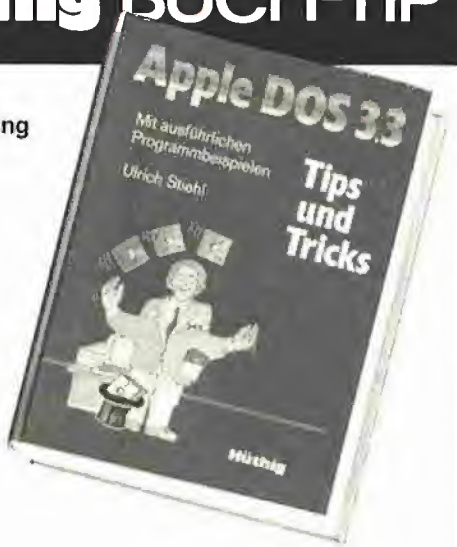
{——}

(Listing 10)

```
{$U-}
program Pascalsystem;

type Daterec = packed record
                 Month : Ø..12;
                 Day   : Ø..31;
                 Year  : Ø..1ØØ
               end; {Daterec}

     Fibp    = ↑Fib;
     Fib     = file;

var  Fill    : array [1..66] of integer;
     TheDate : Daterec; {Das 67. Wort}

{Units nach 'var' nur moeglich mit (*$U-*)}

unit Date; intrinsic code 23;

interface

type Daterec = packed record
                 Month : Ø..12;
                 Day   : Ø..31;
                 Year  : Ø..1ØØ
               end; {Daterec}

procedure GetDate (var D : Daterec);
procedure SetDate (D : Daterec);

implementation

procedure GetDate (var D : Daterec);

begin {GetDate}
  D := TheDate
end; {GetDate}

procedure SetDate (D : Daterec);

begin {SetDate}
  TheDate := D
end; {SetDate}

begin {Date}
end; {Date}

begin {Pascalsystem}
end {Pascalsystem}.
```


Hüthig BUCH-TIP

3. Auflage in Vorbereitung



Apple DOS 3.3 — Tips und Tricks

von U. Stiehl
3., überarb. Aufl. 1985, ca. 230 S., mit zahlreichen, ausführlich kommentierten Programmlistings, kart., ca. DM 28,–

Dr. Alfred Hüthig Verlag · Postf. 10 28 69 · 6900 Heidelberg 1

Utilities von Beagle Brothers

getestet von Franz-Josef Hüskens

Im Peeker, Heft 6/85, Seite 77 wurden bereits die beiden Programme DOUBLE TAKE und GPLE, die beide von der Firma Beagle Brothers angeboten werden, vorgestellt.

In diesem Artikel sollen nun einige „Programmsammlungen" bzw. Programme der gleichen Firma vorgestellt werden. Diese Programme sind mehr oder weniger nützlich, zu einem sehr geringen Teil sogar total unnütz. Auf den Disketten befinden sich teilweise auch Spiele und Demonstrationsprogramme von anderen Disketten der Firma Beagle Brothers. Fast alle Programme sind listbar und können damit nicht nur vom Benutzer eingesehen, sondern – falls gewünscht – auch geändert werden. Das Listen und spätere Analysieren einzelner Applesoft-Programme ist jedem, der selbst programmiert und eine Diskette der Beagle Brothers besitzt, sehr zu empfehlen. Man sieht hier meist eine sehr „clevere" Programmierung. Applesoft-Befehle werden in ungeahnter Weise ausgenutzt und bringen „Amateur"-Programmierer auf vollkommen neue Ideen. Da manche Programme sehr ansprechende Menüs haben, erhält man auch dahingehend Tips für eigene Programme.

Doch nun zu den einzelnen Disketten und deren Inhalten:

1. Utility City

Diese DOS-3.3-Diskette enthält die folgenden 21 Utilities (die Reihenfolge entspricht der im Handbuch und ist keine Wertung):

1) **Filename Zap** verändert jeden beliebigen File-Namen so, daß der Name im Catalog entweder unsichtbar ist oder inverse, blinkende oder normale Zeichen in beliebiger Folge enthält.

2) **Command Zap** ermöglicht die „Eingabe" von unsichtbaren Befehlen oder REM-Kommandos in ein Applesoft-Programm.

3) **Line Search** erlaubt das Auffinden der Programmzeilen eines Applesoft-Programms. Die Beschreibung zu diesem Programm beinhaltet auch eine kurze Abhandlung über die Organisation der Programmspeicherung.

4) **Screenwriter**: Mit dieser Utility können Bildschirm-Layouts, z.B. für Menüs entworfen werden. Es ist möglich, über die Tastatur in den drei Anzeige-Modi (INVERSE, FLASH, NORMAL) direkt Text in Groß- oder Kleinschreibung zu erzeugen. Wörter oder ganze Textblöcke können verschoben oder zentriert bzw. links- oder rechtsbündig formatiert werden. Fertige Layouts können bildschirmweise auf Diskette gespeichert oder von Diskette geladen werden. Die Ausgabe über einen Drucker ist natürlich auch möglich.

5) **Int Converter** „übersetzt" Integer-BASIC-Programme in Applesoft-Programme, allerdings ohne die Syntax zu korrigieren. Die Bedienungshinweise zu dem Programm enthalten jedoch eine Gegenüberstellung der Integer- und Applesoft-Befehle, die noch „von Hand" berichtigt werden müssen.

6) **Bfind** ist ein EXEC-File, der die Startadresse und die Länge des zuletzt geladenen Binär-Files anzeigt.

7) **Sortfile**: Dieses Programm ist zum „Ausbau" gedacht. Es gestattet das Sortieren und Speichern von eindimensionalen Feldern.

8) **Connect** vereinigt zwei kleine Applesoft-Programme zu einem großen.

9) **Text Dump** gibt den aktuellen Textbildschirminhalt auf dem Drukker aus. Das Programm arbeitet alleine oder als Teil eines anderen Programms.

10) **Rem Zap** ermöglicht die inverse Anzeige von REM-Statements. Dazu wird allerdings noch das Programm **Rem Find** benötigt. Nach Eingabe von PR#0 oder Betätigen der RESET-Taste werden die inversen Statements unsichtbar.

11) **Address Checker** gestattet eine „Expedition" durch den Apple-Speicher. Ein vom Benutzer ausgewählter Adreßbereich wird auf Unterschiede des Speicherinhalts zu zwei verschiedenen Zeitpunkten untersucht. Geänderte Inhalte werden invers angezeigt.

12) **Multi Cat** gestattet die Ausgabe des Disketten-Catalogs in beliebiger Spaltenzahl (3, 4, 5 Spalten usw.) und in beliebigem Seitenformat.

13) **Key Cat** erlaubt die Auswahl der Programme einer Diskette mit einem Tastendruck. Die gewählten Programme werden automatisch gestartet. Zusätzlich zeigt das Programm den freien Speicherplatz einer Diskette an.

14) **Kill Cat**: Ctrl-C oder eine beliebig wählbare Taste unterbricht einen langen Disketten-Catalog. Jeder andere Tastendruck startet die Catalog-Ausgabe erneut.

15) **Double Loader**: Mit einem Textfile, der bei jeder Benutzung neu erstellt wird, kann ein Applesoft-Programm durchgeführt werden, während ein anderes BASIC-Programm im Speicher verbleibt.

16, 17) **Run Counter** und **Run Dater**: Beide Programme sollten einem eigenen Applesoft-Programm vorangestellt werden. Run Counter zählt die Anzahl der Programmbenutzungen, zeigt sie an, vermerkt diese Zahl im Programm und speichert dieses dann auf Diskette. Run Dater arbeitet genauso mit der Ausnahme, daß nicht die Anzahl der Programmläufe gespeichert wird, sondern das Datum der Benutzung.

18) **Chr$ Poker** zeigt auf Wunsch die Speicheradresse jeder Bildschirmzelle und den ASCII-Wert jedes eingegebenen Zeichens an.

19) **Bigliner** gestattet das Numerieren einzelner Programmzeilen eines Applesoft-Programms mit der Zeilenzahl 65535.

20) **Hex, Dec & Dex** ermöglicht Zahlenkonvertierung von dezimal in hexadezimal und umgekehrt sowie die Umwandlung in Binärzahlen. Dabei können sowohl negative als auch positive Zahlen eingegeben werden.

21) **Xlister** formatiert Applesoft-Programme so, daß sie leichter lesbar werden: Jeder Befehl einer Programmzeile wird in einer eigenen Zeile dargestellt, Befehle innerhalb einer FOR-NEXT-Schleife werden eingerückt. Die Ausgabe ist auf den Bildschirm oder über Drucker möglich, wobei die Spaltenzahl frei wählbar und auch Seitenumbruch auf dem Drucker möglich ist.

Die Benutzung der Programme wird in einem 20seitigen englischsprachigen Handbuch erklärt. Das Gesamtpaket „Utility City" bietet auf den ersten Blick zwar eine Unmenge von Programmen, jedoch muß dabei berücksichtigt werden, daß verschiedene Programme nur umständlich zu handhaben sind oder andere Utilities nur dann benutzt werden können, wenn der entsprechende File auf der momentan benutzten Diskette vorliegt. Eine bessere Lösung wären hier entsprechende Routinen in Maschinensprache, die jederzeit im Speicher präsent sind.

Ich selbst benutze zwei Programme (Xlister und Screenwriter) regelmäßig und fünf andere Programme weniger häufig. Die restlichen Utilities habe ich bisher selten bzw. gar nicht benutzt. Vom Nutzen der Utilities her scheint der Preis von ca. DM 120,- überhöht, jedoch erhält man zusätzlich noch ein 25seitiges Büchlein mit verschiedenen Tips und ein Poster, auf dem wichtige Speicherstellen bzw. Monitorroutinen aufgelistet sind. Außerdem können die Programme – wie bereits erwähnt – alle gelistet werden; man lernt dadurch neue Programmiertechniken und erhält neue Ideen.

2. DOS Boss

Hierbei handelt es sich um einen Editor, mit dem es möglich wird, DOS-Kommandos und -Fehlermeldungen zu verändern sowie das Layout des Catalogs einer Diskette völlig neuartig und ansprechender zu gestalten. DOS Boss funktioniert nur bei unverändertem DOS 3.3, nicht bei Diversi-DOS usw. Das Programm, das im zugehörigen Handbuch vollständig gelistet wird, ist ein fast lupenreines Applesoft-Programm. Im Handbuch wird nicht nur die Benutzung sehr klar und ausführlich (besonders für den Anfänger) beschrieben, sondern auch Aspekte des Disketten-Betriebssystems so einfach erklärt, daß es auch für Anfänger verständlich ist.

Was ist mit DOS Boss alles möglich?

1) **Änderungen der DOS-Kommandos** – Alle DOS-Befehle werden angezeigt und können dann nach eigenen Vorstellungen umbenannt oder eingedeutscht werden. Für verschiedene Befehle werden als Beispiel andere oder bessere Kommando-Versionen vorgestellt. Außerdem werden noch Regeln aufgezählt, die beim Ändern von Befehlen unbedingt beachtet werden sollten.

2) **Änderung der Fehlermeldungen** – Man erhält alle Fehlermeldungen angezeigt und kann dann nach eigenem Gutdünken Veränderungen vornehmen. Hierfür werden im Handbuch auch Beispiele gezeigt.

3) **Änderungen des Catalog-Formats** – Normalerweise können nur maximal 23 File-Namen auf dem Bildschirm angezeigt werden. Mit DOS Boss kann diese Anzahl bis auf 88 erhöht werden.

4) **Änderungen in der Volume-Zeile** – DOS Boss erlaubt auch Änderungen in der Volume-Zeile. Die Zeile kann in den drei Ausgabemodi (NORMAL, FLASH, INVERSE) angezeigt werden; der Text kann beliebig geändert werden (maximal 16 Zeichen sind möglich). Die Volume-Nummer wird wahlweise angezeigt oder weggelassen. Mit „Spezialzeichen" können Zeichen eingerückt oder überschrieben sowie Zeilenvorschub oder ein Return eingegeben werden.

5) **Änderungen des File-Codes** – Der normale File-Code (A, I, B, T) kann genauso wie das UNLOCK-Symbol (Leertaste) und das LOCK-Symbol (*) geändert werden; falls erforderlich sogar in inverse oder blinkende Zeichen.

Sind alle gewünschten Änderungen vorgenommen, so können diese auf zwei Arten auf Diskette „verewigt" werden:

1. Initialisierung einer frischen Diskette.

2. Speicherung von entsprechend vorbereiteten Programmzeilen in einem Applesoft-Programm mit Hilfe eines EXEC-Files.

DOS Boss ist jedoch nicht das einzige Programm, das auf der Diskette vorhanden ist:

Key Cat ist eine Utility, die eine leichtere Auswahl von Programmen aus der Catalog-Routine heraus gestattet. Jedes Programm wird mit einem einfachen Tastendruck geladen und gestartet. Auf Wunsch werden die freien und belegten Sektoren der benutzten Diskette angezeigt.

Bait Cat zeigt die Files einer Diskette sortiert nach Filetypen an.

Zwei Programme, **LP** und **NU**, werden mit entsprechenden Befehlsänderungen über DOS Boss zu den „Pseudobefehlen" Help und Menu.

Neben der Programmbedienung erklärt das Handbuch auch, wie man DOS ohne DOS Boss ändern kann. Verschiedene Tips und Tricks werden in einem besonderen Büchlein und in einem 10seitigen Teil des Handbuchs mitgeliefert.

Der Preis von DM 100,- ist gerechtfertigt. Das Programm ist nützlich für diejenigen, die ihre Programme verkaufen oder verleihen. Aber auch derjenige, der keine direkte Anwendung für DOS Boss hat (so wie ich), lernt viel über DOS 3.3.

3. Tip Disk 1

Jedes von den Beagle Brothers verkaufte Objekt umfaßt neben diversen Zutaten wie PEEK- und POKE-Poster, Befehlstabelle, Farbtabelle, ASCII-Tabelle usw. noch ein Tip-Buch oder eine Tip-Chart (Poster). Neben Hardware-Tips enthalten diese Bücher und Poster noch verschiedene Programme und Programmiertricks.
Die Tip Disk 1 enthält alle Programme, die in den Tip Büchern 1 bis 4 erschienen sind.

Auf der Diskette befinden sich zwei Utilities zur Konvertierung von Programmen in die jeweils andere Sprache (Applesoft bzw. Integer-BASIC), eine FLASH-Befehl-Simulierung für Integer-BASIC, ein Begrüßungsprogramm und ein Zählprogramm für die Anzahl von Programmläufen. Des weiteren gibt es Programme, die den Textbildschirm auf den Drucker ausgeben, eine vollständige ASCII-Tabelle ausdrucken und das Layout des Textbildschirms zeigen. Ein kleines Spielprogramm ist ebenso auf der Diskette zu finden wie eine Alphabetisierungsroutine. In weiteren Programmen wird die Anwendung von Shapes gezeigt, die Slot-Belegung dargestellt und erklärt sowie das Applesoft-ROM durchsucht.

Einige Programme sind nützlich, viele sind unnütz, jedoch sind alle Programme interessant. Sie können beliebig kopiert und gelistet werden und erfüllen damit einen gewissen Lehrzweck. Der Preis von DM 70,- ist meines Erachtens gerechtfertigt.

4. Silicon Salad

Auf dieser Diskette sind wie auf der Tip Disk 1 jede Menge Programme. Unter anderem alle Programme der Tip-Bücher 5 bis 7 und der Tip-Chart 1.

Die Beagle Brothers führen eine Art Wettbewerb durch, in dem die Programmkäufer bzw. die Tip-Buch-Leser aufgefordert werden, zweizeilige, voll lauffähige Programme einzusenden. Da Programme einen Zweck erfüllen sollen und in zwei Zeilen weder ein umfangreiches Frage- und Antwortspiel noch ein Tabellenkalkulationsprogramm programmierbar ist, bleiben nur Grafik- und Musikprogramme übrig. Der größte Teil der Zweizeiler zeigt daher auch Grafiken auf den Lores- bzw. Hires-Seiten. Diese Programme sind sehr unterhaltsam (ich könnte manchen stundenlang zuschauen!) und zeigen außerdem, was effektives Programmieren ist und wie man kompakten Programmcode „produziert".
Unter den Programmen aus den Tip-Büchern befindet sich ein Spielprogramm (Black Jack), eine Sortier- und eine Text-Zentrierroutine.
Des weiteren gibt es ein Programm, das es gestattet, Hires-Bilder mit BRUN zu starten. Dabei wird automatisch die richtige Hires-Seite gewählt und eingeschaltet.
Ein Programm, genannt **Text Imprint**, nimmt allen Text vom Textbildschirm und „zeichnet" ihn auf den Hires-Grafikbildschirm.
Ein **Text Screen Formatter** erlaubt es, zuerst ein Bildschirm-Layout zu entwerfen und „formt" dieses Layout später zu einem reinen Applesoft-Programm (VTAB-, HTAB- und PRINT-Kommandos) um.
Help Screen speichert eine kurze Bedienungsanleitung oder einen Hilfstext auf die Textseite 2 und zeigt diese nach Eingabe von Ctrl-I.
Ein weiteres Programm verschiebt einen beliebigen Bereich der Textseite 1 in die Seite 2, ein anderes gibt einen beliebigen Bereich des Textschirms auf dem Drucker aus.

Zusätzlich befinden sich noch sechs Utilities auf der Diskette:
1) **Disk Scanner** sucht nach physikalisch defekten Sektoren einer Diskette. Werden „schlechte" Sektoren gefunden, wird die VTOC (Inhaltstabelle) der Diskette so geändert, daß die schlechten Sektoren nicht mehr benutzt werden können.
Als Fazit kann ich sagen, daß ich den Kauf der hier beschriebenen Programmpakete nicht bereue. Wenn ich auch verschiedene Programme aus unterschiedlichen Gründen nicht benutze, so hat mir jedoch jedes Paket sehr beim Kennenlernen des Apple und beim Programmieren geholfen.

2) **DOS Killer** entfernt DOS von einer Diskette. Damit erhält man weitere 32 frei benutzbare Sektoren.
3) **Key Clicker** verlangt die Eingabe einer Zahl, die als Wert für einen „Klick" gespeichert wird. Danach sendet jeder Tastendruck einen Ton (Klick) aus.
4) **Program Splitter** teilt ein großes Applesoft-Programm in getrennte Speicherbereiche auf. Benutzt nämlich ein sehr langes Applesoft-Programme die Hires-Grafik, so kann es selbst oder dessen Variablen mit den Hires-Bildern derart in Konflikt geraten, daß sie sich gegenseitig löschen. Program Splitter produziert ein „Loch" im Applesoft-Programm, in dem dann die Hires-Seite liegt.
5) **Two Track Cat** stellt DOS so um, daß für den Catalog zwei Spuren statt einer Spur zur Verfügung stehen. Damit kann statt 105 Files die doppelte Anzahl gespeichert werden. Die Files müssen natürlich entsprechend kurz sein.
6) **Undelete** „entlöscht" gelöschte Files einer Diskette.
Diese Utilities alleine rechtfertigen den Preis von ca. DM 90,- DM für Silicon Salad.

Triple-Dump

Ein Bildschirm-Druckprogramm

Erfahrungsbericht von Rolf W. Becker

Das von der Firma Beagle Bros. entwickelte Programm Triple-Dump ist jetzt bei einschlägigen deutschen Importeuren für etwa DM 160,- erhältlich. Es dient dem Zweck, 40- und 80-Z/Z-Text-, Lores-, Double-Lores-, Hires- und Double-Hires-Bildschirminhalte auf den Drucker auszugeben. Für Double-Hires werden die mit Beagle-Graphics (s. Peeker, 8/85, S.77) gezeichneten Bilder benötigt.
Zum Lieferungsumfang gehören je eine beidseitig zu benutzende DOS-3.3- und ProDOS-Diskette. Auf der ersten Seite befindet sich das Triple-Dump-Programm, Textdateien mit den häufigsten Druckern und Interfaces (je ca. 50). Damit kann man Triple-Dump auf seine Konfiguration einstellen, abspeichern und hat bei jedem Programmstart den richtigen Drucker mit dem entsprechenden Interface eingestellt. Mit dem Programm Transfer kann diese Einstellung dann auf die DOS-lose Rückseite übertragen werden, wie im übrigen auch auf andere Disketten, denn

alle Programme sind wie gewohnt listfähig und kopierbar und können somit in eigene Programme eingebunden werden.
Auf der Rückseite (ohne DOS) sind mehrere Hilfsprogramme, um alle 6 Arten von Bildschirminhalten abzuspeichern oder zu drucken, z.B. ein Programm, das Text in die Hires oder Double-Hires schreibt, wozu sich die Fonts von Beagle-Graphics eignen. Außerdem sind Beispiel-Bilder auf beiden Diskettenseiten vorhanden.

Implementierte Befehle

Nach dem Starten von Triple-Dump erscheint ein Optionsmenü mit der abgespeicherten Drucker-Interface-Konfiguration. Danach kann man u.a. wählen:

Type: Art des Bildschirminhalts, der mit Return ausgewählt wird.

Load Picture: Nach Return wird der Catalog gezeigt, mit den Pfeiltasten die Bilddatei ausgewählt und mit Return geladen.

Crop Picture: Das Bild kann von allen vier Seiten verkleinert werden, damit nur ein Ausschnitt ausgedruckt wird. Der Ausschnitt kann mit „Magnify" der Papierbreite entsprechend vergrößert gedruckt werden.

Print Picture: Das Bild wird ausgedruckt. Falls der Vergrößerungsmaßstab oder die linke Randeinstellung die Papierbreite überschreiten sollte, wird nicht gedruckt, sondern darauf hingewiesen, daß neu eingestellt werden muß.

Rotate: Der Ausdruck kann um 0, 90, 180, 270 Grad gedreht werden.
Indent: Der Abstand vom linken Rand kann der Papierbreite entsprechend beliebig festgelegt werden.
Negative: Damit kann der Bildschirm invertiert bzw. negativ gedruckt werden (funktioniert nicht bei Textbildschirm).
Magnify X, Y: Die Vergrößerung des Bildes wird festgelegt.
Density: Die druckerabhängige Druckart wird festgelegt (Punkte je Zoll).
Disk Drive: 1 oder 2.

Um die Einstellungen für Drucker, Interface, Slot, Papierbreite, Verzögerung, Zeilenvorschub usw. braucht man sich normalerweise nicht zu kümmern. Nach Betätigung der Leertaste kann jederzeit das im Speicher befindliche Bild angesehen werden. Mit F wird ein Blattvorschub erzeugt, mit L ein Zeilenvorschub. Mit ESC wird das Programm verlassen.

Triple-Dump ist ein komfortables und gut funktionierendes Druckprogramm, das viele Druckmöglichkeiten bietet. Es ist damit auch ganz einfach, Bilder zwischen Text zu drucken oder umgekehrt. Bei allen Varianten wie Rotate, Magnify, Crop, Indent usw. werden die Ausdrucke gleich gut und sind einwandfrei.

Als Zugabe ist auf der Rückseite noch „Big Banner" enthalten, der bis zu 200 Zeichen lange Texte in einem Schriftgrad von 8 Zoll ausdruckt. Da das Programm auflistbar ist, habe ich am Anfang noch eine kleine Abfrage eingefügt, die die Schriftgröße festlegt. Genauso wie dieses Programm können auch alle anderen eingedeutscht werden. Die englische Beschreibung ist übersichtlich und gut verständlich.

Diversi-DOS 4.1-C mit Garbage-Collection

von Rudolf Rötering

Seit kurzem ist der von mir im Peeker 6/85, S. 72ff. beschriebene RAM-Disk-Driver und eine schnelle Garbage-Collection im normalen Lieferumfang der Version 4.1-C zum Preis von $ 30.00 enthalten. Ab 1. Juli 1985 hat DSR Inc. folgende neue Anschrift:

Diversified Software Research, Inc.
34880 Bunker Hill
Farmington, MI 48018-2728
Tel. 313 553-9460
USA

Die Ziffernfolge 815 877-1343, die im Nachwort hinter meinem Bericht im Peeker 6/85 folgt, ist die Telefonnummer von DSR Inc. in Rockford, Ill. und nicht die Kontonummer der (VISA/Master-Card). Es geht wirklich am einfachsten und schnellsten, wenn, wie in meiner Vorlage erwähnt, die 30,- Dollar in bar per Post im Bestellbrief übersandt werden. Das mag für deutsche Verhältnisse ungewöhnlich sein, ist aber in den USA durchaus üblich.

Garbage-Collection von DSR

Die neue schnelle Garbage-Collection wird mit BRUN GARB aktiviert. GARB setzt die Ampersand- und Ctrl-Y-Vektoren und ersetzt die jeweilige INIT-Routine (64K-DOS oder 48K-DOS) durch die neue Collection-Routine. Sie kann daher sowohl durch den Ampersand- (mit „&") als auch durch den Ctrl-Y-Vektor mit „CALL 1016" ausgelöst werden. Durch

POKE 48816,N (48K-DOS) oder
POKE 65200,N (64K-DOS)
kann mit N wie bei Harald Grumsers LC.FRE oder RAM.FRE (Peeker 1/85) die Unterschreitung der Anzahl der freien String-Seiten als Bedingung für das Durchführen dieser Routine gewählt werden. Als Vorgabe gilt N = 4, das 1K Restspeicher bedeutet. Mit N = 0 wird die Garbage-Collection erzwungen.
Weil diese Routine den Speicherbereich der INIT-Routine belegt, steht entweder der INIT-Befehl oder die Garbage-Collection zur Verfügung. Hieraus ergibt sich der Vorteil, daß z.B. die Bank 2 der Language-Card für die INPUT-2.0-Routinen (vom Hüthig Software Service) zur Verfügung steht. Beim 64K-DOS kann durch BRUN UNGARB64 der Garbage-Collection-Patch wieder rückgängig gemacht werden.
Mit dem Programm FRE.TEST (aus Peeker 1-2/85) wurde ein Zeitvergleich mit anderen Collection-Routinen gemacht, den die **Tabelle 1** darstellt, wobei die Werte für die bereits getesteten Routinen übernommen wurden, zumal mir nur eine mechanische Stoppuhr zur Verfügung stand. Es wurde außer dem im Peeker 1-2/85 erschienenen Programm LC.FRE, der GARB-Routine von Diversi-DOS 4.1-C auch die FAST-GARBAGE-COLLECTION von Randy Wiggington aus „CALL A.P.P.L.E. in Depth 1" verglichen. Sie belegt beim Test den Bereich von $B900 bis $BEFA, wobei der Bereich ab

$BAA4 = 1110 Bytes als Workspace benutzt wird, der die Verarbeitungsgeschwindigkeit maßgeblich beeinflußt. Bei dieser Routine läßt sich dieser Bereich im Source-Code einstellen.
Zusammenfassend läßt sich sagen, daß Harald Grumsers Routine nicht zu unterbieten ist und die Bezeichnung „Müllabfuhr wie ein Blitz" äußerst treffend erscheint. Die FAST-GARBAGE-COLLECTION von Randy Wiggington und die GARB-Routine von Diversi-DOS 4.1-C dauern 10- bis 50mal länger und sind somit nur Mittelmaß, obwohl sie ca. 30mal schneller sind als das Interpreter-FRE. Für die FAST-GARBAGE-COLLECTION hält Apple Computer Inc. das Copyright, die im Oktober 1980 wohl als erste Routine dieser Art erschienen ist.

APPEND im Geschwindigkeitsvergleich

Beim Geschwindigkeitsvergleich der Disketten-Betriebssysteme fehlen die Werte für APPEND, bei dem der Unterschied zu DOS 3.3 besonders groß ist. ProDOS ist hier wohl genauso schlau wie Diversi-DOS und nimmt die Abkürzung über die Track/Sektor-Liste. Ich habe mir erlaubt, das Programm um diesen Test zu erweitern und die Testdaten in die Tabelle einzufügen. Die Zeitangaben im Peeker 6/85 beziehen sich beim Test auf leere Disketten. Versuche auf zum Teil beschriebenen Disketten können zum Beispiel beim BSAVE unter ProDOS 30 bis 40 Sekunden länger dauern. Beim WRITE- und READ-Test in Zeile 50 bzw. 60 komme ich bei ProDOS auf völlig andere Zeiten. Ich messe beim WRITE 70 statt 108 Sekunden und beim READ 145 statt 105 Sekunden. Beim DOS-3.3-READ in Zeile 60 muß es 283 statt 243 Sekunden heißen.
Die **Tabelle 2** gibt die von mir auf einem Disk-II-Laufwerk gemessenen Werte wieder. Sofern meine Zeiten nicht mehr als 5 Sekunden vom Bericht abwichen, habe ich die Daten aus der Ausgabe 6/85 übernommen.

Tabelle 1: Zeitvergleich der Garbage-Collection-Routinen

	Anzahl der erzeugten String-Leichen				
	100	500	1000	2000	5000
LC.FRE Peeker 1-2/85	0,03	0,11	0,20	0,52	1,3 s
FAST-GARBAGE-COLLECTION	0,30	2,00	4,50	12,00	52,4 s
GARB Diversi-DOS 4.1-C	0,30	1,70	4,30	12,90	67,0 s
Applesoft-Interpreter-FRE	0,94	19,40	74,50	292,00	1800,0 s

Alle Zeitangaben in Sekunden.

Listing des modifizierten FRE.TEST aus Peeker 1-2/85

```
110 HIMEM: 47360
120 A = 5000
130 B$ = CHR$ (7): REM Bell
140 GOSUB 220: DIM A$(A)
150 FOR I = 0 TO A
160 A$(I) = B$:A$(I) = B$
170 NEXT
180 GOSUB 220: PRINT B$;
190 CALL 47370: REM bei FASTGARBAGE aus CALL A.P.P.L.E.
195 REM Diversi-DOS GARB: POKE 65200,0: CALL 1016
200 PRINT B$;: GOSUB 220
210 END
220 PRINT "STRINGS= "; PEEK (115) + PEEK (116) * 256 -
    PEEK (111) - PEEK (112) * 256,
230 PRINT "FRE= "; PEEK (111) + PEEK (112) * 256 - PEEK
    (109) - PEEK (110) * 256
240 RETURN
```

Tabelle 2: DOS-Geschwindigkeitsvergleich

Zeile	Befehl	ProDOS	Diversi-DOS 4.1-C	DOS 3.3
10	BSAVE	383	155	837
20	BLOAD	97	125	628
30	WRITE	30	57	140
40	READ	27	47	165
80	APPEND	(3)	(5)	(127)
50	WRITE	(72)	95	171
60	READ	(145)	86	(283)
70	APPEND	(3)	(5)	(127)

Alle Zeitangaben in Sekunden. Die Klammerwerte sind gegenüber Peeker 6/85 korrigiert bzw. ergänzt.

Modifiziertes und erweitertes Listing aus Peeker 6/85 zum DOS-Geschwindigkeitstest.

```
10 FOR I = 1 TO 20: PRINT CHR$ (4)"BSAVE XXX,A$1000,
   L$7FFF": NEXT : PRINT CHR$ (7): END

20 FOR I = 1 TO 20: PRINT CHR$ (4)"BLOAD XXX": NEXT :
   PRINT CHR$ (7): END

30 A$ = "123456789 123456789 123456789 123456789 123456789
   123456789 123456789 123456789 123456789 123456789"
31 PRINT CHR$ (4)"OPEN YYY": PRINT CHR$ (4)"WRITE YYY":
   FOR I = 1 TO 500:
32 PRINT A$
33 NEXT : PRINT CHR$ (4)"CLOSE": PRINT CHR$ (7): END

40 PRINT CHR$ (4)"OPEN YYY": PRINT CHR$ (4)"READ YYY":
   FOR I = 1 TO 500
41 INPUT "";A$
42 NEXT : PRINT CHR$ (4)"CLOSE": PRINT CHR$ (7): END

50 A = 123456789
51 PRINT CHR$ (4)"OPEN ZZZ": PRINT CHR$ (4)"WRITE ZZZ":
   FOR I = 1 TO 5000
52 PRINT A
53 NEXT : PRINT CHR$ (4)"CLOSE": PRINT CHR$ (7): END

60 PRINT CHR$ (4)"OPEN ZZZ": PRINT CHR$ (4)"READ ZZZ":
   FOR I = 1 TO 5000
61 INPUT "";A
62 NEXT : PRINT CHR$ (4)"CLOSE": PRINT CHR$ (7): END
```

APPEND-Erweiterung

```
70 PRINT CHR$ (4)"APPEND ZZZ": PRINT CHR$ (4)"WRITE ZZZ":
   REM bei ProDOS ohne WRITE!
71 PRINT "123456789"
72 PRINT CHR$ (4)"CLOSE": PRINT CHR$ (7): END

80 A$ = "123456789 123456789 123456789 123456789 123456789
   123456789 123456789 123456789 123456789 123456789"
81 PRINT CHR$ (4)"APPEND YYY": PRINT CHR$ (4)"WRITE YYY":
   REM bei ProDOS ohne WRITE!
82 PRINT A$
83 PRINT CHR$ (4)"CLOSE": PRINT CHR$ (7): END
```

BASF-Flexydisk 1X und 1D

Erfahrungsbericht von Ulrich Stiehl

Auch wenn uns ein Leserbriefschreiber lobend mit der Stiftung Warentest vergleicht, so muß doch darauf hingewiesen werden, daß eine Computerzeitschrift im allgemeinen und der Peeker im besonderen normalerweise ein Produkt nicht in demselben Umfang testen kann, wie dies bei einem staatlich geförderten Testinstitut möglich ist. Eine Zeitschriftenredaktion kann sich um Objektivität bemühen, doch wäre es aus Kostengründen nicht vertretbar, ein Produkt wochen- oder gar monatelangen Dauertests zu unterziehen. Greifen wir als ein Beispiel Diskettenlaufwerke heraus. Wenn man einen Tag lang ein Laufwerk in Verbindung mit verschiedenen Betriebssystemen ausprobiert, Übertragungsraten mißt und allgemeine

Eindrücke sammelt, so sind dies nützliche Hinweise für den Leser, die jedoch andererseits über die Robustheit, den geringen Verschleiß und die Wartungsfreundlichkeit der Laufwerke wenig aussagen. Diese Eigenschaften könnten sich nur im monatelangen Einsatz verifizieren lassen.

Diskettentest

Wie steht es nun mit dem Testen von Disketten einer bestimmten Marke? Würde man ein einzelnes Muster eines Diskettenherstellers erhalten, so könnte man wahrscheinlich überhaupt keine verläßlichen Aussagen machen, denn dieses Muster wäre sicherlich vom Hersteller vor dem Versand geprüft worden. Wäre indessen diese eine Diskette zufällig defekt, so könnte man natürlich nicht generalisieren, daß alle Disketten dieser Marke mangelhaft sind.

Als Forschungsinstitut könnte man z.B. 10.000 Disketten kaufen und einen auf Diskettenmaterialprüfung spezialisierten Ingenieur beauftragen, alle Disketten mit wissenschaftlicher Akribie zu testen und die Ergebnisse statistisch aufzubereiten. Dies würde weit über 50.000,- DM kosten. Selbst die Stiftung Warentest würde hier die Waffen strecken.

Durch unseren Hüthig Software Service bezogen wir bislang BASF-„Flexydisk"-Disketten in größeren Mengen, die natürlich alle formatiert und bespielt werden mußten, was zwar keinen wissenschaftlichen, aber immerhin einen praxisorientierten Test darstellte. Wie die Vergangenheitsform anzeigt, sind wir inzwischen von den BASF-Disketten abgekommen, denn der Ausschuß bei 1X- und 1D-Disketten war viel zu hoch.

Schuhsohleneffekt: In den ersten Monaten dieses Jahres hatten wir bei 1X-Disketten einen Ausschuß von wahrscheinlich gut 1%, d.h. 10 von 1000 Disketten wiesen von vornherein den – wie ich es später nannte – „Schuhsohleneffekt" auf. Gleichviel ob man Schuhsohlen oder solche fehlerhaften 1X-Disketten in das Laufwerk steckte: Eine Formatierung war in beiden Fällen grundsätzlich unmöglich.

Klebfilzeffekt: Daneben trat häufig der „Klebfilzeffekt" auf. Dabei handelte es sich um fabrikneue Disketten, die sich nicht richtig drehen lassen, weil der Reibungswiderstand der Filzschicht zu groß ist. Solche Disketten waren zwar in der Regel formatierbar, mußten jedoch von vornherein aussortiert werden. Darüber hinaus trat der „Klebfilzeffekt" sehr häufig *nach* dem Versand auf. Dazu muß man wissen, daß sich eine Peeker-Diskette in der Lasche eines 3mal gerillten roten 250g/qm-Broschurenumschlags befindet, der in eine Plastikhülle gesteckt wird, die jedoch nicht zugeschweißt wird. Das ganze wird dann in einem Wellpappe-Kuvert mit zusätzlicher Steifpappe-Einlage versandt. Bei einem Preis von nur DM 20,- für Fortsetzungsbezieher verbietet sich eine noch aufwendigere Verpackungsform (Hartplastik-Container o.ä.), denn sonst würde die Verpackung mehr als die Diskette kosten. Außerdem sind die im Kartonagenhandel erhältlichen Spezialdiskettenkuverts erheblich weniger aufwendig. Offenbar sind jedoch BASF-Disketten sehr empfindlich gegen mechanischen Druck und Temperaturschwankungen, was beim Postversand den Klebfilzeffekt hervorrufen kann.

Doch zurück zu dem „Schuhsohleneffekt". Nachdem ich mich verstärkt um das Diskettenproblem kümmerte – das eigentliche Kopieren erfolgt nach wie vor außer Haus (mit normalen Apple-Originallaufwerken) –, wurden in Form von Strichlisten immer mehr „Schuhsohlen" registriert. Diese wurden dann über meinen Heidelberger Apple-Händler, über den wir damals die Disketten bezogen, bei BASF in Ludwigshafen zur Materialprüfung eingereicht. Als Antwort bekamen wir nur die lapidare mündliche Mitteilung, daß alle Disketten in Ordnung seien, obwohl sie auf keinem unserer benutzten Originallaufwerke (Disk-II-Drives, Duodisk und IIc-Drives) formatierbar waren. Uns wurde dann empfohlen, statt der 1X-Disketten (einseitig, einfache Dichte) 1D-Disketten zu erwerben (einseitig, doppelte Dichte). Als dann bei den ersten 400 von 2000 1D-Disketten 12 Disketten den „Schuhsohleneffekt" aufwiesen – dies sind immerhin schon 3% – waren wir gezwungen, von BASF-„Quality"-Disketten Abstand zu nehmen. Aufgrund unserer praktischen Erfahrung mit weit über 10.000 1X- und 1D-Disketten muß man leider konstatieren, daß hier mit einem Ausschuß von über 1% gerechnet werden muß. Möglicherweise sind 1D-Disketten anderer Produzenten auch nicht besser, doch angesichts der relativ hohen Preise der BASF-Marken-Disketten ist man wenig gewillt, beim Formatieren und Kopieren größerer Mengen stets einen Abfalleimer für den Ausschuß bereitzuhalten.

Die Microsoft-Premium-Softcard für den Apple IIe

getestet von Ulrich Stiehl

Wer das Betriebssystem CP/M einsetzen will, kann über die Firma Microsoft zwei verschiedene Arten von Z80-Karten erwerben:
1. die „normale" Softcard, die es bereits seit einigen Jahren gibt und die eine reine Z80-Prozessorkarte ohne Zusatz-RAM darstellt; einsetzbar auf dem Apple II Plus *und* dem Apple IIe,
2. die „prämiierte" Softcard, die es seit Ende 1983 gibt und über zusätzliche Eigenschaften verfügt; *nur* einsetzbar auf dem Apple IIe.

1. Hardware

Die Premium-Softcard vereinigt die Funktionen von drei Karten:
(a) Z80-Prozessorkarte mit 6 MHz
(b) 80-Z/Z-Karte
(c) 64K-RAM-Karte
Die Funktionen (b) und (c) werden normalerweise von der erweiterten 80-Zeichenkarte (= Extended 80 Column Card) wahrgenommen. Da sowohl die Premium-Card wie auch die erweiterte 80-Zeichenkarte nur für den auf der Mitte der Platine befindlichen Slot 3 gedacht sind, kann man nur entweder die eine oder andere Karte benutzen, d.h. niemals beide gleichzeitig. Wann ist die Anschaffung der Premium-Softcard optimal? Genau dann, wenn man unter CP/M arbeiten will und noch keine erweiterte 80-Zeichenkarte für den Apple IIe besitzt. Auch dann, wenn man den Apple IIe mit erweiterter 80-Zeichenkarte erworben hat, ist die Premium-Softcard durchaus in Erwägung zu ziehen, weil sie nur einen Slot belegt, während erweiterte 80-Zeichenkarte plus „normale" Softcard zwei Slots in Beschlag nehmen.

Die Premium-Softcard ist ungewöhnlich groß. Zur Installierung ist es deshalb erforderlich, daß man zunächst die in den benachbarten Slots 2 und 4 befindlichen Karten herauszieht, dann die Premium-Softcard in den Slot 3 steckt und schließlich wieder die benachbarten Karten einsetzt. Diese Prozedur kann lästig werden, wenn man häufig Interface-Karten wechseln muß.

2. Software

Die Premium-Softcard wird mit einer CP/M-80-Masterdiskette geliefert, die die 64K-Version von CP/M **2.26B** enthält. Dieses CP/M 2.26B ist nur für die Premium-Softcard bestimmt und umgekehrt:
a) Würde man die Masterdiskette auf einem Apple IIe booten, der nur die „normale" Softcard enthält, so würde die Version 2.26B nicht die Existenz der Softcard entdekken („Can't find softcard").
b) Würde man umgekehrt auf einem Apple IIe mit Premium-Softcard das „normale" CP/M 2.2 booten, so würde sich das Betriebssystem „aufhängen".
Daraus folgt, daß man das neue CP/M auf alle älteren Programmdisketten kopieren muß. Dabei verfahre man wie folgt:
1. CP/M 2.26B booten
2. „COPY" starten
3. Mit „B:=A:/S"
CP/M von Drive A auf Drive B kopieren.

Neben dem Betriebssystem im engeren Sinne enthält die Masterdiskette folgende Systemprogramme und Utilities:

APDOS: Überträgt Text- und Binärfiles von einer DOS-3.3-Diskette im Drive 2 („B:") auf eine CP/M-Diskette in Drive 1 („A:").

ASM: Assembliert 8080-Quellcode (also keinen Z80-Quellcode i.e.S.).

AUTORUN: Dient dem automatischen Start eines Programms nach dem Booten (ähnlich wie START-UP unter Pascal und ProDOS sowie „HELLO" unter DOS 3.3).

BOOT: Schaltet den Z80-Prozessor ab und bewirkt den (warmen) Bootvorgang eines 6502-Betriebssystems, z.B. von DOS 3.3. Anstelle von BOOT kann man auch die drei Tasten „Ctrl-Offener-Apfel-Reset" drücken. (Ctrl-Reset *allein* genügt hier nicht und würde nur – wie Ctrl-C – einen CP/M-Warmstart herbeiführen.)

CAT: Zeigt im Gegensatz zu „DIR" einen etwas ausführlicheren „Catalog" einer Diskette an, wobei z.B. auch die Längen der Dateien in K angegeben werden.

CONFIGIO: GBASIC-Programm zur Gerätekonfigurierung; wird bei der Premium-Softcard normalerweise nicht benötigt.

COPY: Dient zum Formatieren und Kopieren ganzer Disketten oder des Betriebssystems allein (s.o.).

DDT: Debugger für 8080-Programme (Dynamic Debugging Tool); wird im Microsoft-Handbuch nicht erklärt.

DUMP: Bewirkt einen Dump einer Datei in hexadezimaler Form (vgl. TYPE).

ED: „Primitiv"-Editor für Textfiles; wird im Microsoft-Handbuch nicht erklärt.

GBASIC: Kombination von altem MBASIC und GBASIC (G= HGR-Grafik) speziell für die Premium-Softcard. Das neue GBASIC meldet sich mit „BASIC-80 **Rev. 5.27**" und „33.063 Bytes free". Man verfügt also jetzt über etwas mehr freien Speicher. Ansonsten unterscheidet sich die 5.27-Version nur geringfügig von vorangehenden Versionen. Beispielsweise wurden die Befehle TRON und TROFF in TRACE und NOTRACE umbenannt (wie bei Applesoft). DELETE kann jetzt neben DEL zum Löschen von Programmzeilen benutzt werden. Dagegen wurden die Kassettenbefehle CLOAD und CSAVE gestrichen.

LOAD: Konvertiert eine „HEX"-Datei in eine „COM"-Datei.

MFT: Dateikopierprogramm für 1-Drive-Besitzer (vgl. PIP).

PATCH: Update-Utility für zukünftige CP/M-Versionen.

PIP: Dateikopierprogramm für 2-Drive-Besitzer (vgl. MFT).

STAT: Zeigt den Status von Dateien an usw.

SUBMIT und XSUB: Erzeugt Befehlsdateien (ähnlich wie EXEC in ProDOS, DOS 3.3 und Pascal); wird im Microsoft-Handbuch nicht erklärt.

3. Handbücher

Es werden zwei umfangreiche Loseblattwerke mit jeweils ca. 400 Seiten Umfang mitgeliefert. Das eine Loseblattwerk („Package") ist mehr für Anwender und GBASIC-Programmierer, das anderer mehr für Systemprogrammierer gedacht. Ausführliche Stichwortregister erleichtern das Nachschlagen. Einige der o.g. Utilities werden in den Loseblattwerken nicht erklärt. Deshalb wird zusätzlich in den USA das „CP/M User Guide" von Hogan und in Deutschland das „CP/M Handbuch" von Zaks mitgeliefert.

4. Allgemeine Eindrücke

– Die Premium-Softcard ist dann ihr Geld wert (je nach Händler DM 1000,- bis DM 1500,-), wenn man die erweiterte 80-Zeichenkarte noch nicht besaß oder für sie einen Abnehmer findet, weil beide Karten nicht gleichzeitig benutzt werden können.
– Die Bildschirmsteuerung der 80-Zeichenkarte als Bestandteil der Premium-Softcard ist hervorragend gelöst. Dies mag das nachfolgende GBASIC-Demo verdeutlichen:

```
10 GET X$
20 FOR X = 1000 TO 5000:
30 PRINT X + X; " - ";: NEXT
40 PRINT CHR$ (7): REM 35 Sek.
50 GET X$
60 FOR X = 1000 TO 5000:
70 PRINT " AA-AA ";: NEXT
80 PRINT CHR$ (7): REM 19 Sek.
```

Unter GBASIC (mit 80-Zeichenkarte) wurden 53 resp. 19 Sekunden gemessen. Zum Vergleich wurde das gleiche Programm in Applesoft mit 80-Zeichenkarte *und* Speedemon-Karte programmiert. Die entsprechenden Zeiten waren hier 53 resp. 40 Sekunden. Daraus folgt, daß die 80-Z/Z-Bildschirmsteuerung bei der Premium-Softcard von Microsoft etwa doppelt so schnell wie bei der erweiterten 80-Zeichenkarte von Apple ist, selbst wenn man eine Prozessorkarte wie die Speedemon einsetzt. Lediglich die Accelerator-Karte kann mit der Premium-Softcard mithalten.
– Unter 6502-Betriebssystemen (DOS 3.3, Pascal usw.) kann die Premium-Softcard wie eine normale, erweiterte 64K-Karte eingesetzt werden, z.B. als RAM-Disk usw.

Peeker-Sammeldisketten #1–#11

Benutzungshinweise

Kopieren: Die Sammeldisketten sind normale 5,25-Disketten (140K), die in der Regel im DOS-3.3-Format angelegt sind und mit COPYA oder einem anderen Diskettenkopierprogramm *insgesamt* dupliziert werden können. Ferner kann man mit FID oder einem anderen Dateikopierprogramm *einzelne* Dateien auf eigene Disketten überspielen. Beachten Sie, daß die Disketten kein Betriebssystem enthalten und deshalb nicht bootfähig sind. Also erst DOS-3.3-Systemdiskette mit PR#6 booten, dann mit COPYA (s.a. Disk #2) oder FID (s.a. Disk #10) die Diskette ganz oder dateiweise kopieren. Da die Sammeldisketten in der Regel „randvoll" sind, müssen Programme, die Datendateien anlegen (z.B. TESTGENERATOR), stets auf eigene Disketten umkopiert werden.

Konvertieren: Einige DOS-3.3-Disketten enthalten Pascal- oder CP/M-Quellprogramme, die zunächst auf UCSD-Pascal- oder CP/M-formatierte Disketten konvertiert werden müssen. CP/M-Besitzer verfügen über das Konvertierungsprogramm **APDOS**. Es funktioniert wie folgt:
– CP/M booten von Drive „A:"
– APDOS eingeben
– Sammeldisk in Drive „B:"
– CAT eingeben
– FILE=FILE eingeben
„FILE" steht für den Namen der zu konvertierenden Datei. Näheres siehe CP/M-Handbuch.
UCSD-Pascal-Dateien können mit **GETDOS** (s. Disk #9) konvertiert werden. GETDOS funktioniert wie folgt:
– Pascal booten von Drive „4"
– E(xecute) GETDOS
– Sammeldisk in Drive „5"
– „5" für Sammeldisk-Unit und
– „4" für Pascaldisk eingeben
– FILE eingeben
„FILE" steht für den Namen der zu konvertierenden Datei. Warnung: GETDOS zeigt nicht den DOS-Catalog an; man muß also den Namen bereits kennen. Ferner muß auf der (Kopie der) Sammeldisk der Name bereits auf 15 Zeichen gekürzt worden sein.

Übertragungen nach ProDOS können mit DOSTOPRO (von „ProDOS für Aufsteiger, Bd. 2", Begleitdiskette) erfolgen. Es dürfen jedoch nur diejenigen Programme übertragen werden, die ausdrücklich für ProDOS angegeben sind.

Implementieren: Nachfolgend werden zu jedem Programm bis zu 7 Informationen gegeben:
(1) **Zweck**: Charakterisierendes Kurzstichwort zum Programm.
(2) **Heft/Seitenzahl**: Um das Programm sinnvoll nutzen zu können, sollte man insbesondere bei den unselbständigen Programmen (s.u.) den entsprechenden Aufsatz und ggf. auch das abgedruckte Programm studiert haben.

(3) **Gerätekonfiguration**: Erforderliche Hardware, und zwar II+ = Apple II Plus oder Kompatibler; IIe = Apple IIe; IIc = Apple IIc; Videx = 80-Z/Z-Karte beim II+; G/K = Kleinschreibumrüstsatz beim II+ (kann oftmals durch Videx-Karte ersetzt werden; Großbuchstabenumwandlung mit VERSAL, Disk #10, möglich); LC = Language-Card beim II+; 64K-Karte = 80-Z/Z-Karte plus 64K-Zusatzspeicher beim IIe. 2-Drive-Programme, spezielle Drucker und sonstige Zusatzkarten sind gesondert vermerkt.
(4) **Betriebssystem**: DOS 3.3 (schließt auch Diversi-DOS ein), UCSD-Pascal (1.1, 1.2), ProDOS (1.0.2 usw.), CP/M (2.2 usw.).
(5) **Programmstart**: Bei selbständigen, d.h. in sich geschlossenen Programmen wird der Startbefehl, z.B. RUN DEMO, vermerkt. Bei unselbständigen Hilfsprogrammen, die in eigene Programme eingebaut werden müssen, steht nur das Wort „unselbständig", da deren Anwendung ohne das Durchlesen des entsprechenden Programmlistings im Peeker nicht möglich ist.
(6) **Sonstiges**: Besonderheiten beim Aufruf oder der Namensgebung der Programme.
(7) **Errata**: Berichtigungen und Erweiterungen mit Angabe der entsprechenden Peeker-Hefte.

Disk #1

(DOS 3.3; Heft 1 + 2/84)

T.DISASSEMBLER.65C02
DISASSEMBLER.65C02
(1) 65C02-Disassembler; (2) Heft 1/84, S.14; (3) II+ (mit LC) oder IIe; (4) DOS 3.3; (5) BRUN DISASSEMBLER.65C02

T.ACCEL.WAIT
ACCEL.WAIT
T.ACCEL.BOOT
ACCEL.BOOT
ACCEL.LC.KOPIERER
T.ACCEL.LC.KOPIE
ACCEL.LC.KOPIE
T.ACCEL.ROM.KOPIE1
ACCEL.ROM.KOPIE1
T.ACCEL.ROM.KOPIE2
ACCEL.ROM.KOPIE2
(1) Hilfsprogramme für das Pseudo-ROM der Accelerator IIe; (2) Heft 1/84, S. 19; (3) IIe; Accelerator-Karte; (4) DOS 3.3; (5) RUN ACCEL.LC.KOPIERER, RUN ACCEL.LC.KOPIE, BRUN ACCEL.ROM.KOPIE1, BRUN ACCEL.ROM.KOPIE2

TURTLE GRAFIK MIT REMS
TURTLE GRAFIK OHNE REMS
(1) Turtle-Grafik-Simulierung als Applesoft-Programm; (2) Heft 1/84, S. 26; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN TURTLE GRAFIK OHNE REMS; (6) TURTLE GRAFIK MIT REMS dient nur zur Veranschaulichung, sollte jedoch nicht gestartet werden. Weitere Übungsbeispiele in Heft 2/84, S. 10.

DOUBLE.LORES.SOFTSWITCH-.DEMO
DOUBLE.LORES.APPLESOFT-.DEMO
AMPER.DOUBLE.LORES.DEMO
T.AMPER.DOUBLE.LORES
AMPER.DOUBLE.LORES
T.DOUBLE.LORES
DOUBLE.LORES
(1) Demos und Applesoft-Erweiterung für Double-Lores; (2) Heft 1/84, S. 32; (3) IIe (mit 64K-Karte) oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN DOUBLE.LORES-.APPLESOFT.DEMO, RUN AMPER.DOUBLE.LORES.DEMO

HIRES
T.PRINTHIRES
PRINTHIRES
(1) Programm zur Erstellung von Grafik-Hardcopies; (2) Heft 1/84, S. 40; (3) II+ (mit G/K) oder IIe; Parallel-Interface und Epsondrukker der Serie FX, RX oder MX; (4) DOS 3.3 oder ProDOS; (5) RUN HIRES; (6) Erweiterete Version auf Disk #5 (SUPERDUMP).

DHGR.APSOFT.DEMO
AMPER.DOUBLE.HIRES.BAS
AMPER.DOUBLE.HIRES
T.AMPER.DOUBLE.HIRES
DHGR.LINEPLOTTER
(1) Demos und Applesoft-Erweiterung für Double-Hires; (2) Heft 2/84, S. 24; (3) IIe (mit 64K-Karte) oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN AMPER.DOUBLE-.HIRES.BAS

INSTRING.TEST
INSTRING.OBJ
T.INSTRING.OBJ
INSTRING.LISA.SOURCE
(1) INSTRING-Befehl für Applesoft; (2) Heft 2/84, S. 40; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN INSTRING.TEST

LOESCHEN.EINES.ARRAYS
(1) Löschen eines Arrays als Applesoft-Unterprogramm; (2) Heft 2/84, S. 51; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN LOESCHEN.EINES.ARRAYS

ULTRATERM.ENGLISCH
ULTRATERM.DEUTSCH
(1) Zeichensatz-Files für UltraTerm-Video-Karte; (2) Heft 2/84, S. 60; (3) II+ oder IIe; UltraTerm-Karte; EPROM-Brenner; (4) DOS 3.3 oder ProDOS

PRIMZAHLEN.OVERMEYER
PRIM.OBJ0
PRIM.OBJ1
PRIM.TEST
PRIM.TOOLKIT.SOURCE
(1) Primzahlenprogramm (Hauptgewinn aus Peeker, Heft 1/84, S. 58); (2) Heft 2/84, S. 70; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN PRIMZAHLEN.OVERMEYER

Disk #2
(DOS 3.3; Heft 1-2/85)

T.RAMDISKLC
RAMDISKLC
(1) 16K-RAM-Disk für LC; (2) Heft 1-2/85, S. 8; (3) II+ (mit G/K und LC), IIe oder IIc; (4) DOS 3.3; (5) BRUN RAMDISKLC; (7) Patch in Heft 6/85, S. 69 (Leserbriefe).

T.IBS.RAMDISKDRIVER
IBS.RAMDISKDRIVER
T.AP20.RAMDISKTEST
AP20.RAMDISKTEST
(1) Assemblerlisting des RAM-Disk-Drivers der AP 20 mit Testprogramm; (2) Heft 1-2/85, S. 18; (3) II+ oder IIe; RAM-Karte AP 20 der Firma IBS; (4) DOS 3.3; (5) BRUN AP20.RAMDISKTEST

T.QUICKCOPY
QUICKCOPY
(1) Disketten-Kopierprogramm unter ProDOS; (2) Heft 1-2/85, S. 22; (3) IIe (mit 64K-Karte) oder IIc; 2 Laufwerke; (4) ProDOS 1.0.1; (5) BRUN QUICKCOPY; (6) für andere ProDOS-Versionen mit 0F2C:60 Spurprüfroutine desaktivieren (s.S. 27, Zeile 954).
QUICKCOPY.PUFFER
(1) Programm zur Berechnung der Puffergrößen für QUICKCOPY; (2) Heft 1-2/85, S. 22; (3) IIe oder IIc; (4) ProDOS, jede Version; (5) RUN QUICKCOPY.PUFFER
PRODOS.COPYA
T.PRODOS.COPYOBJ
PRODOS.COPYOBJ
(1) Kopierprogramm unter ProDOS für ein Laufwerk; (2) Heft 1-2/85, S. 22; (3) II+ (mit LC), IIe oder IIc; (4) ProDOS 1.0.1; (5) RUN PRODOS.COPYA

PRODOS.PATCH
(1) Patch-Programm zur Anpassung von ProDOS an Apple-Kompatible; (2) Heft 1-2/85; (3) II+ oder Kompatible; (4) DOS 3.3 mit ProDOS-Version 1.0.1 vom 1. Jan. 84; (5) RUN PRODOS.PATCH; (6) Patch-Programm läuft unter DOS 3.3; (7) folgende Zeile muß geändert werden: 570 CMD = 2: ...

T.APPLESOFT.FRE
T.LC.FRE
LC.FRE
FRE.TEST
(1) Schnelle Garbage-Collection-Routine mit Assemblerlisting der Applesoft-Garbage-Collection; (2) Heft 1-2/85, S. 32; (3) II+ (mit LC), IIe oder IIc; (4) DOS 3.3; (5) RUN LC.FRE.TEST

T.RAM.FRE
RAM.FRE
(1) Schnelle Garbage-Collection ohne LC; (2) Heft 1-2/85, S. 32; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) unselbständig; (7) Berichtigung in Heft 8/85, S. 70; neue Version (RAM.FRE.NEU) auf Sammeldisk #8

T.SCHIRMDISK
SCHIRMDISK.LISA.SOURCE
SCHIRMDISK
(1) Programm zur Abspeicherung des 40-Z/Z-Bildschirms auf Diskette; (2) Heft 1-2/85, S. 42; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) BRUN SCHIRMDISK

T.VIDEXT
VIDEXT.LISA.SOURCE
VIDEXT
(1) Programm zur Abspeicherung des Videx-Bildschirms auf Diskette (für IIe s. Disk #4); (2) Heft 1-2/85, S. 42; (3) II+ mit Videx; (4) DOS 3.3; (5) BLOAD VIDEX; CALL 768

GETPAS
T.GETPAS.ASS
GETPAS.ASS
(1) Konvertierung von UCSD-Pascal- in DOS-Textfiles; (2) Heft 1-2/85, S. 68; (3) II+, IIe oder IIc; 2 Laufwerke in Slot 6; (4) DOS 3.3; (5) RUN GETPAS; (6) GETPAS.ASS wird vom BASIC-Programm gepokt

GETDOS.PASCAL.SOURCE
(1) Konvertierung von DOS- in UCSD-Pascal-Textfiles; (2) Heft 1-2/85, S. 70; Erläuterung s. Einleitung; (6) s.a. Disk #9.

COPYDUPDIR.PASCAL-.SOURCE
(1) Hilfsroutine zum Zero-Befehl; (2) Heft 1-2/85, S. 70.; (3) II+ (mit LC), IIe oder IIc; (4) UCSD-Pascal 1.1/1.2; (5) E(xecute) COPYDUPDIR; (6) s.a. Disk #9.

PRODOS.EDITOR.MACROS
(1) Makro-Definitionsprogramm für Besitzer des PRODOS.EDITOR (Hüthig Software Service); (2) Heft 1-2/85, S. 86; (5) auf Arbeitskopie des PRODOS.EDITOR unter dem Namen EDITOR.MACROS kopieren und dann mit RUN EDITOR.MACROS starten.

Disk #3
(CP/M; Heft 1-2/85)

PASS.BAS
MENUE.BAS
HELP.BAS
A.BAS bis N.BAS
(1) Lohn- und Einkommensteuer-Programm für 1984; (2) Heft 1-2/85, S. 45; (3) II+ (mit Videx), IIe; Z80-Karte; (4) CP/M 56; (5) nach Starten von MBASIC RUN "PASS" eingeben; (6) das einzugebende Passwort lautet „PEEKERDUSKE" (entfällt bei späteren Exemplaren der Disk #3)

Disk #4
(DOS 3.3; Heft 3 + 4/85)

TESTGENERATOR
BAHNFAHRT
ZU
TUN.UND.SOLLEN
IRGEND
SAETZE
(1) Übungsprogramm für Legastheniker; (2) Heft 3/85, S. 22; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN TESTGENERATOR; (6) Programm zuvor auf nicht schreibgeschützte Datendiskette kopieren.

MULTIPRECISION
(1) Höhere Rechengenauigkeit bei den Grundrechenarten in Applesoft; (2) Heft 3/85, S. 32; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN MULTIPRECISION

T.WS.TRANSFER
WS.TRANSFER
T.WS.TRANSFER.2
WS.TRANSFER.2
(1) Konvertierung von CP/M-Wordstar- in DOS-3.3-Textdateien; (2) Heft 3/85, S.35; (3) II+, IIe oder IIc; 2 Laufwerke; (4) DOS 3.3; (5) BRUN WS.TRANSFER; (6) WS.TRANSFER.2 ist eine nochmals verbesserte Version, die nicht mehr im Peeker abgedruckt wurde

GETCPM
(1) Konvertierung von CP/M- in DOS-3.3-Textdateien; (2) Heft 3/85, S.43; (3) II+ (mit G/K), IIe oder IIc; 2 Laufwerke; (4) DOS 3.3; (5) RUN GETCPM; (6) kann für 1 Laufwerk umgeschrieben werden (s. Heft)

PRIM.0.SC.SOURCE
PRIM.0.BIN
PRIM.1.SC.SOURCE
PRIM.1.BIN
PRIM.FP
(1) Primzahlenprogramm (2. Platz aus Peeker, Heft 1/84, S. 58); (2) Heft 3/85, S.56; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN PRIM.FP

ACCELERATOR.ABSTELLEN
(1) Programm zum Abstellen der Accelerator-Karte (vgl. Disk #1); (2) Heft 3/85, S. 66; (3) IIe; Accelerator-Karte; (4) DOS 3.3 oder ProDOS; (5) RUN ACCELERATOR.ABSTELLEN

T.WILDCARD.TEST1
WILDCARD.TEST1
T.WILDCARD.TEST2
WILDCARD.TEST2
(1) Testprogramme für Wildcard-Plus-Karte; (2) Heft 3/85, S. 72; (3) II+ (mit Einschränkungen) oder IIe; Wildcard-Plus-Karte; (4) DOS 3.3; (5) unselbständig

XPLOT.DEMO
XPLOT.ROUTINE
T.XPLOT.ROUTINE
(1) Implementierung des XPLOT-Befehls für Applesoft; (2) Heft 4/85, S. 14; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN XPLOT.DEMO

MENUE.GENERATOR
(1) Programm zur Erstellung von Bildschirm-Menüs; (2) Heft 4/85, S. 20; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN MENUE.GENERATOR

T.MACROS.65C02
(1) Makro-Bibliothek für ältere Big-Mac/Merlin-Versionen zur Simulierung der 65C02-Befehle; (2) Heft 4/85, S. 30; (3) II+, IIe oder IIc; (4) DOS 3.3; Big-Mac oder Merlin; (5) unselbständig

TERMINAL
TERMINAL.B
T.TERMINAL.B
(1) Terminal-Programm für Mailboxen; (2) Heft 4/85, S. 34; (3) II+ oder IIe; Serielle Schnittstelle mit 6850-Baustein; (4) DOS 3.3; (5)

KURZBERICHTE

RUN TERMINAL; (7) Anpassung für Super-Serial-Card in Heft 6/85, S. 72 (Disk #6) 9

CAT.ARRAY
CAT.SAVER
EINTRAG.SUCHER
EINTRAG.ANALYSE
PRODOS.READER
T.PRODOS.READER.OBJ
PRODOS.READER.OBJ
(1) Beispielprogramme für ProDOS; (2) Heft 4/85, S. 41; (3) II+ (mit G/K und LC), IIe oder IIc; (4) ProDOS; (5) RUN CAT.ARRAY; RUN CAT.SAVER; RUN EINTRAG.SUCHER; RUN EINTRAG-.ANALYSE; RUN PRODOS-.READER 10,11,12,13,14

MOUSESTUFF.PASCAL-.SOURCE
MOUSE.ASS.PASCAL.SOURCE
TESTMOUSE.PASCAL.SOURCE
DRAWMOUSE.PASCAL-.SOURCE
(1) Implementierung der Maus in UCSD-Pascal 1.1/1.2 (s.a. Disk #9); (2) Heft 4/85, S. 48; (3) II+, IIe oder IIc; Apple-Maus; (4) UCSD-Pascal; (5) E(xecute) TESTMOUSE oder E(xecute) DRAWMOUSE (nach Installierung); (7) „INDIRECT .WORD 0" (S. 57) löschen und statt dessen zu Beginn „INDIRECT .EQU 04" einsetzen, falls Mausprogramm mit altem 6502-Prozessor benutzt wird. 15,16

INALL.DATA
(1) Eingabe von beliebigen Strings für Applesoft; (2) Heft 4/85, S. 70 und 5/85, S. 50; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN INALL.DATA 18

SCREEN80.DATA
SCREEN80.SAVER
(1) Programm zum Ausdrucken (SCREEN80.DATA) oder Abspeichern (SCREEN80.SAVER) des IIe/IIc-80-Zeichenbildschirms (für Videx s. Disk #2); (2) Heft 4/85, S. 33 und 76; (3) IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN SCREEN80.DATA; danach CALL 768 (Drucker in Slot 1 einschalten); RUN SCREEN80.SAVER 11/12

Disk #5
(DOS 3.3; Heft 5/85) 6

T.FM.BSP
FM.BSP
(1) Beispiele zum Aufruf des File-Managers; (2) Heft 5/85, S. 6; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) unselbständig; vgl. Disk #10 (AS-.FILER)

T.SLOTRAMDISK
SLOTRAMDISK
SLOTRAMDISK.HELLO
(1) RAM-Disk-Driver für Balfer-Interface; (2) Heft 5/85, S. 12; (3) IIe; 64K-Karte; Balfer-Interface; (4) DOS 3.3; (5) INIT SLOTRAM-DISK.HELLO (danach SLOTRAM-DISK auf die neue Diskette kopieren)

PLOT.2.0
T.PLOT.B
PLOT.B
PLOT.PROTECTOR
(1) Programm zur Darstellung mathematischer Funktionen; (2) Heft 5/85, S. 17; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) EXEC PLOT PROTECTOR; (7) Korrektur der Tippfehler (Disk #5 war in Ordnung) in Heft 9/85, S. 69 1

T.CONVERT560
CONVERT560
CONVERT560.DEMO
(1) Wandelt ein Double-Hires-Bild in zwei normale Bilder um; (2) Heft 5/85, S. 25; (3) IIe (mit 64K-Karte) oder IIc; (4) DOS 3.3; (5) RUN CONVERT.DEMO (nur in Verbindung mit Epson FX-80/8132W) 2

T.EDA
EDA
(1) Disassembler für „geheime" 6502-Opcodes; (2) Heft 5/85, S. 31; (3) II+ oder IIe; (4) DOS 3.3; (5) BRUN EDA 3

TRANSCEND.PASCAL.SOURCE
(1) Einbindung von transzendenten Funktionen in UCSD-Pascal 1.1/1.2 (s.a. Disk #9); (2) Heft 5/85, S. 35; (3) II+, IIe oder IIc; (4) UCSD-Pascal; (5) als Unit mit Segment-Nr. 29 im SYSTEM.LIBRARY integrieren

T.BLOCKTRACER
BLOCKTRACER
T.BLOCKTRACER1
BLOCKTRACER1
(1) Tracer für Blockzugriff unter ProDOS 1.0.1; (2) Heft 5/85, S. 51; (3) II+ (mit LC), IIe oder IIc; (4) ProDOS 1.0.1; (5) BRUN BLOCK-TRACER; (6) BLOCKTRACER1 für RAM-Disk bei IIe (mit 64K-Karte) oder IIc 4

FORMAT.LC
FORMAT.LC.START
(1) Formatierung von Disketten unter ProDOS 1.0.1; (2) Heft 5/85, S. 56; (3) II+ (mit LC), IIe oder IIc; (4) ProDOS 1.0.1; (5) RUN FOR-MAT.LC.START 5

T.DISKDRIVER.DEMO
DISKDRIVER.DEMO
(1) Demo für direkten Zugriff auf ProDOS-Disk-Driver; (2) Heft 5/85, S. 51; (3) II+ (mit LC), IIe oder IIc; (4) ProDOS 1.0.1; (5) BLOAD DISKDRIVER.DEMO; CALL 768 2 6

RANDOM.DEMO
(1) Echte Zufallszahlen für Applesoft; (2) Heft 5/85, S. 69; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN RANDOM.DEMO; (6) Testprogramm erzeugt 500.000 Zufallszahlen und läuft deshalb mehrere Stunden. 4

COLUMN80.DEMO
(1) Beispiel zur Lösung des HTAB-Problems beim IIe/IIc; (2) Heft 5/85, S. 69; (3) IIe (mit 80 Z/Z) oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN COLUMN80.DEMO 5

SUPERDUMP.EPSON
SUPERDUMP.IMAGEWRITER
SUPERDUMP.BILD
T.SUPERDUMP
SUPERDUMP
EPSON
IMAGEWRITER
(1) Hardcopy-Programm für im Text spezifizierte Rechner/Drucker/Interfaces; (2) Heft 6/85, S.

Brainware
Ihr Experte in Expertensystemen
Consulting · Schulung · Software

ARTIFICIAL INTELLIGENCE FÜR APPLE II und MACINTOSH

C
LISP
PROLOG
MODULA 2
IDEA PROCESSING
EXPERT SYSTEM SHELLS

Fordern Sie unseren Katalog an.

Brainware GmbH
Kirchgasse 24
6200 Wiesbaden
Tel.: 0 61 21-37 20 11

electronic gmbh
Neue Preise # 11

IMB-PC/XT-COMPATIBLE COMPUTER + ZUBEHÖR
Mainboard XT f. 16bit + 256K-RAM + 8 Slots, Leerkarte 79,—
Mainboard XT-640 K (on Band) f. 16 Bit + 8 Slots Leerkarte 89,—
Disk Controller leer (f. 2 x 360 KB-Drives) 29,—
Disk Controller leer (f. 4 x 360KB-Drives) 34,—
Monochrome Graph. + par. Printer Leerkarte (Hercules comp.) 55,—
Color RGB + Video Graphik Leerkarte 55,—
Parallel Printer Leerkarte 29,—
512K-RAM Leerkarte 55,—
Multifunktions Leerkarte (Printer, Ser, Game gepuffert) 49,—
Disk-Multi I/O Leerkarte, (f. 3x360 K-Drive, 2xRS 232 1xpar. Printer int., Game/Joystick Port, Real Time Clock/Kalender gepuff.) 59,—
384 K-Multifunktions Leerkarte, (1 xpar. Printer Int., Game/Joystick Port, Real/Time Clock/Kalender gepuffert) 55,—
Prototypen/Lochraster Leerkarte, durchkontaktiert 88,—
Mainboard XT-256 f. 16 bit CPU, OK (bis 256K- an board, oder 640 K mit Addr. auf Karte aufr.) + Boot-ROM + 8 Slots best. + geprüft 419,—
Mainboard XT-256 nr. o. jedoch 256K– vollbest. und geprüft 499,—
Mainboard XT-640 f. 16 bit CPU, OK (bis 640 K-an board aufr.) sont w. o. 479,—
Mainboard XT-640 w.o. jedoch 640 K – vollbest. und geprüft 799,—
64KB-RAM-Chip-Aufrüstsatz (9 Stück) einzeln geprüft 21,—
256 KB-RAM-Chip-Aufrüstung (9 Stück) einzeln geprüft 159,—
Disk-Controller gepr. (f. 2 x 360 KB Drives) 99,—
Disk Controller gepr. (f. 4 x 360 KB-Drives) 115,—
Monochrome Karte geprüft 215,—
Monochrome Graph. + par. Printer Karte gepr. (Hercules comp.) 345,—
Color RGB + Video Graphikkarte gepr. 185,—
Parallel Printer-Karte gepr. o. Kabel (Extra Kabel = 49,–) 79,—
512K-RAM-Karte gepr. (DKB) 169,—
384K-Multifunktionskarte, RS 232, par. Printer, Interf. Game/Joystick Port, Real Time Clock gepuff., 0K + RAM-Disk/Spooler/Uhr Softw. 345,—
Multifunktionskarte (Par. + Ser. + Uhr gepuffert) gepr. 289,—
Disk Multi I/O Karte (f. 2 x360 K-Drive, 2 xRS 232, 1 x par. Printer Int., Game/Joystick Port, Real Time Clock/Kalender gepuff.) 397,—
Profi-Funktionskarte (f. color-ASCII + 15er Block 219,—
135W-Netzteil mit eingebautem Ventilator, Kurzschlußfest 229,—
360KB-Diskdrive Slimline ohne Gehäuse 319,—
360KB-Disk Drive slimline-TEAC-FD 55 B-2 x40 Tr. o. Geh. 399,—
PC/XT – KOMPLETT-SYSTEME EINGEBAUT UND GEPRÜFT
XT-256, 16 bit-Rechner (incl. Boot-Eprom) 128K-Ram im IBM-look-like Met.-Gehäuse + Profi-Fkt. Tast. +135 W-Netzt. gepr. 1097,—
XT-256, w.o.+ Disk Contr. +360KBDrive + Color RGB+Video Graphic-Karte 1697,—
XT-256, w.o. jedoch statt Color mit Monochrome Karte 1747,—
XT-256, w. o. jedoch mit Monochrome-Graphic-Printer-Karte 1857,—
IMB-PC-Systemeinheit + Laufwerk 3.719,—
IMB deutsche DIN-Tastatur 698,—
IBM-Bildschirm Color 1.698,—
PC 10-Commodore dt. Test. 4.475,—
GENIE 16 G (256K-RAM bis 640K auf M-board) 2 x 360K-Drive, ser. + par. Printer – int., + Herbgraphik-16 Color-Karte + 6 slots, – incl. dt. Testatur und MS–DOS 4.895,—

APPLE-BUS COMPUTERPLATINEN + PERIPHERIE
Motherboard II 48K leer, durchkont. mit Lötstopplack 59,—
Motherboard II w.o. jedoch 64 K leer 78,—
M-board II 48K, gesockelt, vollbest. + gepr. 349,—
Motherboard II 64K gesockelt, vollbest. + gepr. 408,—
M-board II 64K + Z80 CPU, gesockelt, v. best. + gepr. 438,—
M-board IIe 64K vollbestückt + gepr. 449,—
APOLLO II Kunststoff-Leergehäuse 85,—
APOLLO II Gehäuse incl. Funkt.-Tast. 217,—
Superstarkes Netzteil – Kurzschlußfest 115,—
Leerplatine: 16K-RAM, Z80/CPU-, Controller DOS 3.3, Parallel-Druckerkarte, Ser. Int. V24, 80Z/24Z-Karte
PAL-Modulatorkarte, je 18,95
Entwicklungslochraster-Leerplatine durchkontaktiert 27,—
16K-Speicher-Karte geprüft 79,—
APPLE orig. 16K-Language Karte gepr. 125,—
Z-80 CPU-Karte geprüft 74,—
Controller DOS 3.3-Karte geprüft 73,—
Auto-Controller DOS 3.2/3.3-Karte gepr 119,—
Erphi AFDC 2 Controller + Autoposch Softw. + Handbuch 288,—
Parallel-Drucker-Karte geprüft 74,—
Par.-Drucker-Grappler comp. Leerplatine 27,—
Par.-Drucker Karte-Grappler comp. geprüft 169,—
8bit par. Grappler-Int. + 64K-Buffer + Kabel gepr. 445,—
Serell-Interface-Karte V24 gepr. geprüft 95,—
Super-Serell Leerkarte 34,—
Super-Serell-Interface Karte gepr. 198,—
80-Zeichen/24-Zeilen-Karte geprüft 119,—
80Z/24Z-Karte + Softswitch-Schalter Leerplatine 24,—
80Z/24Z-Karte + Softswitch-Schalter geprüft 139,—
80Z/24Z-Softmitch Karte gepr. für IIe 49,—
80Z/24Z + 64K-Ram + Softsw. Leerkarte f. IIe 27,—
80Z/24Z + 64K-Rem + Softsw.-Karte best.+gepr. f. IIe 79,—
IEEE-488 Int. Karte gepr. 345,—
PAL-Modulator/Color-Karte geprüft 99,—
PAL-RGB-Color Karte gepr. 149,—
128K-Speicher-Leerkarte 29,—
128K-Speicher-Leerkarte + Softwre + Menual 99,—
128K-Speicher-Karte + Software geprüft +Manual 298,—
256K-Speicher Leerkarte 34,—
256K-Speicher Leerkarte + Software +Manual 109,—
256K-Speicher-Karte + Software geprüft +Manual 455,—
Meus incl. Software 179,—
EPROM-Burner (2708/16/32/64) geprüft 119,—
6522-VIA Karte geprüft 135,—
Clock Karte geprüft 97,—
Speech Karte geprüft 56,—
Musik Karte geprüft 107,—
Wild Karte geprüft 95,—
AD/DA-8Bit Karte gepr. 375,—
Accelerator Karte gepr. 775,—
Modem Eurocom-Karte gepr. + Software + Man. 397,—
UHF/TV-Modulator universell 27,50
Lüfter anclipsbar (220 V) 95,—
Datapion s2fd Akustikkoppler-FTZ mit RS 232/V24 Anschl. 275,—

APOLLO//(48K) + UHF-Mod. + Gr/Kl vollbest. + gepr. + F-T. 629,—
APOLLO//(48K) + Disk II F + Contr. +12"-Monitor 1.319,—
APOLLO//A (48K) + Gr/Kl + 15er-Tastatur vollbest. + gepr. + F-T. 689,—
APOLLO//A (48K) + Disk II F + Contr. +12"-Mon. 1.379,—
APOLLO// ASKP (48K) vollbest. + gepr. Sep. Keyboard + Gr/Kl + 15er-Block mit Funkt.-Tasten im „IBM-look like" Gehäuse 839,—
APOLLO// ASKP (48K) w.o. + Disk II F + Contr. 1.289,—
APOLLO// ASKP (48K) w.o. hedcog 2 x Disk II F + Contr. 1.639,—
Apollo II ASFK (48K) w.o.+ 2 x Disk II-F+ Contr.+12" Mon. 1.899,—
Apollo IIe (64K) PAL vollbest. und geprüft 749,—
Apollo IIe (128K) + 80Z/24Z-Karte gepr. 825,—
Apollo IIe (64K) + Disk II F+12" Monitor 1.439,—
Aufpreis f.d. Systeme von 48K zu 64K 89,—
Aufpreis von 48K zu 64K+Z80 CPU 89,—

Appel IIc 128K+ 80Z/24Z Karte+Softw. 2.235,—
APPLE-DuoDisk, 2 x 143 KB + Controller orig. 2.185,—
APPLE-Disk//Laufwerk + Contr. + Kab. DOS 3.3 + ProDOS orig. 777,—
APPLE-Disk//(2 Laufwerk) orig. 666,—
Epson-Duodisk f. IBM-/Similne 2 x143 KB mit Geh. 799,—
Disk//+Contr. + Kabel DOS 3.3 (Siemens) im Geh. 555,—
Disk//(2 Laufwerk, Siemens) im Geh. 486,—
Disk//F + Contr. + Kab. DOS 3.3 (1/2 Höhe) im Geh. 419,—
Disk//F (2. Laufwerk, 1/2 Höhe) i. Geh. 349,—
Erphi DuoDisk 1,2 MByte incl. Netzteil + Contr. 1.789,—

APPLESOFT + Turortel + Reference Manuel engl. 75,—
APPLE-DOS 3.3 Manuel engl. 25,—
APPLE-DOS 3.3 Hb. dt. 45,—
APPLE ProDOS HB + Disk orig. 69,—
APPLE-Pascal Reference + Operating Manual engl. 45,—
APPLE-Pascal UCSD 1.1. Disks + Manuals orig. 175,—
APPLE-Pascal Reference Hb. dt. 49,—
APPLE-Pascal Language Hb. dt. 39,—
APPLE-Forth Manuel engl. 69,—
APPLE-Fortran Manual engl. 69,—
CP/M-Software Vol. I + II, Manuel engl. 89,—
APPLE // Pascal-Handbuch deutsch (te-wi) 59,—
APPLE //-Anwenderhandbuch deutsch (te-wi) 58,—
6502-Programmier-Handbuch deutsch 45,—
6502-Anwendungen Handbuch deutsch (Rod. Zaks) 38,—

APPLE/c (128K) dt. Tastatur 2.495,—
Z80 Karte gepr. f. APPLE/C + RAM-Disk-Software + Hb. 489,—
Macintosh System (128K) 6.745,—
Macintosh System (512K) 7.495,—
Macintosh Umrüstung von 128K auf 512K 679,—

SANYO MSX-Computer MPL 64 dt. Testatur 685,—

Epson LX 80 F/T, 8 Bit/par. 100Z/S. mit der Super N.L.Q. Qualität 987,—
EPSON GX 80+, 939,—
EPSON RX 80+, 8bit/parallel 849,—
EPSON RX 80F/T+, 8bit/parallel 999,—
EPSON RX 100+, 8bit/parallel 1.279,—
EPSON FX 85F/T+, 8bit/parallel 1.299,—
EPSON FX 105+, 8bit/parallel 1.719,—
EPSON LQ 1500+, 8bit/perellel 3.269,—
EPSON Traktorzusatz für FX 80 129,50
APPLE/EPSON-Drucker Grepho-Interface + Kabel 169,—
MX 80/82, FX/RX 80 Spezialfarbband-Kassette 15,95

BROTHER-Typenraddr., HR 15XL 3bit/par, neuestes Mod. 1.387,—
Seikosha SP 800 N.LP (A=EPSON+, I=IBM-Comp.) 819,—
STAR SG10, NLQ- + IBM-Zeichensatz, 8bit/par 998,—

Zenith ZVM 123 gr. 12" oder Zenith ZVM or. 12" 15 MHz 277,—
Zenith ZVM 124 gr. 12" 22 MHz IBM-Comp. 395,—
SANYO DM 2212 12" orange 15 MHz 265,—
SANYO 9112 Monochromer Monitor 12", gr. IMB-Comp. 587,50
Monochromer 12"-Monitor entspiegelt gr. o. or. 22 MHz 389,—

Disketten in Box + Aufkleber i. Weiß, 10er Pack/100er Pack,/ Stückpreis:

5¼"Verstärkt BASF	3M SCOTCH	FUJI	DATA MAGNE-TICS	NEUTRAL
1X, SS/SD 3,99/ 3,79			2,66/2,76	
1D, SS/DD 4,19/ 3,99	4,88/ 4,58	4,27/ 4,07	3,96/3,76	2,75/2,65
2D, DS/DD 5,69/ 5,69	6,58/ 5,98	5,87/ 5,67	4,76/4,46	3,85/3,85
1D/96TPI 5,79/ 5,59	7,18/ 6,68			
2D/96TPI 7,19/ 7,09	7,88/ 7,48	7,97/ 7,77		
HD1.2MB 13,89/12,89		14,97/13,87		
1X, Hard 4,49/ 4,39				
8"				
1X, SS/SD 4,49/ 4,39	6,68/ 8,48	8,97/ 6,67		
1D, SS/DD 4,79/ 4,59	8,58/ 5,28			
2D, DS/DD 7,05/ 6,89	8,98/ 8,58	10,17/ 9,77		
1X, Hard 4,49/ 4,39	6,68/ 6,48			
3½"				
Einseitig 10,99/ 9,99	10,08/ 9,88	7,87/ 7,57		
Zweiseitig 14,99/14,29	15,98/15,18			

Das sind Preise!

3M-Scotch 5¼" Flüssig-Reinigungsset (2 Disk.) zweiseitig 48,95
BASF 5¼" Reinigungsdiskette einseitig 48,50
5¼" Disketten-Archivbox für 10 Disks (1 St./10 St.) 4,75/43,50
5¼" Disketten-Archivbox für 15 Disks 3M 19,95
5¼" Disk-Kartelkasten Kunststoff (ca 80 Disk) 26,50
5¼" Disk-Kartelkasten Kunststoff (ca. 80 Disk) Rauchglas 39,95
5¼" Disk-K.K.Rauchglas, abschließb. (f. ca. 80-100 Disks) 47,50
2000 Bl. Tabpapier (24 cm x 12" perf.) weiß oder grün/weiß perf. 45,—
4000 Et-Aufkl. doppelr (107 x 36 mm auf 240 x 12" perf. Trägerp.) 66,50
16000 Etiketten-Aufkleber w.o. 249,—
4000 Etik.-Aufkl. einreih. (107 x 36 mm auf 125 x12" perf. Träg.) 65,—

OSZILLOSCOPE HAMEG ab Lager
Bei Vorauszahlung frei Empfangsstation unversichert in der BRD, ausgenom mein Papier und Etiketten, sonst N.N. + V.S. ab DM 30,— Preise incl. MwSt.
Öffnungszeiten: Mo, Di, Do, Fr v. 10-18 h, Mi u. Sa v. 10-14 h, la Sa v. 10-18 h.
Telef. Best.: Mo, Di, Do, Fr von 10-19 h, Mi u. Sa wie Öffnungszeiten.
GEWÄHRLEISTUNG: 6 Monate, auf alle bei uns gekauften Geräte, durch unsere eigene Service-Werkstatt.
REPARATUREN an Apple + Compatible Geräte + Zubehör führt unser Spezialistenteam garantiert zuverlässig + besonders kostengünstig aus.
Sprechen sie mit uns. Kostenvorschlag auf Wunsch!

electronic gmbh
Telex. 07 72 642 aaa-d
Habsburgerstraße 134
7800 FREIBURG, Tel. (07 61) 27 68 64
Bauelemente – Bausätze – µP's
Meßgeräte – Zubehör – Fachliteratur
Fachgeschäft für Elektronik + Mikrocomputer

22; (3) II+, IIe oder IIc; Drucker; (4) DOS 3.3; (5) RUN SUPERDUMP.EPSON (SUPERDUMP.-IMAGEWRITER); (7) Anpassung für IIc in Heft 8/85, S. 50

MACMONITOR
Der MACMONITOR aus Heft 5/85, S. 41, ist noch als 3,5-Zoll-Diskette erhältlich (solange Vorrat reicht)

Disk #6
(DOS 3.3; Heft 6/85)

HELLO
ASMDIV
(1) Installierung von Diversi-DOS 2-C; (2) Heft 6/85, S. 72; (3) II+, IIe oder IIc; (4) Diversi-DOS 4-C; (7) Installierung unter DOS 3.3 ist in Heft 10/85, S. 69 (Leserbrief) beschrieben

CURSOR1
T.CURSOR1
CURSOR2
T.CURSOR2
LINIE
T.LINIE
VIERECK
T.VIERECK
BOX
T.BOX
HINTERGRUND
T.HINTERGRUND
PAGE.SWAP
T.PAGE.SWAP
(1) Hilfsprogramme für Grafik-Editor; (2) Heft 6/85, S. 6; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) unselbständig; (7) Hinweis zu den Cursoren in Heft 10/85, S. 63

WANDERNDER.STRICH
KOMPRESSOR.DEMO
KREIS.1
KREIS.2
KREIS.3
FLIPPER
T.FLIPPER
KOMPRESSOR
T.KOMPRESSOR
(1) Pseudo-Double-Hires; (2) Heft 6/85, S. 16; (3) II+ (IIe oder IIc); (4) DOS 3.3; (5) RUN WANDERNDER.STRICH; RUN KOMPRESSOR.DEMO; RUN KREIS.1; RUN KREIS.2; RUN KREIS.3

OLYMPIA
T.OLYMPIA
(1) Hardcopy-Programm für Typenraddrucker; (2) Heft 6/85, S. 34; (3) II+ oder IIe; Olympia ESW 100 RO; (4) DOS 3.3; (5) BLOAD OPLYMPIA; CALL 32768

FOURIER.MAIN
FOURIER.SYN
FOURIER.SPEC
(1) Programm zur Fourier-Analyse; (2) Heft 6/85, S. 38; (3) II+ (mit Videx), IIe oder IIc; Drucker mit Grafik-Interface; (4) DOS 3.3; (5) RUN FOURIER.MAIN; (7) in Heft 9/85, S. 69 sind 20 Zeilen nachgetragen, die im Listing (nicht auf Diskette) fehlten

AS.ERWEITERUNG
T.AS.ERWEITERUNG
AS.ERW.PROSTART
AS.ERW.PRO
T.AS.ERW.PRO
(1) Applesoft-Erweiterungen; (2) Heft 6/85, S. 43; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) BRUN AS.ERWEITERUNG; (6) für ProDOS gilt RUN AS.ERW.PROSTART

INSTALL.PASCAL.SOURCE
RAMDISK94.PASCAL.SOURCE
INIT.PASCAL.SOURCE
(1) RAM-Disk-Driver für UCSD-Pascal 1.1; (2) Heft 6/85, S. 48; (3) IIe (mit 64K-Karte) oder IIc; (4) UCSD-Pascal 1.1, nicht 128K-1.2! (5) Disk #9 enthält fertig assemblierte Version, die mit E(xecute) INSTALL gestartet wird; (7) im Listing (nicht auf Disk #6) wurde ein Teil weggeschnitten, der in Heft 7/85, S. 52 nachgetragen wurde

RAMDISK.INIT.DOS
AUXDRIVER
T.AUXDRIVER
MOVEDRIVER
T.MOVEDRIVER
RAMDISK.FORMATTER
T.RAMDISK.FORMATTER
(1) RAM-Disk-Driver für CP/M 2.2; (2) Heft 6/85, S. 56; (3) IIe mit 64K-Karte; (4) CP/M 56K (nicht 60K oder 64K!); (5) s. Heft

SOLITAIRE.START
SOLITAIRE
SOLITAIRE.B
T.SOLITAIRE.B
(1) Programm zur Lösung des Solitaire-Problems; (2) Heft 6/85, S. 64; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN SOLITAIRE.START

Disk #7
(DOS 3.3; Heft 7/85)

PYRAMID.PITTY
T.PYR.PITTY.0
T.PYR.PITTY.1
PYR.PITTY.0
PYR.PITTY.1
PYR.PITTY.BACK
PYR.PITTY.SHAPE
(1) Ein Reaktionsspiel; (2) Heft 7/85, S. 6; (3) II+, IIe oder IIc (keine externe Tastatur); (4) DOS 3.3 oder ProDOS; (5) RUN PYRAMID-.PITTY

T.MEGAWARP.REL
MEGAWARP.REL
T.MEGAWARP.9900
MEGAWARP.9900
T.SPEEDTEST
SPEEDTEST
(1) RAM-Disk-Testprogramme für die AP33 von IBS; (2) Heft 7/85, S. 8; (3) II+ oder IIe; AP33-Karte; (4) ProDOS; (5) unselbständig: BRUN INSTAL (von IBS-Diskette); dann BRUN MEGAWARP.9900 (oder MEGAWARP.REL); (6) Beschreibung im Heft beachten.

FORMAT
T.FORMAT.OBJ
FORMAT.OBJ
(1) Formatierungsprogramm für bis zu 80 Spuren; (2) Heft 7/85, S. 20; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN FORMAT; (7) Optimierung der Armbewegung im Heft 9/85, S. 69

BITEDITOR
NORMAL
FETT
FETT.INVERSE
(1) Editor zur Erstellung eines Zeichensatzes; (2) Heft 7/85, S. 29; (3) II+ mit Videx; EPROM-Brenner; (4) DOS 3.3; (5) RUN BITEDITOR; (6) NORMAL, FETT und FETT.INVERSE sind Zeichensatz-Files

PASTOPRO.1D
PASTOPRO.2D
T.PASTOPRO.O
PASTOPRO.O
(1) Übertragung von UCSD-Pascal-Textdateien nach ProDOS; (2) Heft 7/85, S. 62 (Pascal-Preisausschreiben); (3) II+ (mit LC), IIe oder IIc; (4) ProDOS; (5) RUN PASTOPRO.1D (oder PASTOPRO.2D für 2 Laufwerke); (6) PASTOPRO.O wird durch PASTOPRO automatisch gepokt

T.CONVERT
CONVERT
(1) Umwandlung von Hex- (32 Bits) in FP-Zahlen; (2) Heft 7/85, S. 69; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) BRUN CONVERT

T.VORLESER
VORLESER
(1) Akustische Ausgabe von Hex-Dumps mit der Sprachkarte S.A.M.; (2) Heft 7/85, S. 71; (3) II+ oder IIe; Sprachkarte S.A.M.; (4) DOS 3.3; (5) BRUN VORLESER

Disk #8
(DOS 3.3; Heft 8/85)

HELLO
ASMDIV
(1) Installierung von Diversi-DOS 4-C; (2) Heft 8/85, S. 68; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) RUN HELLO; (6) Bitte $30,00 an DSR überweisen.

DISKTEST
DISKTEST.START
T.DISKTEST
(1) Testprogramm für Disketten und Laufwerke; (2) Heft 8/85, S. 6; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) EXEC DISKTEST.START

KOPY
BATCHKOPY
T.GETSETINFO
GETSETINFO
BILDTEST
(1) Datei-Kopierprogramm für ProDOS; (2) Heft 8/85, S. 18; (3) II+ (mit LC), IIe oder IIc; (4) ProDOS, BASIC.SYSTEM 1.1; (5) RUN KOPY oder RUN BATCHKOPY; (6) GETSETINFO wird von KOPY/BATCHKOPY gepokt, BILDTEST dient nur als Demo

T.BOX.COPY
BOX.COPY
T.HSCRN
HSCRN
GRAF.QUATTRO.1
(1) Hilfsprogramme für Grafik-Editor; (2) Heft 8/85, S. 24; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) unselbständig; (6) GRAF.QUATTR0.1 ist eine Zusammenstellung aller Grafquattro-Routinen

T.DOUBLE.LORES
DOUBLE.LORES
DOUBLE.LORES.DEMO
(1) Double-Lores-Applesoft-Erweiterung; (2) Heft 8/85, S. 32; (3) IIe (mit 64K-Karte) oder IIc; (4) DOS 3.3; (5) RUN DOUBLE.LORES.DEMO

AD-START.CMD
HMENUE.CMD
AUFNAHME.CMD
AUFMASKE.CMD
AUSMASKE.CMD
EDITFNAM.CMD
SUCHFNAM.CMD
SUCHVNAM.CMD
SUCHBEME.CMD
SCHREIBA.CMD
SCHREIBL.CMD
LOESCH.CMD
SUCH.CMD
(1) Adreßverwaltungsprogramm unter dBase; (2) Heft 8/85, S. 40;

Wollen Sie mit Ihrem MACINTOSH grafisch arbeiten?

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

Wir stellen aus:
SYSTEMS 85, München
28. 10. – 1. 11. 85
Halle 15 OG
Stand B8

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

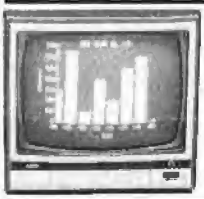
Summagraphics® LIMITED
Niederlassung Deutschland

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

Franklin Disc-Drive-Card für APPLE II u. IIe

Voll Apple tauglich.
Für 2 Disc-Drive.
Best.-Nr. 94014
NUR **DM 48,90**

Fordern Sie unseren kostenlosen Katalog an.

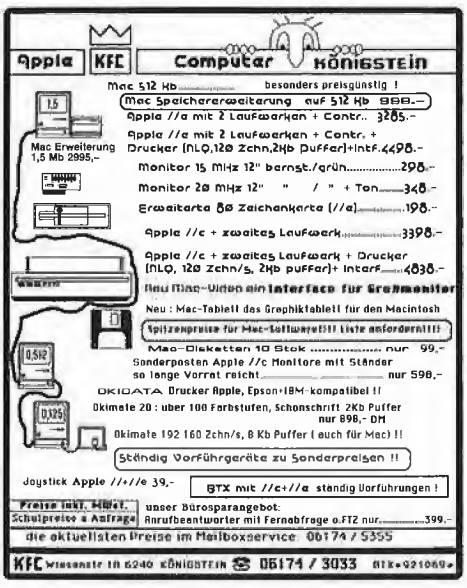


9" Monitor. Monochrom grün entspiegelt. Auflösung über 20 MHz. BAS/Video-Eingang. Eigenes Netzteil 220 V. Passend für alle Apple und Commodore und ähnliche.
Best.-Nr. 65032 NUR **DM 199,90**

Garantie 1 Jahr. Eig. Kundendienst.

Bühler Computer, Postfach 32, 7570 Baden-Baden, Tel. (07221) 71004

Bühler Shop, Waldstraße 46, 7500 Karlsruhe

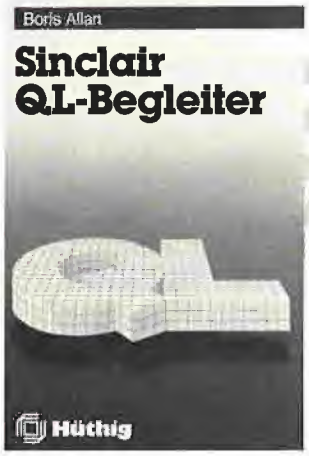


Computer-unterstütztes Lernen

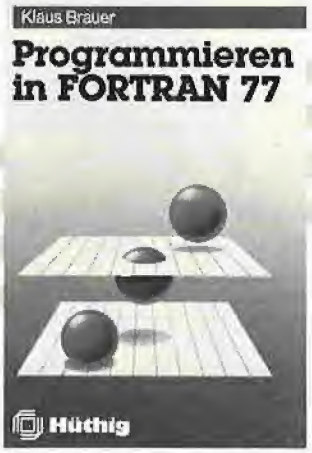
- Programme für Apple IIe/IIc/teilw. + Maschinenschreiben, Basic, Wortschatztrainer, Computer-Simulator, Schnellesen, Rechtschreibtrainer, Deutsche Grammatik, Mathe, Physik/Chemie, Fremdsprachen, Fahrschule, usw.
- Demo-Diskette mit 9 Programmen DM 10,–
- Gesamtkatalog kostenlos.
- Frei-Programme (Public Domain) Liste mit über 5000 Programmen, DM 10,– (bar oder Briefmarken)

INTUS SOFTWARE
Kaiserstr. 21. 7890 Waldshut, Telefon: 07751 - 7920

Computerbücher
die gehen, für Computer die kommen.



Boris Allan
Sinclair QL-Begleiter
1985, ca. 100 S., kart., DM 35,–
ISBN: 3-7785-1101-7



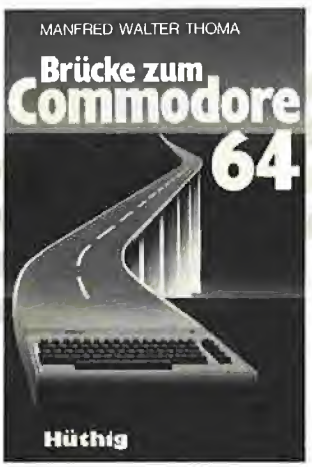
Klaus Brauer
Programmieren in FORTRAN 77
1985, 308 S., kart., DM 46,–
ISBN: 3-7785-1068-1

Wolfgang Eggerichs
dBASE II
Band 1: Einführung
2., verb. Auflage 1985
180 S., kart., DM 39,80
ISBN: 3-7785-1147-5

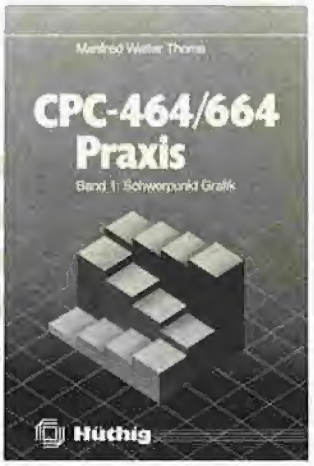
Wolfgang Eggerichs,
Roman Weiß
CBASIC
Das Einführungs- und Nachschlagewerk für den Anwender
1985, 172 S., kart., DM 39,80
ISBN: 3-7785-1015-0

Jürgen Kehrel
Apple-Assembler lernen
Band 1: Einführung in die Assembler-Programmierung des 6502
1985, 180 S., kart., DM 38,–
ISBN: 3-7785-1151-3

Arne Schäpers
Bewegte Grafik
Apple DOS Toolkit Erweiterungen
1985, 305 S., 6 Abb., kart., DM 58,–
ISBN: 3-7785-1150-5



Manfred Walter Thoma
Brücke zum Commodore 64
Erweitertes Handbuch
1985, 277 S., kart., DM 46,–
ISBN: 3-7785-1095-9



Manfred Walter Thoma
CPC 464/664-Praxis
Band 1: Schwerpunkt Grafik
1985, 188 S., kart., DM 34,–
ISBN: 3-7785-1149-1

Weitere Titel und Informationen finden Sie in unserem Computerbuch-Katalog:
Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1

Hüthig

Ausgabe und **Eingabe** mit
TYPETERM®
im Slot Ihres
APPLE II/IIe

Das bedeutet: Computertextverarbeitung von der Schreibmaschinentastatur. Steckerfertig ohne Umbau. Kein weiteres Interface erforderlich.

TYPETERM – ein starkes Interface für starke Maschinen!

- Alle Cursor- u. CTL-Befehle bei Eingabe.
- 2 Zeichensätze deutsch/ASCII (m. ASCII-Typenrad)
- Für alle Betriebssysteme
- Hoch/Tiefstellen, autom. Unterstreichen
- Var. Zeichen- u. Zeilenabstände
- Autom. Papierzuführung mit CF-100 (ab EM-80) vom Stapel

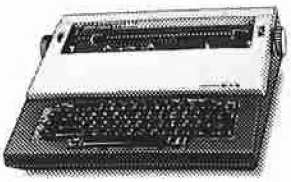
Ausgabe mit
TYPETERM® JUNIOR

- 3 verschiedene Schriftstärken
- Automatisches Unterstreichen
- 2 Zeichensätze deutsch/ASCII (m. ASCII-Typenrad)
- 2 Zeichenabstände
- Für alle Betriebssysteme

ab **DM 899,–** incl. AX-10

AX-10

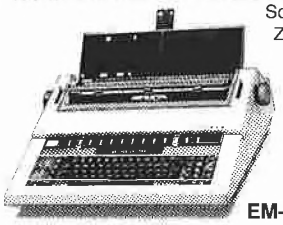
TYPETERM-Interface
für CE-51 bis EM-250
DM 479,–



CE-51

TYPETERM - Mit diesem steckfertigen Interface an Ihrer BROTHER-Typenradschreibmaschine lassen Sie Ihren APPLE korrespondenzreif schönschreiben. Wenn Sie wollen, benutzen Sie mit allem Komfort die Schreibmaschinentastatur zur Eingabe für Ihre Textverarbeitung oder als Ersatz für Ihr Keyboard. Anschließend druckt oder schreibt Ihre Schreibmaschine wahlweise ASCII- oder deutschen Zeichensatz, Hoch- und Tiefstellungen, automatisch unterstreichend in variablen Zeichen- und Zeilenabständen, viele Sonderzeichen und vieles andere mehr. Selbstverständlich für alle Betriebssysteme. Ideal für den beliebten Wordstar unter CP/M. Die moderne elektronische Typenradschreibmaschine steht Ihnen natürlich auch allein weiterhin unverändert zur Verfügung.

TYPETERM JUNIOR - wahlweise mit AX-10 oder CE-25. Unsere besonders günstigen Gespanne zur Ausgabe am APPLE, ebenfalls steckfertig. Mit Ihrer Textverarbeitung und TYPETERM JUNIOR können Sie dann selbstverständlich auch in verschiedenen Schriftstärken wahlweise im deutschen oder ASCII-Zeichensatz schönschreiben, zentrieren, automatisch unterstreichen und rechts- oder linksbündig schreiben. Für alle Betriebssysteme. Auch als moderne elektronische Typenradschreibmaschinen können sich AX-10 und CE-25 sehen lassen: Lift-off-Korrektur, schneller Typenradwechsel im Drop-In-System für viele verschiedene Schriftarten, 40-bzw. 20-Zeichen-Korrekturspeicher usw. usw.



EM-80

Paketpreise:
Alle Maschinen bereits incl. TYPETERM Interface . (DM 479,–)

CE- 51 mit TYPETERM	1348,–
CE- 60 mit TYPETERM	1652,–
CE- 61 mit TYPETERM	1737,–
CE- 68 mit TYPETERM	2297,–
CE- 70 mit TYPETERM	2758,–
EM- 80 mit TYPETERM	2037,–
EM- 85 mit TYPETERM	3072,–
EM-100 mit TYPETERM	3072,–
EM-250 mit TYPETERM	5075,–

Cable Kit A (immer erford. ab EM-80)	103,–
AX-10 mit TYPETERM JUNIOR	899,–
CE-25 mit TYPETERM JUNIOR	1048,–
TYPETERM-Kit für CE-50	468,–

brother
Die Zukunft heute

interkom electronic
Kock & Mreches GmbH
Postf. 3004 Isernhagen 4
Telefon 0 51 39 - 8 73 93

(3) II+ (mit Videx) oder IIe; Z80-Karte, 2 Laufwerke; (4) CP/M; dBase; (5) DO HMENUE (nach Start von dBase); (7) Nachtrag in Heft 10/85, S. 21

IDSEARCH
(1) Schlüsselwörter in UCSD-Pascal halbfett drucken (s.a. Disk #9); (2) Heft 8/85, S. 48; (3) II+, IIe oder IIc; (4) UCSD-Pascal; (5) E(xecute) IDSEARCH

FAKULTAET.DEMO
T.FAKULTAET
FAKULTAET
(1) Berechnung von Fakultäten bis 9999!; (2) Heft 8/85, S. 56; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN FAKULTAET.DEMO

GRAFIK.DEMOS
(1) Demos zur Anwendung der HGR-Grafik; (2) Heft 8/85, S. 67; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN GRAFIK-.DEMOS

ZEICHENJAGD
(1) Reaktionsspiel als Applesoft-Einzeiler; (2) Heft 8/85, S. 69; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN ZEICHENJAGD

T.RAM.FRE.NEU
RAM.FRE.NEU
(1) Neufassung von RAM.FRE; (2) Heft 1-2/85, S. 22; (3) II+, IIe oder IIc; (4) DOS 3.3; (5) unselbständig; (6) die Fassung auf Disk #2 enthält einen Fehler

Disk #9
(UCSD-Pascal; Hefte 1-2/85 - 10/85)
Folgende Programme sind bereits auf anderen Sammeldisketten im DOS-3.3-Format erschienen.

DUPDIR.TEXT
DUPDIR.CODE
(vgl. Disk #2)

GETDOS.TEXT
GETDOS.CODE
(vgl. Disk #2)

MOUSESTUFF.TEXT
MOUSESTUFF.CODE
TESTMOUS.TEXT
TESTMOUS.CODE
DRAWMOUSE.TEXT
DRAWMOUSE.CODE
MOUSE.ASS.TEXT
MOUSE.ASS.CODE
MOUSE.LIBRARY
(vgl. Disk #4)

TRANSCEND.TEXT
TRANSCEND.CODE
(vgl. Disk #5)

RAMDISK94.TEXT
INIT.TEXT
INSTALL.TEXT
RAMDISK94.CODE
(vgl. Disk #6)

IDSEARCH.TEXT
IDSEARCH.CODE
(vgl. Disk #8)

ALLG.TEXT
ALLG.CODE
MUEL.TEXT
MUEL.CODE
(1) Konvertierung von ProDOS- in Pascal-Textfiles; (2) Heft 9/85, S. 58; (3) II+ (mit LC), IIe oder IIc; (4) UCSD-Pascal 1.1/1.2; (5) E(xecute) ALLG.CODE (oder MUEL.-CODE); (6) ALLG.CODE oder MUEL.CODE muß in Laufwerk #4 sein.

CRUNCH.TEXT
CRUNCH.CODE
MOVER.TEXT
MOVER.CODE
CRUNCHER.TEXT
CRUNCHER.CODE
(vgl. Disk #10)

Disk #10
(DOS 3.3; Heft 9 + 10/85)

T.AS.FILER
AS.FILER
STARTUP
ENDUP
RAMDISKLC
(1) Datei-Kopierprogramm als Applesoft-Erweiterung; (2) Heft 9/85, S. 12; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3 (auch 64K); (5) RUN STARTUP; RUN ENDUP (6) RAMDISKLC (von Disk #2) wird von STARTUP aufgerufen

MKBOOT.BAS
T.MKBOOT.OBJ
MKBOOT.OBJ

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

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

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

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

GRAFIK.EDITOR
GRAF.QUATTRO.1
(1) Editor für Hires-Grafik; (2) Heft 10/85, S. 6; (3) II+ (mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN GRAFIK.EDITOR; (6) GRAF-.QUATTRO.1 (schon auf Disk #8) wird von GRAFIK.EDITOR automatisch geladen

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

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

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

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

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

UCSD.TEXT
TURB.PAS
(1) Pascal-Kompaktkurs für Applesoft-Programmierer; (2) Heft 10/85, S. 46; (3) II+, IIe oder IIc; (4) UCSD.TEXT für Apple-Pascal 1.1/1.2 und TURB.PAS für Turbo-Pascal 3.0 oder älter; (5) E(xecute) oder R(un); (6) vorher UCSD.TEXT mit GETPAS auf UCSD-Pascal-Format und TURB.PAS mit APDOS auf CP/M-Format konvertieren; Pascal-Kurs als 16seitiger Sonderteil in Heft 12/85

Disk # 11
(DOS 3.3; Heft 11/85)

SPRITE.EDIT
T.SPRIT.EDIT.ASS
SPRITE.EDIT.ASS
T.ASTA
ASTA
T.ASTA.DEMO
ASTA.DEMO
ASTA.DEMO.1
AD1
AD2
AD3
AD4
AD5
(1) Grafik-Animation mit bewegten Sprites und Sprite-Editor; (2) Heft 11/85, S. 6; (3) II+ (für SPRITE.-EDIT mit G/K), IIe oder IIc; (4) DOS 3.3; (5) RUN ASTA.DEMO.1 oder BLOAD ASTA; BRUN ASTA-.DEMO

T.PLAKAT
PLAKAT.INSTALL

Inserentenverzeichnis peeker 11/85

PLAKAT
PLAKAT.DEMO
(1) Vergrößerung einer HGR-Seite zum Ausdruck von Plakaten; (2) Heft 11/85, S. 20; (3) DOS 3.3; (4) II+ (mit G/K), IIe oder IIc; (5) RUN PLAKAT.DEMO; Anpassung des Applesoft-Programms an eigenes Drucker-Interface unbedingt erforderlich!

DISK.CHIRURG
(1) Überprüfung einer Diskette auf schadhafte Sektoren; (2) Heft 11/85, S. 24; (3) DOS 3.3 (nur 48K); (4) II+, IIe oder IIc; (5) RUN DISK-.CHIRURG

FILER.0
FILER.1
FILER.2
SYSTEM.PATCH.0
SYSTEM.PATCH.1
PRODOS.0
PRODOS.1
PRODOS.2
PRODOS.3
PRODOS.4
(1) Umstellung von ProDOS auf Laufwerke mit höherer Kapazität; (2) Heft 11/85, S. 29; (3) II+ (mit LC) oder IIe; Ehring-Controller oder Kompatibler (4) ProDOS 1.0.1 und FILER 1.0.1; (5) s. Heft

T.PRINT.USING
PRINT.USING
T.PRINT.USING.G
PRINT.USING.G
PRINT.USING.DATA
PRINT.USING.G.DATA
PRINT.USING.DEMO
(1) Formatierte Ausgabe von Zahlen als Applesoft-Erweiterung; (2) Heft 11/85, S. 49; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) RUN PRINT.USING.DEMO (6) PRINT.USING.G gibt führende Null bei negativen Zahlen (-0.0 bis -0.9) aus

T.REF.TEST
REF.TEST
(1) Hilfsprogramm zur Überprüfung von Zeilenverweisen in Applesoft; (2) Heft 11/85, S. 55; (3) II+, IIe oder IIc; (4) DOS 3.3 oder ProDOS; (5) BRUN REF.TEST; zu prüfendes Applesoft-Programm laden; dann & eingeben

mMm wants you!

Gewinnen Sie eine Reise mit **mini Micro magazin** nach Las Vegas

Erleben Sie professionelle Computertechnik hautnah.
Erfahren Sie die Entwicklungen professioneller Computertechnik von morgen.
Treffen Sie kompetente Entscheidungshelfer in Sachen 'Systemintegration von Mini- und Mikrocomputern'.
Wie? Mit etwas Glück direkt vor Ort.
Bei einer Reise (Flug und Hotel) nach Las Vegas zur NCC vom 16. – 19. Juni 1986, die Sie mit uns gewinnen können. Ganz bestimmt aber durch die Lektüre von **mini Micro magazin.**
Schicken Sie am besten noch heute den Coupon. Es lohnt sich in jedem Fall.

adressieren an:
mini Micro magazin
Postfach 10 28 69
6900 Heidelberg

☒ **JA** **ich möchte an der Verlosung Ihrer Reise zur NCC nach Las Vegas im Juni 1986 teilnehmen. Schicken Sie mir in jedem Fall die Erstausgabe von mini Micro magazin zum Kennenlernen. Kostenlos und unverbindlich.**

Vorname _____ Name _____

Firma _____ Funktion _____ Abteilung _____

Postfach/Straße _____ PLZ/Ort _____ Telefon _____

(Die Verlosung findet unter Ausschluß des Rechtsweges statt. Der Gewinner wird kurz nach der „Systems 85" schriftlich benachrichtigt. Von der Verlosung – die unabhängig von irgendeiner Bestellung ist – sind die Mitarbeiter der Verlagsgruppe Dr. Alfred Hüthig Heidelberg, München, New York ausgeschlossen.)

Hüthig PUBLIKATION

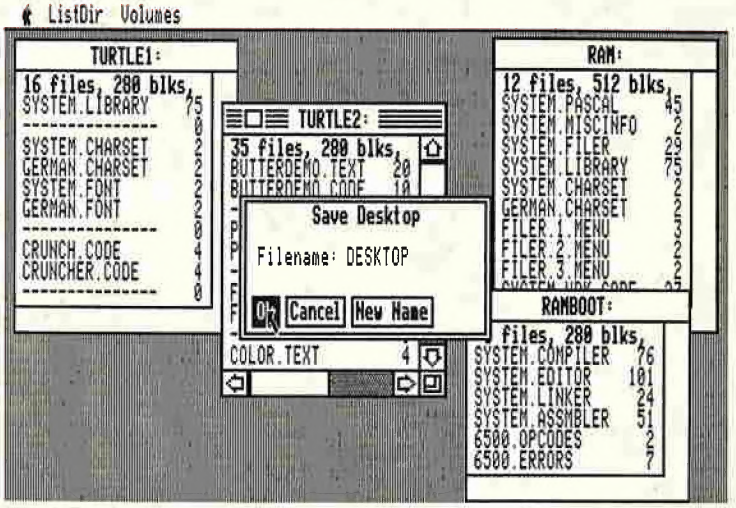
63-1/85

In Vorbereitung

TurtleGraphics-Library-Paket

von Dieter Geiß

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



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

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

Im einzelnen bietet das Paket folgende Möglichkeiten:

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

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg